2016

Contributions to Cryptographic Solutions towards Securing Medical Applications

Clementine Gritti
University of Wollongong

Recommended Citation

Gritti, Clementine, Contributions to Cryptographic Solutions towards Securing Medical Applications, Doctor of Philosophy in Computer Science thesis, School of Computation and Information Technology, University of Wollongong, 2016. https://ro.uow.edu.au/theses/4891

Contributions to Cryptographic Solutions towards Securing Medical Applications

Clémentine Gritti

Supervisor:
Professor Willy Susilo
Co-supervisor:
Doctor Thomas Plantard

This thesis is presented as part of the requirements for the conferral of the degree:

Doctor of Philosophy in Computer Science

The University of Wollongong
School of Computation and Information Technology

September 2016

Declaration

I, Clémentine Gritti, declare that this thesis submitted in partial fulfilment of the requirements for the conferral of the degree Doctor of Philosophy in Computer Science, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Clémentine Gritti
February 7, 2017

Abstract

Medical records have been moving from paper-based systems to electronic form in recent decades. This evolution to electronic health records (EHRs) brings new benefits and possibilities for healthcare providers, physicians and patients. Involved users can easily and flexibly deal with EHRs: they can broadcast and share the data amongst themselves rather than share on an individual to individual basis. The data can be moved from limited local storage systems at hospitals to externally hosted systems which enable multiple parties to access and maintain these records. However, with the change arise new practical, legal, ethical and financial challenges. EHRs contain sensitive personal medical information, and thereby demand that integrity and confidentiality are assured. Nevertheless, EHR-based systems improve individual outcomes and cut implementation costs. Stored data has to be accessible only to authorised users and always available especially in the case of emergencies.

EHR-based access and storage is a wide topic with numerous issues, including security and privacy concerns as well as efficiency and practical matters. At a first sight, one would look to design a solution that solves all of these problems. However, it is difficult to find a satisfactory solution with strong user autonomy and guarantees confidentiality along with flexibility and computational efficiency. Therefore, we propose dividing the general EHR-based context into subcases and studying each individually, defining them through concrete scenarios and to present specific solutions. We believe these solutions will be more effective in both security and utility compared to approaches which look at the EHR environment as a whole and aim to deliver a single solution.

Access control and storage services are the two main categories we studied. On one hand, we focus on the data accessibility. Since privacy of both user identities and data must be guaranteed and there is a threat of compromise by malicious actors, we have to ensure that only authorised users can access and manipulate EHR contents. On the other hand, we concentrate on storage of the data. We want to enable authorised users to be able to upload EHRs to cloud servers, selectively request access to stored data, and finally update and selectively share EHRs with other authorised people. Because it is difficult to design a single fully secure and effective EHR-based system which handles all of these actions, we divide the problem into different situations. We adopt an assumption that cloud servers are not fully trustworthy and design accordingly. This increases flexibility for healthcare providers when selecting a cloud service provider.

In this thesis, we outline various realistic scenarios, focus on their functional, security and practical requirements, and we then propose cryptographic primitives to address the requirements and issues.

We first present two primitives which involve broadcast encryption with membership and certificate-based broadcast encryption to enable secure and efficient broadcast and sharing of EHRs among the involved users. The first primitive allows hospital staff members authorised by a medical institute to access EHRs encrypted by the hospital. The second primitive enables staff members authorised by the hospital and holding valid certificates delivered by health legislators to access EHRs.

We then propose a primitive involving certificate-based encryption with keyword search to enable secure and efficient access and retrieval of EHRs stored on cloud servers. This primitive allows hospital staff members to search for EHRs stored on a cloud server using a trapdoor that embeds a keyword describing the contents of the records and a valid certificate.

We also design two primitives involving on-line/off-line ciphertext-policy attribute-based proxy re-encryption and ciphertext-policy DNA-based encryption to securely address patient privacy in an efficient manner by reducing the computation and communications resources needed. The first primitive enables the hospital to pre-encrypt an EHR regarding credentials and lets the patient finalise the encryption using other credentials. A staff member recovers the EHR if and only if s/he satisfies at least the patient's credentials. The second primitive considers DNA sequences for their uniqueness and closeness. A first patient encrypts his/her EHR using his/her DNA sequence and a second patient can retrieve the EHR if and only if his/her DNA sequence is close enough to that of the first patient.

Finally, we propose the primitive which involves dynamic provable data possession with public verifiability and data privacy to enable secure and efficient management of EHRs in cloud computing. This primitive allows hospital staff members to upload and update non-encrypted EHRs to a cloud server. A third party auditor is required to check that the cloud server correctly stores the EHRs by regular auditing.

Acknowledgements

I wish to thank my supervisor Willy Susilo and my co-supervisor Thomas Plantard for the time and energy they invested in our collaboration, as well as for their advice and encouragement. I wish to thank my other collaborators for the works comprising this thesis: Kaitai Liang, Duncan S. Wong and Rongmao Chen.

I also wish to thank my partner Pierre-Jean Coltat for his support. I finally wish to thank all the people that I met in Australia, for their contribution to this amazing and adventurous experience. In particular, I wish to thank the Illawara Speleogical Society for their help when my partner and I arrived in Australia, the Sydney University Mountaineering and Rock Climbing Club for their friendship and awesome climbing trips in the Blue Mountains and the so-called Saufen group for their friendship and fabulous lunches on the campus.

List of Publications

Publications listed below are related to this thesis:

1. Clémentine Gritti, Willy Susilo, Thomas Plantard, Kaitai Liang, and Duncan S. Wong: *Empowering Personal Health Records with Cloud Computing: How to encrypt with forthcoming fine-grained policies efficiently*. In Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume 5, pages 3-28, 2014.

2. Clémentine Gritti, Willy Susilo, Thomas Plantard, and Khin Than Win: *Privacy-Preserving Encryption Scheme using DNA Parentage Test*. In Theoretical Computer Science (TCS), volume 580, pages 1-13, 2015.

3. Clémentine Gritti, Willy Susilo, and Thomas Plantard: *Efficient File Sharing in Electronic Health Records*. In Proceedings of the 11th International Conference on Information Security Practice and Experience (ISPEC 2015), pages 499-513, 2015.

4. Clémentine Gritti, Willy Susilo, and Thomas Plantard: *Efficient Dynamic Provable Data Possession with Public Verifiability and Data Privacy*. In Proceedings of the 20th Australasian Conference on Information Security and Privacy (ACISP 2015), pages 395-412, 2015.

5. Clémentine Gritti, Willy Susilo, Thomas Plantard, Kaitai Liang, and Duncan S. Wong: *Broadcast Encryption with Dealership*. In International Journal of Information Security (IJIS), volume 15, pages 271-283, 2016.

6. Clémentine Gritti, Willy Susilo, and Thomas Plantard: *Enabling Secure Authorization in Electronic Health Record with Certificate-Based Encryption with Keyword Search*. In Journal of Internet Services and Information Security (JISIS), volume 6, pages 1-33, 2016.

7. Clémentine Gritti, Rongmao Chen, Willy Susilo, and Thomas Plantard: *New Dynamic Provable Data Possession Protocols with Public Verifiability and Data Privacy*. Submitted

Other publication that is not included in this thesis:

1. Clémentine Gritti, Willy Susilo, and Thomas Plantard: *Logarithmic Size Ring Signatures without Random Oracles*. In IET Information Security, volume 10, pages 1-7, 2016.

Contents

Chapter 1

Introduction

Electronic health records (EHRs) are the digital form of medical information. They have been emerging in the last decades as the evolution of traditional paper records in the healthcare environment. EHRs offer the possibility of sharing the medical information among various users, such as healthcare providers, physicians and patients, as well as to archive them over a long period of time. Ideally, EHR systems should be designed to benefit clinical, organisational and societal outcomes, such as reducing frequency of medical error, decreasing financial costs and improving the general population health [154].

The medical information contained within EHRs is personal and sensitive, and thus EHRs have to ensure that availability, integrity, confidentiality and acknowledgement are guaranteed [200]. EHR-based models have been expanded nationally with the aim to protect patient autonomy, privacy and confidentiality. However, they have irredeemably brought practical, legal, ethical and financial challenges [107, 211]. In particular, analysing the financial costs for implementing EHR systems in primary care remains essential despite EHR systems noticeably improving the quality of patient care and reducing medical errors [234, 85].

EHRs should integrate personal medical information from different sources, and provide a complete and correct personal health and medical summary through the Internet and on portable electronic devices. For instance, EHRs could be fragmented and distributed to several healthcare related sites to address the needs and obligations of the involved users. Designing EHR structure and ownership of EHR information impacts on patient access control and privacy. Therefore, carefully defining a framework is important to control threats to accessibility and security, and to maintain the privacy of patients [151]. EHR information exchange systems allow users to autonomously store, access and share medical information recorded in EHRs. In addition to medical information exchange, functionalities offered by EHR systems include clinical decision support and structured data entry [194].

One implementation option for EHR systems involves a cloud-based environment. Cloud computing enables EHR systems to transfer and store the medical information on servers hosted by a third party which are accessible via the Internet. A cloud solution can deliver gains in flexibility and reduce operational costs. However, a cloud environment increases patient privacy challenges because the cloud infrastructure and service provider should not be fully trusted.

In each chapter, we provide scenarios describing specific recurrent situations that appear in EHR systems. To help with interpreting the figures, we provide lists of users and objects to illustrate the scenarios. Figures 1.1 and 1.2 depict these lists.

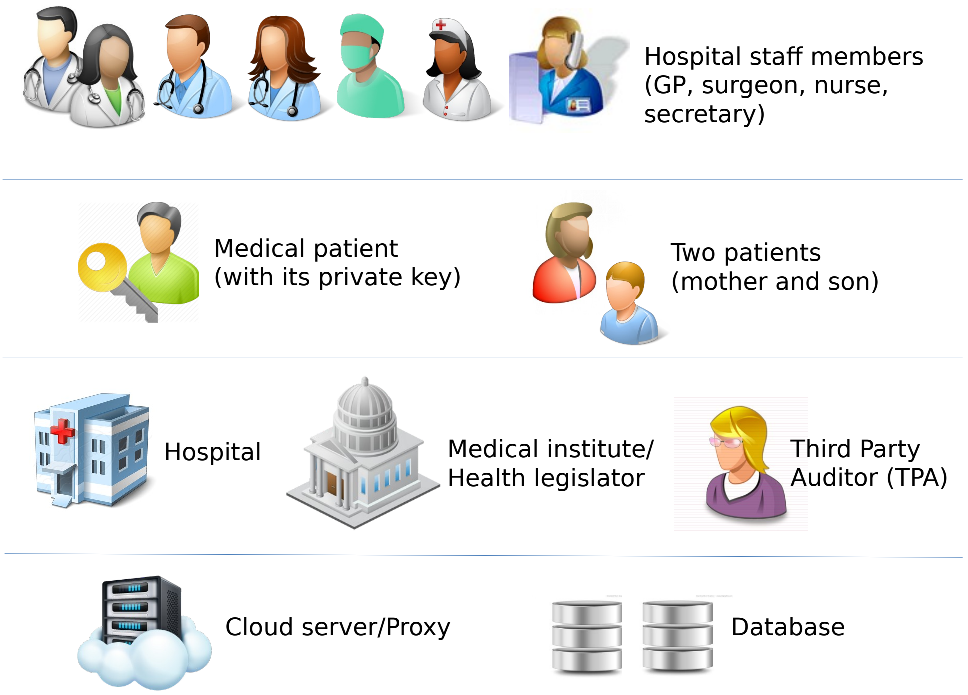

Hospital staff members (GP, surgeon, nurse, secretary)
Medical patient (with its private key)
Two patients (mother and son)
Hospital
Medical institute/ Health legislator
Third Party Auditor (TPA)
Cloud server/Proxy
Database

Figure 1.1: List of Users. Let us consider first the hospital staff members: they can be either medical staff members (e.g. general practitioners (GP), surgeons, anesthetists, gynaecologists, etc), or paramedical staff members (e.g. nurses, physiotherapists, etc), or administrative staff members (e.g. secretaries, accountants, IT technicians, etc). We also suppose the existence of medical patients: Bob that holds a public and private key pair, and Alice and her son Charles. Then, several places come into the scenarios: the hospital and the medical institute (seen as a health legislator). The hospital is the place where the users act and the medical institute plays the role of group manager and/or certifier (i.e. distributing public and private key pairs and/or the certificates respectively). In addition, we take into account a third party auditor (TPA) that can help the patients during some processes to be determined, and a proxy or a cloud server that steps in the protocols as an interface. In particular, the cloud server is linked to a database as its local storage.

1.1 Scenarios

At first glance, storing sensitive information outside the owner's local storage seems hazardous since information leakage might compromise the patient privacy and interruption to communication networks might threaten the owner's accessibility. Medical information includes patients EHRs as well as administrative and financial documents owned by a hospital, all of them being stored on an external server (such as a private cloud server) and thus are subject to leakage risks. Without losing meaning, we employ the expression *medical documents* to denote all the documents held by the hospital, that is the health documents (containing medical information), the administrative documents (containing

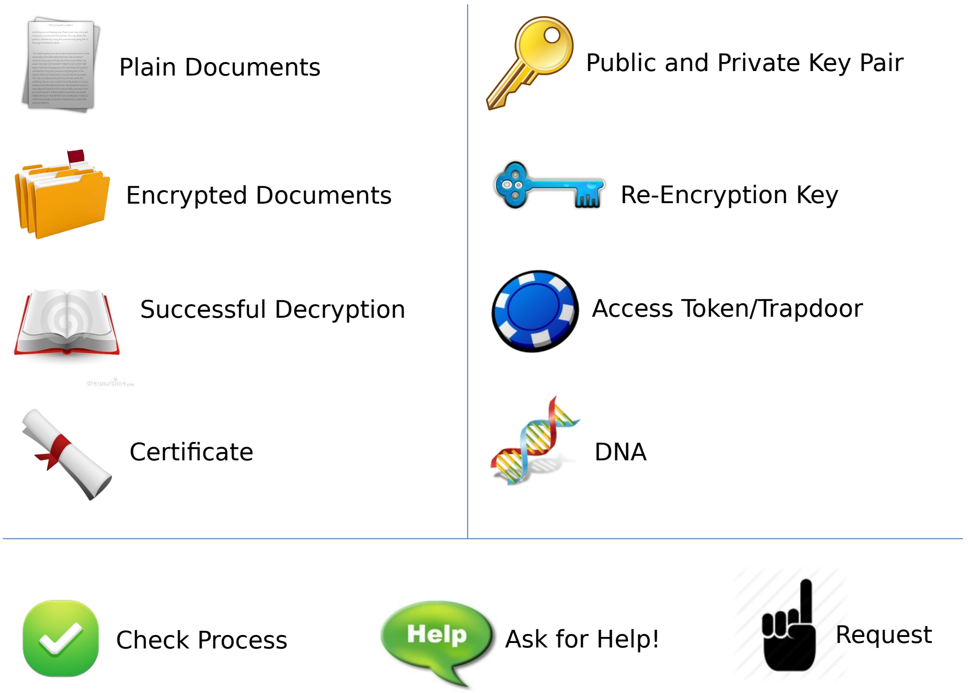


Figure 1.2: List of Objects. Electronic health records (EHRs) can be medical and personal documents. They can be found in: 1) the plain form (a user can see them in clear); 2) the encrypted form (a user has encrypted them using some parameters to be determined), that we call a ciphertext. A user tries to decrypt a ciphertext and can meet one of the following results: 1) the user retrieves the original plain EHRs, meaning that the decryption is successful; 2) the user retrieves a fake EHRs, meaning that the decryption fails. Each of the main characters involved in the protocols will receive a public and private key pair. Moreover, a user may receive the following items from another party: 1) a certificate (that can represent a license for instance); 2) an access token or a trapdoor embedding some specific elements such that a keyword or a date (e.g. an object containing the security credentials for a login session and identifying the user and his/her privileges); 3) a re-encryption key in order to change the decryption rights. DNA can be seen as a key since each user has his/her own DNA sequence that makes this key unique (except for some cases to be specified). Finally, some symbols illustrate actions as follows: a party can ask to check some process done by another party; 2) a party may ask for help to another party (for instance, a party does not have enough resources to do the task, thus asks to another party that is more able to do it); 3) a party can forward a request to another party.

financial and organisational information), and any other kinds of documents related to the hospital.

Studying the storage and access management of EHRs is a wide topic: it is difficult to define a single scenario and then design a solution that addresses all of the situations that patients, medical institutes and hospitals can encounter when dealing with EHRs. Indeed, we require the solution to be an EHR-based protocol that is efficient, practical, secure and addresses privacy matters.

At first sight, we can see an EHR as a simple message that can be encrypted and decrypted indefinitely. This means that public-key cryptographic primitives and its derivatives can simply be applied in this case. We can also combine all the EHRs, and hence, treat them as big data (i.e. data sets that are large or complex and for which traditional data processing applications are inadequate). Studies on big data are widespread in the literature. Therefore, if we consider EHRs as basic data (i.e. one EHR is a simple message and all the EHRs together form a big data set), then we observe that numerous existing cryptosystems have been proposed as solutions for such a problem. However, EHRs are a particular kind of data that contains highly sensitive personal information which should be kept secret from most people. It introduces specific problems that need to be solved one by one. These problems relate to preservation of the privacy and soundness of the EHR-based protocols.

In this thesis, we propose splitting the general EHR-based context into subcases that are related to particular problems, and present solutions for each of them. By choosing this study direction, we are able to design satisfactory cryptographic primitives, such that computational and communication costs are not a bottleneck and security and privacy requirements are achieved.

We start by dividing the wider EHR-based context into subcases by considering the format in which EHRs are stored and exchanged among users. We examine the two following cases:

1. The medical documents are encrypted before being uploaded to a cloud server, so that the server gets no information about the contents being stored. Therefore, data security and privacy are ensured. However, this design is cumbersome as it requires an encryption/decryption process for each document, especially each time the contents have to be updated (i.e. added, deleted or modified).

2. The medical documents are uploaded to the cloud server in plain, meaning that the content is accessible from the cloud server. This is easy for the data owner to add, delete or modify the content of the document, since s/he does not need to download it to a local storage; instead s/he can request the cloud server to do the task. To ensure that the cloud server stores and updates the documents as expected, audit processes are required. This means that any unexpected behaviour can be detected. For instance, if the cloud server adds, deletes or modifies parts of the content without being asked to do so. The audit processes should be conducted by an external party since the data owner is unlikely to have sufficient capability to do this adequately.

We propose cryptographic primitives that implement each of these cases in the electronic Health (e-Health) context. The first case ensures that EHRs are hidden from non-authorised users' view, however computation and communication costs might increase. The second case allows data owners to easily manage their data stored on a cloud server, however the content can be learnt from the cloud server. Thus, in this particular case, we also enable data owners to be able to externally check the integrity of this data.

Storage issues are not the only problems we encounter, the other class of problems relates to access control. Since we want to restrict access to EHRs only to users that are explicitly authorised, how should we control for non-authorised users? Here, another two problems emerge. Firstly, we must avoid the case where a non-authorised user manages to intercept some of the stored data and discovers its contents. Secondly, we do not want the cloud server to be able to manipulate the data without being detected.

To describe the access control framework, we need only to consider the situation where the documents are stored in their encrypted form on the cloud server. We have to ensure that only authorised users have access to the medical information, and so are eventually able to retrieve it in plain after decryption. Since the medical documents are encrypted, their contents are inaccessible to non-authorised users and to the cloud servers. To guarantee access control efficacy, we have decided to provide users with private keys which correspond to public keys that were used to encrypt the medical documents on their behalf. However, we also decided that this is not sufficiently robust since users keys can be compromised, lost or stolen, and thereby used to gain unauthorised access to EHRs. We therefore further divide the access control problem into four subcases. Each of which uses one or two special features along with a private key in order to access EHRs. Additionally, these feature options can be combined to further enhance access control.

1. Personal and professional credentials are required to encrypt and decrypt medical documents. For instance, personal credentials include details related to the identity of the user, while professional credentials contain information on the user's position within the organisation. The document may only be decrypted with credentials matching those which were used to encrypt it.

2. Certificates are delivered to users by a trusted authority. The document may only be decrypted by a user with a valid certificate.

3. A group of authorised users are nominated during the encryption of a document. The document may only be decrypted by one of the nominated users.

4. Some metadata related to the encrypted medical document is asked to the user attempting decryption. This could be a keyword that briefly describes the contents of the document. The document may only be decrypted by correctly specifying the metadata.

We will present a design using cryptographic primitives to implement these features based on encrypted medical documents. Some primitives will implement a single feature while others will address more than one.

1.2 Context Description

The following users contribute into the storing, accessing and sharing processes: a medical institute and possibly other health legislators (for instance, the government), a hospital and its staff members, and a private cloud server operated by the hospital. Possibly, patients might intervene into the processes.

Hospital staff members include administrative staff (e.g. secretaries, accountants, IT technicians, etc), paramedical staff (e.g. nurses, physiotherapists, etc) and medical staff (e.g. general practitioners (GP), surgeons, anesthetists, gynaecologists, etc). Throughout this thesis, a practitioner is either a member of the paramedical group or the medical group.

Medical institutes and other health legislators deliver public and private key pairs and certificates to hospital staff of a given hospital with their authorisation to let them practise. A legislator is responsible for giving rights (e.g. access) and privileges (e.g. credentials) to hospital staff.

The hospital maintains a preferably private cloud server for storing EHRs. Depending on the function being described the EHR may or may not be encrypted.

When practitioners or administrative staff are authorised, they are able to upload (encrypted or non-encrypted) medical documents to the cloud server, access stored ones and retrieve original ones. Medical documents can be shared amongst hospital staff only when previously specified requirements are met. The requirements include the access rights, the credentials and other characteristics that a staff member satisfies personally and/or professionally.

Hospital staff must satisfy controls before being granted access to a requested medical document. More precisely, staff should be first identified (for instance, they provide their name, their working registration number, etc), then authenticated (for instance, they use their private key and/or their certificate(s)) and finally authorised (for instance, their right to access the medical document is verified). Should a clinician, Dan, wish to access a medical document stored on the cloud server, he must satisfy three steps:

1. *Identification:* where Dan says who he is (e.g. with a login user name);

2. *Authentication:* where Dan proves his identification given in the first step (e.g. with a password or a PIN number);

3. *Authorisation:* where Dan's access rights are verified.

Patient EHRs and other documents held by the hospital contain sensitive information that must be kept private from non-authorised users. Specifically, only an authorised user can be permitted to access and discover their contents. The patients may be granted restricted access to some of their medical information. Consider a patient, Alice, being able to modify her EHR, and if she has a blood test to indicate illicit drug use, she should be prevented from being able to modify the results for legal reasons.

Access controls and restrictions are not the only issues to take into account. Availability and access to the cloud server cannot be guaranteed. The following scenarios are examples where this might be a problem.

- A patient may have poor or no access to the Internet. Whether transitory due to inadequate or failed technology, or because of socio-economic reasons. There must be a means for the patient to deal with his/her EHR in these circumstances.

- The hospital will from time to time encounter failures and interruptions with network infrastructure, theirs and upstream, which will cause communication with the cloud server to be unavailable.

One of our goals is to avoid these issues as much as possible. For instance, we will propose allowing users to perform most of the data encryption processes off-line, and then finalising them on-line when possible.

We now describe the trust level assigned to each user. Since the cloud server is not assumed to be fully trusted, one solution is to encrypt the medical documents before uploading to the cloud server, in order to hide their contents from its view. Another solution is to leave the medical document in plain, and regularly audit the cloud server to verify that medical documents are correctly stored and that no unrequested updates (additions, deletions, modifications) have occurred. For instance, the cloud server might delete medical documents that are not often requested in order to save storage space and so, might propose its services to other hospitals and increase its business profits.

Hospital staff must be restricted from accessing medical documents for which they are not authorised. For instance, a physiotherapist does not need to know that his/her patient suffered from depression few years ago. Access rights and privileges control the access to the medical information for each hospital staff member.

Medical institutes and other legislators are fully trusted; indeed they are responsible for delivering public and private key pairs and possibly certificates to the hospital staff members. This means that they hold secret components and should carefully deliver them to the intended recipients. Public and private key pairs and certificates are randomized to ensure that these elements are unique. If two hospital staff members were to receive the same certificate and do not have the same access rights, then one of them will be able to access more information than allowed.

In Figure 1.3, we summarize the functions of the above users in the e-Health context.

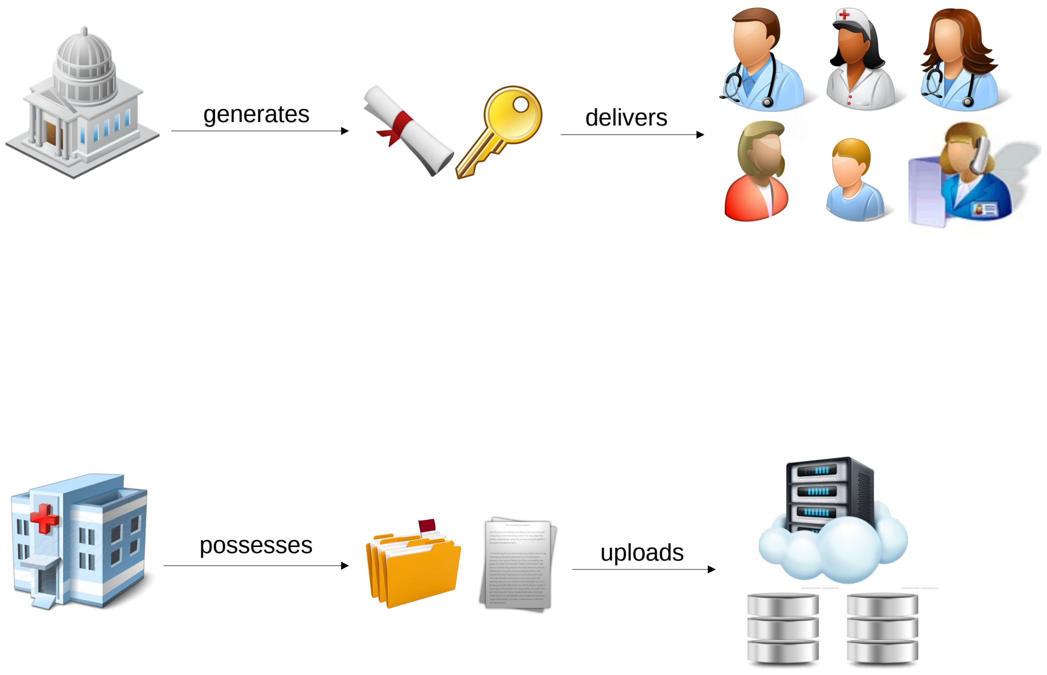


Figure 1.3: Occupations of the Users. The medical institute generates the certificates and/or the public and private key pairs and distributes them to all hospital staff members (and possibly to the patients). The hospital, on behalf of the hospital staff, gets a large amount of medical information, such as the patients EHRs, administrative documents, financial documents, and so on. The hospital uploads them, either in plain or encrypted, to a private cloud server that holds enough database resources.

1.3 Existing Work

1.3.1 Published Literature on Health Records

Ferreira et al. [83] reviewed material published in the literature on access control in healthcare between 1996 and 2006. This review reveals that most of the access control systems are just studies or prototypes in which healthcare professionals and patients did not participate in the definition of the access control policies, models or mechanisms. In addition, the authors noticed obstacles to successful integration within the healthcare practice. These obstacles affect the security and costs due to time and effort, and may be induced by relational and educational issues.

Thereafter, Alemán et al. [7] gave a literature review on security and privacy in EHRs. They selected 49 papers that followed standards or regulations related to privacy and security of EHRs, and enhanced the promulgation of directives concerning security and privacy in EHR systems. The authors observed that the most recurrent access control model is role-based access control (RBAC): 27 papers focus on this model. In addition, they noticed that the standards and regulations should be further adopted in the existing access control models and security of EHR-based systems should be enhanced.

More recently, Rezaeibagha et al. [185] reviewed 55 selected studies on security and privacy in EHRs, published between 1998 and 2013. Highlighted features were system and application access control, compliance with security requirements, interoperability, integration and sharing, consent and choice mechanisms, policies and regulation, applicability and scalability and cryptography techniques. The authors noticed the importance of materials that include mandated access control policies and consent mechanisms that provide patients' consent, scalability through proper architecture and frameworks, and interoperability of health information systems, to EHR security and privacy requirements.

1.3.2 Access Control

We enable a user to carry out specific tasks and actions on EHRs by giving him/her access to these EHRs. Since EHRs contain sensitive personal information, controlling their access is crucial. Medical information should be accessible to authorised users as well as be available when needed in life-critical situations (e.g. emergencies). Along with basic access controls, various derivatives, such as discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC) can be found in the literature.

Bos [44] first used digital signatures to exchange medical information. Thereafter, Mavridis et al. [153] proposed a certificate-based encryption (CBE) protocol where the access to medical data is controlled through a public key setting. Users require their credentials to access the medical data and receive three certificates, namely identity certificates for authentication, attribute certificates for authorisation and access-rule certificates for propagation of access control policy. Meanwhile, Ueckert et al. [220, 219] proposed granular access control mechanisms where patients grant reading and/or writing access to their EHRs. A special case of read access is enabled for emergencies. Ruotsalainen [199] proposed a cross-platform model to allow secure EHR access and sharing. The platform provides automatic security negotiation services to enable access to EHRs inside connected domains.

Røstad and Edsberg [193] studied the access logs from a EHR system with extensive use of exception-based access control. The authors observed that exception mechanisms were used too frequently. They suggested designing a more structured and fine-grained logging in order to analyse access logs efficiently and to learn how to reduce the need for exception-based access. Then, Win et al. [238] studied the access control security of EHR-based systems and recommended users to require PIN, password and credentials to access these documents. Dekker and Etalle [69] presented an access control system where the authentication process is performed in two steps: an authorisation request or an authentication credential corresponding to a logical formula is first given, and the authorisation or authentication decision corresponding to a proof of the formula is then created. Meantime, France et al. [87] designed an access control protocol to exchange medical information using digital signatures and electronic identification cards. Al-zharani et al. [6] gave an EHR access control model such that authentication is possible using digital signatures, login and password.

Afterwards, Alhaqbani and Fidge [8] reviewed the security models for discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC). By using a case study, they showed that none of the aforementioned access controls is sufficient in a federated healthcare environment, in terms of security data level. They also demonstrated that this level can be reached by combining the three access controls. In a parallel work, Alhaqbani and Fidge [9] proposed a privacy-preserving access control system using pseudonym identifiers. Authentication is possible based on digital signatures and public key infrastructure. A special emergency access policy is agreed between the patient and health provider.

Later, Benaloh et al. [28] constructed an EHR privacy-preserving access-control based system. The patients select both their medical documents and the users to share these documents with. Huang et al. [119] presented a privacy-preserving access control scheme where authentication is based on digital signatures, logins and passwords. Narayan et al. [171] used an attribute-based encryption (ABE) framework to implement a system that preserves EHR privacy. Their system requires mutual user authentication such that each user is associated with a username that is needed to define the access policies that are managed by the patients. The patients have the control over the medical information access and hospital staff members can delegate access to other hospital staff members.

More recently, Haas et al. [111] designed an access control system where privacy policies are defined and checked by the patients. Access policies are bypassed in the case of emergencies. Quantin et al. [182] implemented medical search engines using pseudonymised patient identity to obtain a distributed database between healthcare institutions. Jian et al. [125] proposed an access control protocol for Taiwanese patients and hospital staff members. In order to access their EHRs, the patients are given an initial password and non-authorised users cannot view the EHRs. Gajanayake et al. [90] combined the four existing access control models (DAC, MAC, RBAC and purpose-based access control (PBAC)) to obtain a new access control model that satisfies patient privacy requirements, confidentiality of private information and the need for flexible access for health professionals for EHRs.

Futhermore, Khan and McKillop [129] suggested a patient-centric access control in order to enforce patient privacy preferences. Such solution permits medical data exchanges across several systems under different administrative domains. The authors addressed the problem by using logic such that each access control decision taken at system level can be automated and independently verified for validity and correctness. Flores Zuniga et al. [86] approached the problem of exchanging medical information by designing an ABE system. Confidentiality of patient information during the exchange of EHRs among healthcare providers is preserved by reinforcing the access policies, while non-authorised access to this sensitive information is reduced.

Role-Based Access Control. One of the methods widely spread in the literature is the role-based access control (RBAC). Such access control enables the regulation of access to devices or networks according to the roles of users within an organisation. For instance, in the EHR-based context, the organisation can be a hospital and roles can be defined based on the positions in the organisational hierachy including the privileges and responsibilities within the hospital.

While basic access controls accord and withdraw user access on a rigid and static basis, RBAC authorises and regulates users to perform tasks and actions, regarding their roles. Observe that these roles are easily determined, modified or removed with the evolution of the organisation, without the needs to update them for each user.

Motta et al. [165] developed a contextual RBAC authorisation model for EHRs. This model is based on an organisational role-tree hierarchy that allows authorisation inheritance, positive and negative authorisations, static and dynamic separation of duties based on weak and strong role conflicts. Tsiknakis et al. [218] suggested a RBAC system to securely access and share life-long multimedia EHRs. Meanwhile, Reni et al. [184] focused on the Italian case for RBAC-based EHR systems. In case of emergencies, a read only option is available. A chief medical officer (CMO) defines the roles, and patients and the CMO grant access to EHRs.

Thereafter, Yu and Chekhanovskiy [244] suggested a RBAC protocol using SmartCard and EHRs. Authentication is possible using digital signatures and public key infrastructure. The patients grant access to their EHRs. In a national context (United Kingdom), Slevin and Macfie [212] studied the applicability of a RBAC system within an existing medical database in the Oncology Department at St. Bartholomew's Hospital in London. Moreover, Agrawal and Johnson [2] proposed an efficient RBAC system where patients grant access permissions regarding policies defined by health organisations.

Subsequently, Riedl et al. [191, 190, 189] presented RBAC systems where roles are defined by health providers and authentication is performed with a security token that contains the access keys. Bouwman et al. [46] studied right management for EHR systems based on RBAC. Users are authenticated by their digital certificates. A specific role is defined for emergencies. Meantime, Røstad [192] presented a RBAC system where roles are defined in two different ways: either as system roles such as patient, provider, researcher and so on, or as user roles that are determined by the user him/herself. Moreover, the patients decide which users can access their data. In addition, Kahn and Sheshadri [128] suggested a RBAC protocol in a digital environment. The roles are determined by healthcare organisations and include employees, administrative staff, health (medical and paramedical) staff and information technology (IT) staff. Users are identified individually before accessing medical information. Choe and Yoo [61] suggested a multi-agent architecture based on web services as a RBAC and exchange of the medical information. Each hospital has the capability to define its own security policies and assign roles.

Later, Daglish and Archer [67] reviewed the security and privacy problems in access control-based systems. They gave RBAC schemes where roles are given beforehand to users. The patients are able to refine roles' definitions and the access to their data. Two access control methods are suggested: either authentication through physical location or combination of web and trusted organisations' certificate. Van der Linden et al. [222] gave a inter-organisational RBAC protocol. They assumed that either patients implicitly give their consent for access permissions or they explicitly determine which users cannot access their EHRs.

Then, Elger et al. [76] focused on the case of secondary, cross-institutional clinical research and proposed authentication through tokens in their RBAC. Hembroff and Muftic [113] implemented a RBAC system called SAMSON (secure access for medical smart cards over networks). Authentication is done given a health card PIN and the patient's fingerprint. They also adapted their scheme in case of emergencies: the access is allowed by using a PIN of a staff member's card as well as the patient's fingerprint. Moreover, Zhang and Liu [248] presented a RBAC system such that roles are related to the position of the hospital staff members. The use of anonymous digital credentials allows users to be authenticated for accessing medical information stored on cloud servers. Data exchange is done through secure connection, such as SSL (secure sockets layer), TLS (transport layer security) or IPsec (internet protocol security). Sun and Fang [216] proposed a distributed EHR system based on RBAC where roles are defined regarding the hospital staff members' positions and are used for access delegation, and based on proxy signature to achieve fine-grained access control. Meanwhile, Al Faresi et al. [5] gave a RBAC protocol created regarding the HIPAA privacy regulations. EHR access is managed by the patients and patients' access preferences may be controlled by medical and paramedical staff members. Ardagna et al. [11] designed a RBAC system such that access control is done using policy spaces. The patients are able to access their EHRs as well as to manage the access control by a notification mechanism. Access control models are bypassed in case of emergencies. Jafari et al. [121] constructed a scheme based on RBAC and licenses that indicate access permissions. Roles comprise leader, researcher and assistant. Patients must consent to give their medical information for research purposes. In addition, Sucurovic [215, 214] designed a hierarchical RBAC protocol that allows decomposition of policy engines into components. Authentication and authorisation are done through digital signatures, public key certificate and RSA cryptosystem. There are several hierarchies, such as of roles, professions, regions, and so on.

Afterwards, Neubauer and Heurix [172] presented a RBAC scheme where authentication is done through login and password and roles are defined beforehand. Patients fully control their EHRs since they are able to create data access authorisations and grant full access rights. Horvath et al. [116] designed the DEDUCE guided query tool based on RBAC. Roles are established in the EHR system beforehand. Their tool authenticates users using Microsoft Active Directory accounts. Moreover, Lemaire et al. [138] created a RBAC system to handle the situation of accessing medical information of patients with traumatic brain injury. Roles are defined in the EHR system beforehand and certificate authentication is performed upon log-in.

Recently, Zhang et al. [249] presented a role-based and time-bound access control model (RBTBAC). This model embeds a privacy-aware and dynamic-key structure focusing on the consistency of access authorisation (including data and time interval) with the activated role of user, that enables a role-based privacy-aware access and the management of EHR data. It also uses a time-tree method for generating time granule values, offering fine granularity of time-bound access authorisation and control.

Situation-Based Access Control. Pelega et al. [178] defined a new access control model called situation-based access control (SitBAC). This model comprises scenarios where patients data access is permitted or denied, by using an object process methodology. The situation-based model creates a pattern consisting of the following users and their features: data requestor, patient, EHRs, access task, legal authorisation and response, along with their properties and relations.

1.3.3 Cloud Computing

Cloud computing in an EHR-based context is an important topic since the medical records are in the process of being stored digitally on cloud servers. Cloud-based EHRs enable medical information to be stored externally to the hospital's IT infrastructure and shared and updated by users from any location. Compared with traditional EHR storage systems cloud computing promises higher levels of availability and accessibility. The EHRs are practically available and accessible to all users all the time.

Jafari et al. [122] proposed a patient-centric digital rights management protocol that protects the privacy of the EHRs stored on a cloud storage from the service provider. Meanwhile, Stingl and Slamanig [211] approached the issue of storing highly sensitive health data in the cloud and protecting patients' privacy based on anonymous communication and authentication, anonymous yet authorised transactions, and pseudonymisation of databases.

Then, Huang et al. [118] combined identity-based encryption (IBE) and attribute-based encryption (ABE) to obtain a cloud computing based system that satisfies data privacy, fine-grained access control and scalable access between different clouds. The authors also overcame the issue of improper data access caused by a user with multiple roles and access rights to an EHR. Chen et al. [58] presented a system in healthcare clouds to enable sharing and integration of EHR information, and so provide professional medical programs with consultancy, evaluation, and tracing services and improve accessibility to the public receiving medical services or medical information at remote sites. Meantime, Chen and Hsieh [56] suggested a solution to support patient-controlled encryption and privacy-preserving keyword search. They combined ciphertext-policy ABE (CP-ABE) and public-key encryption with keyword search (PEKS) schemes to implement their solution satisfying patients' concerns in security, privacy and trust. Furthermore, Chen et al. [57] proposed a patient-centered access control in a cloud computing environments in order to store, access and share personal medical records. The authors constructed their scheme using the Lagrange interpolation polynomial framework and proved it flexible and secure. Their protocol effectively corresponds to real-time appending and deleting user access authorisation and appending and revising EHRs. Wu et al. [240] suggested a systematic access control mechanism to support selective sharing of composite EHRs aggregated from various healthcare providers in clouds. The authors ensured that privacy concerns are accommodated for processing access requests to patients' healthcare information in their system.

Thereafter, Singh et al. [210] proposed a trusted based dynamic reputation system that restricts the access of sensitive medical data stored on the cloud. This system follows ABE framework. Li et al. [144] exploited the cloud-based multi-authority ABE primitive to allow dynamic modification of access policies or file attributes, supports efficient on-demand user/attribute revocation and break-glass access under emergency scenarios. In particular, the authors focused on the multiple data owner scenario, and divided the users in the EHR system into multiple security domains that greatly reduces the key management complexity for owners and users.

More recently, Wang et al. [231] presented a new model for cloud computing, called fair remote retrieval (FRR) and based on fair multi-member key exchange and homomorphic privately verifiable tags. This model allows fairly retrieving encrypted private medical records outsourced to remote untrusted cloud servers in the case of medical accidents and disputes. The FRR construction is proved secure in the random oracle model.

1.3.4 Architecture

Van der Haak et al. [221] developed and implemented a cross-institutional architecture for EHRs in a national context (Germany). The authors first identified the specific legal requirements concerning data security and data protection of patient health data. Then, Aljarullah and El-Masri [10] proposed an EHR system for a national integration. Their system is implemented following a semi-centralised approach as an intermediate solution between the centralised architecture and the distributed architecture that has the benefits of both approaches. Khan and Sakamura [130] developed a robust authentication scheme and a hybrid access control model for healthcare informatics following the fact that authentication is an indispensable precursor to access control. Later, Bouhaddou et al. [45] focused their study in the particular example of the US department of veterans affairs (VA) such that a formal organisation within VA is responsible for helping to develop and implement standards. The authors reviewed the adoption of four standards in the categories of security and privacy, terminology, health information exchange, and modelling tools. More recently, Murray et al. [166] studied insurance portability and use of national guidelines for electronic health communication in a national context (United States).

1.4 Research Questions

We now present our research questions that will be solved through this thesis.

Broadcasting and Sharing Electronic Health Records among the Involved Users. How to enable a user to broadcast a medical document and share it with other users (possibly pre-authorised) without having to upload it to a cloud server? To be more precise, we want this user to be able to send an encrypted medical document to multiple recipients (broadcasting) without needing to upload it to a cloud server, and we want to ensure that only some of these recipients (authorised in any way by the sender) can successfully recover the medical document in plain (sharing).

The user can be either a patient that wants to share his/her EHR with his/her clinician, or can be a practitioner that wants to share medical documents with related patients and colleagues, or any other member of the hospital staff that needs to share medical information with other users.

Noting that sharing medical documents containing sensitive information must be done securely: we do not wish a non-authorised user to get any information about the contents. Broadcasting should also be carefully executed: if we wish to send the encrypted documents to a broad range of recipients, then the authorised ones should be able to successfully recover the data in plain, while the non-authorised ones must not learn anything about the data (except perhaps who the authorised users are).

Accessing and Retrieving Electronic Health Records from Data Storage Servers. How to allow a user to access and retrieve a medical document stored on a cloud server, where the medical document has been encrypted by someone else? More precisely, we want a user that asks for a medical document encrypted and stored on a cloud server to be able to get access to it and to obtain the plain version. Moreover, the cloud server should easily and quickly find the requested medical document.

The user can be either a patient that wishes access to his/her EHR, or a practitioner that needs to get access to medical documents, or any other member of the hospital staff that requires access to the medical information.

The medical documents are encrypted by the hospital staff members or the patients, and stored on a cloud server. Browsing all the documents to find the one that is requested is a burden, even if we can assume that the cloud server has huge resources to store and search. Imagine that the cloud server has to check the medical documents that it stores one by one until it finds the one than the user has requested. This situation needs to be avoided and a solution enabling efficient search must be found.

Ensuring Patient Privacy and Saving Resources in Electronic Health Records-based Storage using Limited Resource Devices and Tools. How to ensure identity privacy and data privacy of encrypted documents stored on a cloud server, while saving resources? To be more precise, we want to ensure that the privacy regarding the identity of the patient and the contents of his/her EHR are preserved while the EHR is stored on a cloud server. Meeting this requirement could have a significant computation and communications cost and we wish to keep it minimal.

The user can be either a patient that wants his/her identity and data not leaked to a non-authorised user, or a practitioner that manipulates medical documents and wishes to protect patients' and colleagues' privacy, or any other member of the hospital staff that requires to preserve the user identity and data privacy.

Observe that enhancing privacy in EHR systems leads to an increased workload. Moreover, we have to keep in mind that involved users might have limited computing and communications resources. A patient may have an old or low end personal device with limited storage or processing power or have a low bandwidth internet connection. Hospitals too can be constrained with communications infrastructure and some users have limited IT literacy. The tools must solve the identity and data privacy requirements whilst being easy to use and not requiring excessive resources.

Managing Electronic Health Records in Cloud Computing. How to allow a user to easily manage his/her medical documents stored on a cloud server? To be more precise, we want a user to be able to upload all his/her medical data to a cloud server, and be easily able to change it over time (by adding, deleting or modifying documents), while ensuring their integrity.

The user can be either a patient with an EHR, or a practitioner with professional and personal medical documents, or any other member of the hospital staff with administrative, financial or health documents. To save resources, encryption can be avoided and so, medical information is stored in plain on the cloud server.

In addition, the user might need to add, delete and modify parts of the stored medical documents. If the user had to download all the medical documents from the cloud server in order to do this operation and then re-upload all of them, this process is cumbersome and inefficient.

In order to confirm that the cloud server has stored the data as expected, the owner should be able to check its integrity. An audit process can be employed to achieve this but requires significant computation and communications resources. In many cases a third party, rather than the hospital, will best be able to do this. Note that we do not wish this external party to learn any information about the medical information stored on the cloud server.

1.5 Contributions

Firstly, in order to securely and efficiently broadcast and share EHRs between the hospital and the medical staff members, there are two solutions. The first one is to send the encrypted medical document individually to each recipient, in this case the medical document is encrypted with the public key of each recipient which allows him/her to recover the medical document in plain by using his/her corresponding private key. Observe that this process is cumbersome and inefficient. With the second solution the encryptor chooses a public key and encrypts the medical document and then forwards to all recipients. The encryptor then gives the corresponding private key to users that s/he authorises. Note that sending a public and private key pair that is common to several users is not particularly secure. In addition, the encryptor has to generate a public and private key pair each time that a medical document has to be encrypted and shared, and this makes the process onerous. We propose new solutions by designing the following cryptographic primitives.

Broadcast Encryption with Membership. A medical institute acting as a group manager and a broadcaster, computes the public and private key pairs for all hospital staff members. Then, a hospital chooses a group G of hospital staff members and delivers a token $P(G)$ to the medical institute. The medical institute uses $P(G)$ to compute a ciphertext C that encrypts a medical document for group G, and forwards it to the hospital staff members. A hospital staff member can recover the medical document if and only if s/he belongs to G.

We observe the following features:

- the medical institute delivers the public and private key pairs;
- some (possibly all) hospital staff members belong to a group G chosen by the hospital;
- the token $P(G)$ is computed on input G, although $P(G)$ hides G from the medical institute's view;
- the medical institute is not aware of the chosen group G, but encrypts medical information according to it by using the token $P(G)$ provided by the hospital;
- only hospital staff members belonging to the group G and using the private key can correctly recover the data.

Certificate-Based Broadcast Encryption. A hospital acting as a broadcaster, encrypts some medical information according to a group G of hospital staff members that it has selected, and forwards it to all hospital staff members. A medical institute and other health legislators (such as the government for instance) acting as certifiers, deliver certificates (seen as licenses) to the hospital staff members. A hospital staff member can recover the medical information if and only if his/her certificate is valid at the time of the decryption process and s/he belongs to the group G determined by the hospital.

We observe the following features:

- the health legislators deliver the certificates, such as each legislator generates and returns a unique certificate to each medical staff member;

- the hospital encrypts the medical information according to a group G of hospital staff members, and forwards the resulting ciphertext to all hospital staff members;
- only hospital staff members that are authorised by the hospital (meaning that they belong to G) and that hold certificates valid at the current time, can recover the medical information.

Secondly, in order to securely and efficiently access and retrieve EHRs stored on cloud servers, we propose the following cryptographic primitive.

Certificate-Based Encryption with Keyword Search. A medical institute acting as a certifier, first generates certificates for all hospital staff members. It also computes update keys according to groups of hospital staff members, such that only a hospital staff member belonging to a specified group can successfully refresh his/her certificate. These certificates contain labels that can be seen as a description of the access rights given to the hospital staff members.

The hospital encrypts medical information and uploads the resulting ciphertexts to a cloud server. For each encrypted document, the hospital chooses a keyword that can be seen as a summary of the contents of the document, encrypts it and appends the resulting ciphertext to the encrypted document.

When a hospital staff member wants to retrieve a document, s/he sends a request to the cloud server in the form of a trapdoor that was generated with a keyword and his/her most refreshed certificate (that encrypts a valid label). The cloud server checks that the keyword and the label embedded into the trapdoor match the keyword appended to the encrypted document and the current label that it sustains.

We observe the following features:

- the medical institute delivers the certificates to all hospital staff members;
- each hospital staff member encrypts medical documents, and has to append an encrypted keyword component to each of them, the keyword acting as a description of the document contents;
- the cloud server on behalf of the hospital stores a current label regarding current law, administrative and financial situations;
- a hospital staff member has to provide a trapdoor containing a keyword and a valid certificate (embedding a label) in order to access and retrieve medical information.

Then, in order to securely and efficiently ensure patient privacy and save computational and communications resources in EHR-based storage, while these resources are limited, we propose the following cryptographic primitives.

On-line/Off-line Ciphertext-Policy Attribute-Based Proxy Re-Encryption. The medical institute acting as a certifier, generates the public and private key pairs for all hospital staff members. The hospital pre-encrypts a patient EHR with some credentials that results in an intermediate ciphertext *IntC*. Note that this step can be done off-line, and so can be executed at any time. Then, it uploads *IntC* to a cloud server acting as a proxy.

Later, this patient comes to the hospital for either an emergency or a check-up visit. Once a patient has decided which requirements the practitioner needs to satisfy, s/he sends a re-encryption key *rk* to the cloud server containing the credentials corresponding to these requirements. The cloud server re-encrypts *IntC* with these credentials and results in a final ciphertext *C*. Finally, a hospital staff member can recover the medical document if and only if his/her credentials match the ones embedded into *C*, and so s/he is authorised to meet the patient. This step is done on-line; however, it does not need many resources from the patient.

We observe the following features:

- the medical institute delivers the public and private key pairs to all hospital staff members;
- each hospital staff member can upload a patient EHR to the server, encrypted beforehand with some credentials such that encryption is available off-line;
- the patient can finalise the encryption on-line by specifying the credentials that s/he wants the recipient to satisfy;
- the cloud server is not fully trusted: it can re-encrypt the uploaded encrypted document, possibly with other credentials than the ones used for the first encryption. However, it should obtain neither information about the contents of the encrypted document nor information about the credentials.

Ciphertext-Policy DNA-Based Encryption. In order to design this primitive, we turn towards encryption (public) and decryption (private) keys using natural features that rely on both uniqueness and closeness. We offer a tool easy to find (since it does already exist), that is unique for each user but can be close enough to someone else's tool. More precisely, this tool is based on DNA sequences. Except for twins and other specific cases, each user has his/her own DNA sequence, that is a combination of his/her mother's DNA sequence and his/her father's DNA sequence. This means that the DNA sequence of a given user is unique but is closely related to his/her parents' DNA sequences. Even if this primitive applies for really specific situations, we include it here since it does satisfy the privacy-preserving and resource-saving requirements arisen above.

A patient, Alice, encrypts a medical document in a particular way, using her DNA sequence and uploads the resulting ciphertext *C* to a cloud server acting as a proxy. The server has to keep *C* on its local storage until a token is released (for instance, as soon as Alice becomes unable to proceed by herself). When this happens, the cloud server forwards the ciphertext *C* to the patient Charles, Alice's son. Charles uses his DNA sequence (close enough to Alice's one) to retrieve the medical document.

We observe the following features:

- the DNA sequence satisfies the properties of uniqueness and closeness: each user has his/her own DNA sequence (except for twins) while this is close enough to his/her relatives;
- the medical document cannot be retrieved if the DNA sequence of the decryptor is not close enough to the DNA sequence of the encryptor (a closeness metric has to be defined);

- the cloud server should not obtain any information on the data that it is storing and on the identities of the encryptor and the decryptor.

Finally, in order to securely and efficiently manage EHRs in cloud computing, we propose the following cryptographic primitive.

Dynamic Provable Data Possession with Public Verifiability and Data Privacy. The hospital staff members upload the medical information in plain to a cloud server. They can easily add, delete and modify contents without retrieving all the docuemts by requesting the cloud server to process the data operations. A third party auditor (TPA) is required to regularly verify that the cloud server does not misbehave and correctly stores the medical information. To make the audit possible, the hospital staff member who uploaded a document to the cloud server, generates a verification metadata and appends it to this document. Using both the document and the corresponding verification metadata, the cloud server can prove that it properly stores this document when it is challenged by the TPA on behalf of the medical staff member. We require that no information on the stored data is leaked to the TPA when the TPA is auditing the cloud server.

We observe the following features:

- the medical information is stored in plain on the cloud server, along with a verification metadata that is used to check that the cloud server correctly stores the medical information;
- the medical information does not need to be downloaded from the cloud server and then entirely re-uploaded when a staff medical member wants to add, delete or modify some contents;
- the cloud server is not fully trusted: a TPA audits it to verify that it correctly stores the medical information and has correctly done the data operations requested by the hospital staff members.
- the TPA is not fully trusted: we must avoid that the TPA obtains some information on the contents of the medical information stored on the cloud server.

Chapter 2

Preliminaries

2.1 Notations

$\mathbb{N}$ denotes the set of natural numbers $\{1,2,\cdots\}$ and $\mathbb{Z}$ denotes the set of integers $\{\cdots,-2,$ $-1,0,1,2,\cdots\}$. $\mathbb{Z}_p$ refers to the set $\{0,\cdots,p-1\}$ and $\mathbb{Z}_p^*$ is the set $\{k|1 \leq k \leq p$ and $\gcd(k,$ $p)=1\}$, i.e. the set of positive integers smaller than or equal to p and relatively prime to p. When p is a prime, we get that $\mathbb{Z}_p^* = \{1,\cdots,p-1\}$. $[1,n]$ denotes the set $\{1,2,\cdots,n\}$ of natural numbers smaller than or equal to $n \in \mathbb{N}$. $[1,n)$ denotes the set $\{1,2,\cdots,n\} \setminus \{n\} =$ $\{1,2,\cdots,n-1\}$ of natural numbers smaller than $n \in \mathbb{N}$. Given $a,b \in \mathbb{N}$, the notation $a \mid b$ denotes a divides b, meaning that there is an element $c \in \mathbb{N}$ such that $a \cdot c = b$. On the other side, the notation $a \nmid b$ says that a does not divide b.

The security parameter is denoted by $\lambda \in \mathbb{N}$. If $\mathbb{S}$ is a set (possibly called a *group*), then $|\mathbb{S}|$ is its cardinality. If $\mathbb{S}$ is a non-empty set, then we can write $a \in \mathbb{S}$ to mean that is an element belonging to $\mathbb{S}$. Generally, given a set $\mathbb{S}$ with identity element $1_{\mathbb{S}}$, we let $\mathbb{S}^*$ be the set $\mathbb{S} \setminus \{1_{\mathbb{S}}\}$. If $s_1, s_2 \in \{0,1\}^*$ are strings, then $s_1||s_2 \in \{0,1\}^*$ is the concatenation of binary strings s_1 and s_2.

A probabilistic polynomial time algorithm will be named in short as PPT algorithm. If $\mathbb{S}$ is a non-empty set, then $x \in_R \mathbb{S}$ denotes that x has been randomly and uniformly chosen in $\mathbb{S}$. Additionally, if Alg is an (PPT) algorithm, then $x \leftarrow \mathsf{Alg}(\_)$ means that Alg has been executed on some specified inputs $\_$ and its (random) output has been assigned to the variable x. In the security proofs, we denote by $\mathscr{A}$ a PPT adversary that wants to break the security of a given scheme, while we denote by $\mathscr{B}$ the challenger that interacts with $\mathscr{A}$ with goal to break the related security problem.

Let O be the notation to sort algorithms by how they respond to changes in input size when analysising them. A function $g(\cdot)$ written inside the notation $O(\cdot)$ is selected to be as simple as possible, omitting constant factors and lower order terms. Let f and g be two functions defined on some subset of the real numbers. We write $f(x) = O(g(x))$ as $x \to \infty$, if and only if there is a positive constant c such that for all sufficiently large values of x, the absolute value of $f(x)$ is at most c multiplied by the absolute value of $g(x)$. More precisely, $f(x) = O(g(x))$ if and only if there exists a positive real number c and a real number x_0 such that

$$|f(x)| \leq c|g(x)| \text{ for all } x \geq x_0.$$

In the protocol constructions, we let g be an element of the multiplicative cyclic group $\mathbb{G}_1$ of prime order p when considering symmetric pairings, while we let g_1, g_2 be elements of the multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p when considering asymmetric pairings. If $< g >= \mathbb{G}_1$, we say that g generates the group $\mathbb{G}_1$, such that$< g >$ is the cyclic subgroup of the powers of g and so, $\mathbb{G}_1$ is a cyclic group. Saying that an element g generates a group $\mathbb{G}_1$ is equivalent in saying that $< g >$ equals the entire group $\mathbb{G}_1$. For a finite group $\mathbb{G}_1$ of order p as we consider, it is also equivalent to say that g has order $|\mathbb{G}_1| = p$.

An exponential function is a function of the form $f(x) = a^x$, in which the input variable x occurs as an exponent. The *natural exponential function* $\exp(\cdot)$ maps x to the value $\exp(x)$. The exponential function $\exp(\cdot)$ can be defined by the following power series:

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

The logarithm is the inverse operation to exponentiation, such that the logarithm of a number is the exponent to which another fixed value, the base, must be raised to produce that number. More precisely, the logarithm of x to base a, denoted as $\log_a(x)$, is the unique real number y such that $a^y = x$. We simply denote by $\log(\cdot)$ the *natural logarithmic function* $\log_2(\cdot)$ of base 2. Therefore, for any real value x, we have the following property: $\exp(\log(x)) = \log(\exp(x)) = x$.

Let M be a matrix with l rows and n columns, such that $l, n \geq 1$. Let $m_{i,j}$ be the elements of the matrix M, for $i \in [1,l]$ and $j \in [1,n]$. We let M^T denote the transpose of the matrix M, such that M^T has n rows and l columns. Let $m^T_{j,i}$ be the elements of the matrix M^T, for $i \in [1,l]$ and $j \in [1,n]$. Therefore, for all $i, j \in [1,l] \times [1,n]$, we have $m^T_{j,i} = m_{i,j}$.

2.2 Mathematical Tools

In this section, we define the mathematical concepts that base our cryptosystems and their security. Mathematical notions include group theory and number theory.

2.2.1 Groups

A group is a set $\mathbb{G}$ along with an operation $\cdot$, that we call the group law of $\mathbb{G}$, that combines any two elements g and h to create another element that we denote $g \cdot h$. We sometimes write this element gh for short. The pair of set and operation $(\mathbb{G}, \cdot)$ must satisfy the follwoing group axioms to be a group:

1. *Closure:* For all elements $g, h \in \mathbb{G}$, the result of the operation $g \cdot h$ is also in the set $\mathbb{G}$.

2. *Associativity:* For all elements $g, h_1, h_2 \in \mathbb{G}$, $(g \cdot h_1) \cdot h_2 = g \cdot (h_1 \cdot h_2)$.

3. *Identity Element:* There exists $1_{\mathbb{G}} \in \mathbb{G}$ such that for every element $g \in \mathbb{G}$, the equation $1_{\mathbb{G}} \cdot g = g \cdot 1_{\mathbb{G}} = g$ holds. This elements $1_{\mathbb{G}}$ is unique.

4. *Inverse Element:* For every $g \in \mathbb{G}$, there exists an element $h \in \mathbb{G}$, denoted g^{-1}, such that $g \cdot h = h \cdot g = 1_{\mathbb{G}}$, where $1_{\mathbb{G}}$ is the identity element.

In some group $(\mathbb{G},\cdot)$, another property may be met: for all elements $g,h \in \mathbb{G}$, the equation $g \cdot h = h \cdot g$ holds. This property is called *commutativity*. Groups for which the commutativity equation $g \cdot h = h \cdot g$ always holds are called *abelian* groups. Observe that the equation $g \cdot h = h \cdot g$ may not be true since the result of an operation may depend on the order of the operands: the result of combining an element g with an element h may not yield the same result as combining the element h with the element g.

The set $\mathbb{G}$ is called the underlying set of the group $(\mathbb{G},\cdot)$. We use the underlying set $\mathbb{G}$ as a short name for the group $(\mathbb{G},\cdot)$.

Bijection

A *bijection* is a mapping that is both one-to-one (called an injection) and onto (called a surjection). This means that a bijection is a function which relates each element of a set $\mathbb{S}$ (called the domain) to a separate and distinct element of another set $\mathbb{S}'$ (called the range), where each element in $\mathbb{S}'$ also has a corresponding element in $\mathbb{S}$.

Homomorphisms and Variants

Let two groups be $(\mathbb{G}_1,\cdot)$ and $(\mathbb{G}_2,*)$. A *group homomorphism* from $(\mathbb{G}_1,\cdot)$ to $(\mathbb{G}_2,*)$ is a function $\eta : \mathbb{G}_1 \to \mathbb{G}_2$ such that, for all $g,h \in \mathbb{G}_1$, we get that $\eta(g \cdot h) = \eta(g) * \eta(h)$ where the group operation on the left hand side of the equation is that of $\mathbb{G}_1$ and on the right hand side that of $\mathbb{G}_2$.

From this property, one can deduce that η maps the identity element $1_{\mathbb{G}_1}$ of $\mathbb{G}_1$ to the identity element $1_{\mathbb{G}_2}$ of $\mathbb{G}_2$, and it also maps inverses to inverses as follows: $\eta(g^{-1}) = \eta(g)^{-1}$. Therefore, η is saied to be compatible with the group structure.

An *endomorphism* is a homomorphism from a mathematical object to itself. For instance, an endomorphism of a group $\mathbb{G}$ is a group homomorphism $\eta : \mathbb{G} \to \mathbb{G}$. An *isomorphism* is a homomorphism that admits an inverse. Two mathematical objects are isomorphic if an isomorphism exists between them. An *automorphism* is an isomorphism whose domain and range coincide. For instance, an isomorphism a group homomorphism $\eta : \mathbb{G}_1 \to \mathbb{G}_2$ such that η is a bijective function, an automorphism is a group $\mathbb{G}$ is a group homomorphism $\eta : \mathbb{G} \to \mathbb{G}$ such that η is a bijective function.

Consequences of the Group Axioms

Uniqueness of the Identity Element and Inverses. Two consequences of the group axioms are the uniqueness of the identity element $1_{\mathbb{G}}$ and the uniqueness of inverse elements. There is only one identity element in a group, and each element in a group has exactly one inverse element.

We now prove the uniqueness of an inverse element of g as follows. We suppose that g has two inverses h_1 and h_2 in a group $(\mathbb{G},\cdot)$. Using the group axioms 2, 3 and 4,

$$h_1 = h_1 \cdot 1_{\mathbb{G}} = h_1 \cdot (g \cdot h_2) = (h_1 \cdot g) \cdot h_2 = 1_{\mathbb{G}} \cdot h_2 = h_2.$$

Thus, there is only one inverse element of g. Similarly, we prove that the identity element of a group is unique by assuming that $(\mathbb{G},\cdot)$ is a group with two identity elements $1_{\mathbb{G}}^{(1)}$ and $1_{\mathbb{G}}^{(2)}$. Then, we get that $1_{\mathbb{G}}^{(1)} = 1_{\mathbb{G}}^{(1)} \cdot 1_{\mathbb{G}}^{(2)} = 1_{\mathbb{G}}^{(2)}$, and so $1_{\mathbb{G}}^{(1)}$ and $1_{\mathbb{G}}^{(2)}$ are equal.

Group Action. If $(\mathbb{G},\cdot)$ is a group and $\mathbb{S}$ is a set, then a (left) group action $\phi_{\mathbb{G},\mathbb{S}}$ of $\mathbb{G}$ on $\mathbb{S}$ is a function

$$\begin{array}{rcl} \phi_{\mathbb{G},\mathbb{S}} : \mathbb{G}\times\mathbb{S} & \to & \mathbb{S} \\ (g,x) & \to & \phi_{\mathbb{G},\mathbb{S}}(g,x) = g\cdot x \end{array}$$

that satisfies the two following axioms:

1. *Identity:* For all $x\in\mathbb{S}$, $1_{\mathbb{G}}\cdot x$.
2. *Compatibility:* For all $g,h\in\mathbb{G}$, for all $x\in\mathbb{S}$, $(g\cdot h)\cdot x = g\cdot(h\cdot x)$.

The group $(\mathbb{G},\cdot)$ is said to act on $\mathbb{S}$ on the left. The set $\mathbb{S}$ is called a left $\mathbb{G}$-set.

From the two axioms, we get that for every $g\in\mathbb{G}$, the function which maps $x\in\mathbb{S}$ to $g\cdot x$ is a bijective map from $\mathbb{S}$ to $\mathbb{S}$. The inverse of this function is the function which maps x to $g^{-1}\cdot x$. Hence, another definition of a group action of $\mathbb{G}$ on $\mathbb{S}$ is as a group homomorphism from $\mathbb{G}$ into the symmetric group $Sym(\mathbb{S})$ of all the bijections from $\mathbb{S}$ to $\mathbb{S}$.

In a similar way, we can define a right group action of $\mathbb{G}$ on $\mathbb{S}$ as an operation $\mathbb{S}\times\mathbb{G}\to\mathbb{S}$ mapping (x,g) to $x\cdot g$ and satisfying the two following axioms:

1. *Identity:* For all $x\in\mathbb{S}$, $x\cdot 1_{\mathbb{G}}$.
2. *Compatibility:* For all $g,h\in\mathbb{G}$, for all $x\in\mathbb{S}$, $x\cdot(g\cdot h) = (x\cdot g)\cdot h$.

The difference between left and right actions is in the order in which a product like $g\cdot h$ acts on x. For a left action, h acts first and is followed by g, while for a right action, g acts first and is followed by h. Because of the formula $(g\cdot h)^{-1} = h^{-1}\cdot g^{-1}$, we can construct a left action from a right action by composing with the inverse operation of the group. Moreover, a right action of $\mathbb{G}$ on $\mathbb{S}$ is the same thing as a left action of its opposite group $\mathbb{G}^{op}$ on $\mathbb{S}$. Thus, it is sufficient to only consider left actions without any loss of generality.

Division. In a group $(\mathbb{G},\cdot)$, the invertibility of the group action means that division is possible: given two elements $g,h\in\mathbb{G}$, there is exactly one solution $g_1\in\mathbb{G}$ to the equation $g_1\cdot g = h$. More precisely, multiplying the equation on its right side with the inverse g^{-1} of the element g gives $g_1 = g_1\cdot g\cdot g^{-1} = h\cdot g^{-1}$. In the same way, there is exactly one solution $g_2\in\mathbb{G}$ to the equation $g\cdot g_2 = h$, and so $g_2 = g^{-1}\cdot h$. If the operation $\cdot$ is commutative, then we get that $g_1 = g_2$. If the operation $\cdot$ is not commutative, then we may get that g_1 and g_2 are not equal.

Therefore, multiplying by a group element g is a bijection. Specifically, if $g\in\mathbb{G}$, then there is a bijection from $\mathbb{G}$ to itself called left translation by$g$ sending $h\in\mathbb{G}$ to $g\cdot h$. Similarly, right translation by g is a bijection from $\mathbb{G}$ to itself sending h to $h\cdot g$. If $\mathbb{G}$ is abelian, then left and right translations by a group element are the same.

2.2.2 Elliptic Curves

We start by considering elliptic curves over real numbers. In this case, an elliptic curve E is a plane curve given by an equation of the form $y^2 = x^3 + Ax + B$ for some real numbers A, B. Such equation is called a *Weierstrass equation*.

To ensure that the elliptic curve is non-singular, the discriminant $\Delta = 4\cdot A^3 + 27\cdot B^2$ should be non-zero. In other words, the polynomial $P(x) = x^3 + Ax + B$ has distinct roots. A non-singular elliptic curve means that the graph has neither cusps nor self-intersections nor isolated points.

Field

A *field* is a set $\mathbb{K}$ that is a commutative group with respect to two compatible operations, usually called addition and multiplication (where "compatible" is formalized by the property of distributivity) and denoted $+$ and $\cdot$ respectively, and such that the additive identity element 0 and the multiplicative identity element 1 have to be distinct.

- *Closure of* $\mathbb{K}$ *under addition and multiplication:* For all $A, B \in \mathbb{K}$, both $A+B$ and $A \cdot B$ are in $\mathbb{K}$ (meaning that $+$ and $\cdot$ are binary operations on $\mathbb{K}$).
- *Associativity of addition and multiplication:* For all $A, B, C \in \mathbb{K}$, we get the following: $A+(B+C) = (A+B)+C$ and $A \cdot (B \cdot C) = (A \cdot B) \cdot C$.
- *Commutativity of addition and multiplication:* For all $A, B \in \mathbb{K}$, we get the following: $A+B = B+A$ and $A \cdot B = B \cdot A$.
- *Existence of additive and multiplicative identity elements:* There exists an element of $\mathbb{K}$, called the additive identity element 0, such that for all A in $\mathbb{K}$, $A+0 = A$. In addition, there is an element, called the multiplicative identity element 1, such that for all A in $\mathbb{K}$, $A \cdot 1 = A$. Note that the additive identity element and the multiplicative identity element are required to be distinct.
- *Existence of additive inverses and multiplicative inverses:* For every $A \in \mathbb{K}$, there exists an element $-A$ in $\mathbb{K}$ such that $A+(-A) = 0$. Similarly, for any $A \in \mathbb{K}, A \neq 0$, there exists an element A^{-1} in $\mathbb{K}$ such that $A \cdot A^{-1} = 1$. The elements $A+(-B)$ and $A \cdot B^{-1}$ are also denoted $A - B$ and A/B, respectively, meaning that subtraction and division operations exist.
- *Distributivity of multiplication over addition:* For all $A, B, C \in \mathbb{K}$, we get the following: $A \cdot (B+C) = (A \cdot B) + (A \cdot C)$.

A field is thus an algebraic structure $(\mathbb{K}, +, \cdot, -, ^{-1}, 0, 1)$ consisting of two abelian groups that are $\mathbb{K}$ under $(+, -, 0)$ and $\mathbb{K} \setminus \{0\}$ under $(\cdot, ^{-1}, 1)$ such that $0 \neq 1$ and with $\cdot$ distributing over $+$.

Group Law

We first define a group structure on a smooth cubic curve in the projective plane. We consider two cases: the cubic curve is in Weierstrass normal form or is not.

First Case. The cubic curve *in Weierstrass normal form* has an additional point at infinity 1 which serves as the identity element of the group. Thus, the elliptic curve E is the set $E = \{(x,y) : y^2 = x^3 + Ax + B\} \cup \{1\}$.

Since the curve E is symmetrical about the x-axis, for any point g_1, we say that g_1^{-1} is the point opposite it. For the point at infinity, we assume that $1 = 1^{-1}$.

Let g_1 and g_2 be two points on the curve E. Thus, we can get a unique third point $g_1 \cdot g_2$ as follows. Firstly, we draw a live between the two points g_1 and g_2. By doing so, the cubic is generally intersected at a third point g_3. Then, we let $g_1 \cdot g_2$ to be g_3^{-1}, i.e. the point opposite g_3.

Now, let one of the two points be the point at infinity 1. Therefore, we define $g_1 \cdot 1 = g_1 = 1 \cdot g_1$ and so, 1 is the identity element of the group. We now consider that g_1 and g_2 are opposites of each other, and we define $g_1 \cdot g_2 = 1$. In addition, if $g_1 = g_2$, then we have only one point, meaning that we cannot define the line between g_1 and g_2. Instead, we need the tangent line to the curve at the point $g_1 = g_2$. Observe that in general, the tangent intersects a second point g_3, and as above, we can use its opposite g_3^{-1}. Nevertheless, when g_1 is an inflection point (that is, a point where the concavity of the curve changes), we have to take g_3 to be g_1 and so, $g_1 \cdot g_1$ is the point opposite it.

Second Case. The cubic curve *not in Weierstrass normal form* can still have a group structure defined by designating one of its nine inflection points as the identity element 1. In the projective plane, each line will intersect a cubic at three points when accounting for multiplicity. For instance, given a point g_1, the opposite g_1^{-1} is defined as the unique third point passing through the identity element 1 and g_1. Therefore, given any two points g_1 and g_2 of the curve E, the point $g_1 \cdot g_2$ is defined as g_3^{-1} such that g_3 is the unique third point on the line containing both g_1 and g_2.

Let $\mathbb{K}$ be a field and let E be the curve defined on $\mathbb{K}$. This means that the equation $y^2 = x^3 + Ax + B$ of the curve E has coefficients A and B in $\mathbb{K}$. Thus, the $\mathbb{K}$-rational points of E are the points on E whose coordinates are in $\mathbb{K}$, including the point at infinity 1. Let the set of the $\mathbb{K}$-rational points be denoted as $E(\mathbb{K})$. This set forms a group; indeed, one can show that if g_1 is in $E(\mathbb{K})$, then g_1^{-1} is also in $E(\mathbb{K})$, and if two points in $\{g_1, g_2, g_3\}$ are in $E(\mathbb{K})$, then the third one is in $E(\mathbb{K})$ too. Morevoer, if $\mathbb{K}$ is a subfield of $\mathbb{K}'$, then $E(\mathbb{K})$ is a subgroup of $E(\mathbb{K}')$.

The above group $E(\mathbb{K})$ can also be described as follows. Let the curve E be of equation $y^2 = x^3 + Ax + B$ over the field $\mathbb{K}$, such that the characteristic is assumed to be neither 2 nor 3, and let two points be g_1 with coordinates $(x_1, y_1$ and g_2 with coordinates $(x_2, y_2$ on the curve E. We first assume that $x_1 \neq x_2$. Let h be the slope of the line containing g_1 and g_2, then we get that $h = \frac{y_1 - y_2}{x_1 - x_2}$. Note that h is well defined because $\mathbb{K}$ is supposed to be a field. Let us define g_3 with coordinates (x_3, y_3) be equal to $(g_1 \cdot g_2)^{-1}$ as follows:

$$\begin{aligned} x_3 &= h^2 - x_1 - x_2 \\ y_3 &= y_1 + h(x_3 - x_1) \end{aligned}$$

On the other side, if $x_1 = x_2$, then

- if $y_1 = -y_2$ (that includes the case $y_1 = y_2 = 1$), then g_3 is defined as 1 and so, the inverse of each point on the curve E is found by reflecting it across the x-axis;

- if $y_1 = y_2$ and $y_1, y_2 \neq 1$, then g_3 with coordinates (x_3, y_3) is equal to $(g_1 \cdot g_1)^{-1} = (g_1^2)^{-1}$ as follows:

$$\begin{aligned} h &= \frac{3x_1^2 - A}{2y_1} \\ x_3 &= h^2 - 2x_1 \\ y_3 &= y_1 + h(x_3 - x_1) \end{aligned}$$

The Group $E(\mathbb{F}_q)$

A prime field $\mathbb{F}_q$, for a prime number q, is a finite field of order q and easily constructed as the integers modulo q. More precisely, we can represent the elements of $\mathbb{F}_q$ by the integers in the range $0, \cdots, q-1$.

We observe that the group $E(\mathbb{F}_q)$ is a finite group since points on E have coordinates in $\mathbb{F}_q$.

Theorems. We first recall three theorems before going forward. The first one was established by Poincaré: Let $\mathbb{K}$ be a field. We suppose that an elliptic curve E is given by an equation of the form $y^2 = x^3 + Ax + B$ with $A, B \in \mathbb{K}$. Let $E(\mathbb{K})$ denote the set of points of E with coordinates in $\mathbb{K}$, i.e. $E(\mathbb{K}) = \{(x,y) \in E : x,y \in \mathbb{K}\} \cup \{1\}$. Then $E(\mathbb{K})$ is a subgroup of the group of all points of E.

The second theorem is the following: Working over a finite field, the group of points $E(\mathbb{F}_q)$ is always either a cyclic group or the product of two cyclic groups.

A third theorem enunciated by Hasse precises: Let E be an elliptic curve given by an equation of the form $E : y^2 = x^3 + Ax + B$ with $A, B \in \mathbb{F}_q$. Then $|\#E(\mathbb{F}_q) - (q+1)| \leq 2\sqrt{q}$.

Consequences. The group $E(\mathbb{F}_q)$ has no more than $2q+1$ points. More precisely, for each $x \in \mathbb{F}_q$, the value of $f(x) = x^3 + Ax + B$ is a square in $\mathbb{F}_q^*$ with probability $1/2$. In addition, if $f(x) = y^2$ is a square, then two points (x,y) and $(x,-y)$ can be found in $E(\mathbb{F}_q)$. We recall that the point 1 at infinity belongs to $E(\mathbb{F}_q)$ too. Therefore, $E(\mathbb{F}_q)$ roughly contains $\#E(\mathbb{F}_q) \simeq \frac{1}{2} \cdot 2 \cdot q + 1 = q + 1$ points.

2.2.3 Weil Pairing

The Weil pairing is a mapping that is efficiently computable using Miller's algorithm [160], and thus suitable for designing cryptosystems. Such algorithm works for supersingular elliptic curves defined over a prime field $\mathbb{F}_q$ with $q > 3$. We note that the curve $y^2 = x^3 + 1$ over $\mathbb{F}_q$ with $q = 2 \mod 3$ is an example of these supersingular elliptic curves.

What is stated below can be generalized to other elliptic curves. Follow some properties about these curves [209]:

1. Let $q > 3$ and $n | q+1$. A supersingular curve $E/\mathbb{F}_q$ contains $q+1$ points in $\mathbb{F}_q$. Let 1 be the point at infinity. The group of points over $\mathbb{F}_q$ forms a cyclic group of order $q+1$. Let $g \in E(\mathbb{F}_q)$ be a point of order p.

2. The group of points $E(\mathbb{F}_{q^2})$ contains a point h of order p such that h is linearly independent of the points in $E(\mathbb{F}_q)$. Therefore, $E(\mathbb{F}_{q^2})$ contains a subgroup that is isomorphic to the group $\mathbb{Z}_p^2$. The group is generated by $g \in E(\mathbb{F}_q)$ and $h \in E(\mathbb{F}_{q^2})$. Let this group be denoted $E[p]$.

Then, $\mathbb{G}_T$ denotes the subgroup of $\mathbb{F}_{q^2}^*$ of order p. The Weil pairing e that we consider maps pairs of points in $E[p]$ to $\mathbb{G}_T$: we can write $e : E[p] \times E[p] \to \mathbb{G}_T$ [135]. Let g and h be two points in $E(\mathbb{F}_q^2)$.

Divisors. A divisor is a formal sum of points on the curve $E(\mathbb{F}_q^2)$. Let a divisor be defined as $D = \sum_g d_g(g)$ where $d_g \in \mathbb{Z}$ and $g \in E(\mathbb{F}_q^2)$. In particular, $D = 3(g_1) - 2(g_2) - (g_3)$ is a divisor. In our case, divisors $D = \sum_g d_g(g)$ such that $\sum_g d_g = 0$ are considered.

Functions. A function f on the curve $E(\mathbb{F}_q^2)$ can be seen as a rational function $f(x,y) \in \mathbb{F}_q^2(x,y)$. For any point $g = (x,y) \in E(\mathbb{F}_q^2)$, we write $f(g) = f(x,y)$.

Divisors of functions. Given a function f on the curve $E(\mathbb{F}_q^2)$, its divisor is defined as $(f) = \sum_g ord_g(f) \cdot (g)$. Note that $ord_g(f)$ denotes the order of the zero that f gets at the point g.

In particular, if $ax + by + c = 0$ is the line passing through the points $g_1, g_2 \in E(\mathbb{F}_q^2)$ with $g_1 \neq +/-g_2$, then this line intersects the curve at a third point $g_3 \in E(\mathbb{F}_q^2)$. In addition, the function $f(x,y) = ax + by + c$ has three zeroes g_1, g_2, g_3 and a pole of order 3 at infinity. $(f) = (g_1) + (g_2) + (g_3) - 3(1)$ is thus the divisor of the function f.

Principal divisors. We consider a divisor D. We assume that f is function such that its divisor satisfies $(f) = D$, then D is said to be a *principal divisor*. Moreover, $D = \sum_g d_g(g)$ is a principal divisor if and only if $\sum_g d_g = 0$ and $\sum_g d_g g = 1$. Note that the second condition requires the group action on the curve. If D is a principal divisor, then a unique function f (up to constant multiples) does exist such that $(D) = (f)$.

Equivalence of divisors. Two divisors D, D' are said to be equivalent if their difference $D - D'$ is a principal divisor. Moreover, any divisor $D = \sum_g d_g(g)$ such that $\sum_g d_g = 0$ is equivalent to a divisor of the form $D_0 = (h) - (1)$ for some $h \in E$. We finally note that $h = \sum_g d_g g$

Notations

Let f be a function and $D = \sum_g d_g(g)$ be a divisor. Let $f(D)$ be defined as $f(D) = \prod_g f(g)^{d_g}$. We know that $\sum_g d_g = 0$ which means that $f(D)$ remains unchanged: for instance, instead of f, one considers cf for any $c \in \mathbb{F}_{q^2}$.

We now show that the Weil pairing is well defined. Let two points $g, h \in E[n]$. Let D_g be a divisor which is equivalent to the divisor $(g) - (1)$. Since pD_g is a principal divisor, meaning that this divisor is equivalent to the principal divisor $p(g) - p(1)$, a function f_g does exist such that $(f_g) = pD_g$. In addition, we can define D_h and f_h in the same way.

The Weil pairing of g and h is defined as $\tilde{e}(g,h) = \frac{f_g(D_h)}{f_h(D_g)}$, assuming that no division by zero occurred. If this ratio is undefined (i.e. there is a zero at the denominator), one requires different divisors D_g and D_h in order to determine $\tilde{e}(g,h)$.

We have to show that the element of $\tilde{e}(g,h)$ is independent of the choice of the divisor D_g (respectively D_h) as long as D_g (respectively D_h) is equivalent to $(g) - (1)$ (respectively $(h) - (1)$) and D_g (respectively D_h) leads to a well defined value.

We specify D'_g as a divisor equivalent to D_g and we denote f'_g as a function such that $(f'_g) = pD'_g$. Therefore, we get that $D'_g = D_g + (f_0)$ and $f'_g = f_g \cdot f_0^p$ for some function f_0. Moreover, let us recall the Weil reciprocity as follows: for any two functions f, f_0, we have that $f((f_0)) = f_0((f))$. Then, we obtain that $\tilde{e}(g,h) = \frac{f'_g(D_h)}{f_h(D'_g)} = \frac{f_g(D_h) f_0(D_h)^p}{f_h(D_g) f_h((f_0))} = \frac{f_g(D_h)}{f_h(D_g)} \cdot \frac{f_0(pD_h)}{f_h((f_0))} = \frac{f_g(D_h)}{f_h(D_g)} \cdot \frac{f_0((f_h))}{f_h((f_0))} = \frac{f_g(D_h)}{f_h(D_g)}$. We can thus conclude that the Weil pairing is well defined.

Properties

The number q is prime such that $q = 2 \mod 3$ and the number $p > 3$ is a prime factor of $q + 1$. The elliptic curve E is defined by the equation $y^2 = x^3 + 1$ over $\mathbb{F}_q$. $E(\mathbb{F}_{q^r})$ specifies the group of points on E defined over $\mathbb{F}_{q^r}$ for some $r \in \mathbb{N}$. Follow some properties about E [209, 41]:

1. $x^3 + 1$ is a permutation on $\mathbb{F}_q$. Thus, the group $E(\mathbb{F}_q)$ contains $q+1$ points. We recall that the point at infinity is written as 1. In addition, $g \in E(\mathbb{F}_q)$ is a point of order q and then, $\mathbb{G}_1$ is the subgroup of points generated by g.

2. If $y_0 \in \mathbb{F}_q$, then (x_0, y_0) is a unique point on $E(\mathbb{F}_q)$, namely $x_0 = (y_0^2 - 1)^{1/3} \in \mathbb{F}_q$. Let (x,y) be a random non-zero point on $E(\mathbb{F}_q)$. Therefore, y is uniform in $\mathbb{F}_q$.

3. $\zeta \in \mathbb{F}_{q^2}$ such that $\zeta \neq 1$ is a solution of $x^3 - 1 = 0$ in $\mathbb{F}_{q^2}$. Moreover, the map $\varphi(x,y) = (\zeta x, y)$ is an automorphism of the group of points on the curve E. If g with coordinates $(x,y) \in E(\mathbb{F}_q)$, then $\varphi(g) \in E(\mathbb{F}_{q^2})$, but $\varphi(g) \notin E(\mathbb{F}_q)$. Thus, $g \in E(\mathbb{F}_q)$ is linearly independent of $\varphi(g) \notin E(\mathbb{F}_{q^2})$.

4. We know that the points $g \in \mathbb{G}_1$ and $\varphi(g)$ are linearly independent. Hence, they generate a group isomorphic to $\mathbb{Z}_p \times \mathbb{Z}_p$. This group of points is denoted as $E[p]$. Moreover, $\mathbb{G}_T$ refers to the subgroup of $\mathbb{F}^*_{q^2}$ of order p. Therefore, the Weil pairing on the curve $E(\mathbb{F}_{q^2})$ is the mapping $\tilde{e} : E[p] \times E[p] \to \mathbb{G}_T$ such that it satisfies $e(h,h') = 1$ given $h, h' \in E(\mathbb{F}_q)$. This means that the Weil pairing is degenerate on $E(\mathbb{F}_q)$, and so it is degenerate on the group $\mathbb{G}_1$. To get a non-degenerate map, we define the modified Weil pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ as $e(g,h) = \tilde{e}(g, \varphi(h))$.

As a summary, the Weil pairing has the following properties for points in $E[p]$:

- For all $g \in E[p]$, we have $\tilde{e}(g,g) = 1$.
- *Bilinearity:* $\tilde{e}(g \cdot g', h) = \tilde{e}(g,h) \cdot \tilde{e}(g',h)$ and $\tilde{e}(g, h \cdot h') = \tilde{e}(g,h) \cdot \tilde{e}(g,h')$.
- If $g, h \in E[p]$ are collinear then $\tilde{e}(g,h) = 1$. Similarly, $\tilde{e}(g,h) = \tilde{e}(h,g)^{-1}$.
- *p-th root:* For all $g,h \in E[p]$, we have $\tilde{e}(g,h)^p = 1$, i.e. $\tilde{e}(g,h) \in \mathbb{G}_T$.
- *Non-degeneracy in the following sense:* If $g \in E[p]$ satisfies $\tilde{e}(g,h) = 1$ for all $h \in E[n]$, then $g = 1$.

As suggested above, one can use the modified Weil pairing $e(g,h) = \tilde{e}(g, \varphi(h))$ to obtain a non-degenerate map, where φ is an automorphism on the group of points of E.

The Modified Weil Pairing

Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be defined as in the previous section. The modified Weil pairing satisfies the following properties (in addition of non-degeneracy):

1. *Bilinear:* For all $g,h \in \mathbb{G}_1$ and for all $a,b \in \mathbb{Z}$, we have $e(g^a, h^b) = e(g,h)^{ab}$.

2. *Non-degenerate:* If g is a generator of $\mathbb{G}_1$, then $e(g,g) \in \mathbb{F}_{q^2}$ is a generator of $\mathbb{G}_T$.

3. *Computable:* Given $g,h \in \mathbb{G}_1$, there is an efficient algorithm due to Miller [160] to compute $e(g,h) \in \mathbb{G}_T$. Its running time is comparable to exponentiation in $\mathbb{F}_q$.

2.2.4 Tate Pairing

There are a lot of curves, in general Abelian varieties, that can support the mapping$e :$ $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. In particular, let a prime number $q = 3 \mod 4$. Then, the curve defined by the equation $y^2 = x^3 + x$ over $\mathbb{F}_q$ and its endomorphism $\phi : (x,y) \rightarrow (-x, iy)$ where $i^2 = -1$. For instance, Galbraith [91] proposed to take supersingular elliptic curves over a field of small characteristic in order to reduce the component size in [41]. Other Abelian varieties are proposed by Rubin and Silverberg [198].

The Tate pairing [88] is another bilinear pairing with the required properties for cryptosystems. The modified Tate pairing [41] can be expressed as follows. Let two points $g, h \in E[p]$. Then, we define the pairing $T(g,h) = f_g(D_h)^{|\mathbb{F}^*_{q^2}|/p}$ where f_g and D_h are defined as in the section about the Weil pairing. Thus, we obtain the computable bilinear pairing $T : E[p] \times E[p] \rightarrow \mathbb{G}_T$.

2.3 From Mathematical Tools to Cryptography

Public-key cryptography is based on the intractability of mathematical problems that we express below. The first public-key cryptosystems were shown to be secure by relying on the difficulty of factoring a large integer as the multiplication of two or more large prime factors. Then, for cryptosystems based on elliptic curves, finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is assumed to be infeasible. Such assumption is called the Elliptic Curve Discrete Logarithm (ECDLP) problem. Thus, The security of elliptic curve-based systems depends on the ability to compute a point multiplication and the inability to compute the multiplicand given the original and product points. Moreover, the size of the elliptic curve determines the difficulty of the problem.

We first introduce elliptic curves in cryptography as a highlight of the concept. We then present group theory in cryptography, where we give more details on intractable mathematical problems required for public-key cryptography.

2.3.1 Elliptic Curves in Cryptography

In 1985, Miller [160] first suggested elliptic curves as a cryptographic tool. By the fact that elliptic curves have a group structure, these curves can replace groups used in Discrete Logarithm (DL)-based cryptosystems (for instance, Diffie-Hellman systems and ElGamal systems). In addition, since no algorithm does exist to compute the discrete logarithms on elliptic curves, then high security levels with (relatively) short keys can possibly be reached.

Later, Mezenes et al. [155] found that a type of elliptic curves, called supersingular, are weaker than general elliptic curves. Indeed, some properties specific to these curves enable an attacker to reduce the DL problem to a finite field such that more efficient algorithms can be found in order to compute the DL. Since supersingular elliptic curves had been chosen for cryptosystems, the community became concerned about this discovery.

However, some papers [88, 91, 223, 197, 62] showed that cryptosystems can be based on weak elliptic curves such that these curves contain additional properties that are used as properties of the underlying systems.

Elliptic Curve Discrete Logarithm (ECDL) Problem

The Elliptic Curve Discrete Logarithm (ECDL) problem is the Discrete Logarithm (DL) problem for the group of points on an elliptic curve E over a finite field $\mathbb{K}$. The best known algorithm to solve the ECDL problem for an elliptic curve E over $\mathbb{F}_q$ takes time $O(\sqrt{q})$ (exponential).

Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$, with equation $E : y^2 = x^3 + Ax + B$, where $A, B \in \mathbb{F}_q$. Let g and h be two points in $E(\mathbb{F}_q)$. The problem is to find an integer a so that $h = g^a$. We call the (smallest) integer a such that $h = g^a$ the DL of h with respect to g and we denote it as $a = \log_g(h)$. Let p be the order of g in the group $E(\mathbb{F}_q)$. Then, the map $\log_g$: (subgroup of E generated by g) $\rightarrow \mathbb{Z}/p\mathbb{Z}$ is a group isomorphism, and the inverse of a is g^a.

Bilinear Pairings

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of order p for some large prime p. Let a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We can determine three types of bilinear pairing [92], namely:

1. *Symmetric pairing* or *Type 1 pairing*, such that $\mathbb{G}_1 = \mathbb{G}_2$. Such pairing can be expressed on supersingular curves.

2. *Asymmetric pairing* or *Type 2 pairing*, such that $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ with $\psi(g_2) = g_1$ where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_1$. Such pairing can be expressed on ordinary curves.

3. *External asymmetric pairing* or *Type 3 pairing*, such that $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no known isomorphism ψ between $\mathbb{G}_2$ and $\mathbb{G}_1$. Such pairing can be expressed on ordinary curves.

No advantage was observed in using an asymmetric (type 2) pairing over an external asymmetric (type 3) pairing. Moreover, when converting a protocol from a symmetric (type 1) pairing to an external asymmetric (type 3) pairing, some tradeoffs may have to be made based on the performance.

2.3.2 Group Theory in Cryptography

The Diffie-Hellman key exchange (DHKE) was introduced by Diffie and Hellman [72] from an idea proposed by Merkle [156]. The protocol allows two parties to communicate by establishing a shared private key over an insecure public channel, without requiring any prior knowledge of each other. Before this work, two parties could initiate a secure encrypted communication by exchanging keys through a secure physical channel. There exists a variant of the above DHKE protocol based on elliptic curves, called the elliptic curve DiffieHellman (ECDH) [136].

ElGamal [75] presented the ElGamal encryption scheme which is a public-key encryption scheme based on the DHKE protocol. The security of this system relies on the difficulty of the Decisional Diffie-Hellman (DDH) problem. This problem is related to discrete logarithms in the cyclic group $\mathbb{G}_1$ of prime order p. However, this scheme does not satisfy the non-malleability property (the definition of this notion can be found in Section 3.3.4).

Cramer and Shoup [65] then gave a public-key encryption scheme, called the Cramer-Shoup system and proven secure against adaptive chosen ciphertext attack using standard cryptographic assumptions (the definition of this notion can be found in Section 3.3.5). The security is based on the computational intractability of the Decisional Diffie-Hellman (DDH) assumption. This scheme can be seen as an improvement of the ElGamal system since non-malleability is achieved.

Bilinear Pairings

Symmetric Bilinear Pairings. Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two cyclic groups of order p for some large prime p. A symmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ must satisfy the following properties:

1. *Bilinear:* e is bilinear if $e(g^a, h^b) = e(g,h)^{ab}$ for all $g,h \in \mathbb{G}_1$ and $a,b \in \mathbb{Z}_p$.

2. *Non-degenerate:* e is non-degenerate if e does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_T$. We know that $\mathbb{G}_1$ and $\mathbb{G}_T$ are groups of prime order, so that if g is a generator of $\mathbb{G}_1$, then $e(g,g)$ is a generator of $\mathbb{G}_T$.

3. *Computable:* e is computable if there is an efficient algorithm to compute $e(g,h)$ for any $g,h \in \mathbb{G}_1$.

A bilinear map e that fulfills the three aforementioned conditions is said to be an *admissible* bilinear map. We recall that $\mathbb{G}_1$ is a subgroup of the multiplicative group of points of an elliptic curve $E/\mathbb{F}_q$ and $\mathbb{G}_T$ is a subgroup of the multiplicative group of a finite field $\mathbb{F}^*_{q^2}$. The existence of this bilinear map e leads to the MOV reduction [155] and the easiness of the Decision Diffie-Hellman (DDH) problem [34], as explained below.

The name symmetric bilinear pairings come from the fact that the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is symmetric: $e(g,h) = e(h,g)$ for all $g,h \in \mathbb{G}_1$.

Asymmetric Bilinear Pairings. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of order p for some large prime p. A asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ must satisfy the following properties:

1. *Bilinear:* e is bilinear if $e(g_1^a, h_2^b) = e(g_1, h_2)^{ab}$ for all $g_1 \in \mathbb{G}_1$, $h_2 \in \mathbb{G}_2$ and $a,b \in \mathbb{Z}_p$.

2. *Non-degenerate:* e is non-degenerate if e does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_2$ to the identity in $\mathbb{G}_T$. We know that $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order, so that if g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$, then $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.

3. *Computable:* e is computable if there is an efficient algorithm to compute $e(g_1, h_2)$ for any $g_1 \in \mathbb{G}_1$ and $h_2 \in \mathbb{G}_1$.

A bilinear map e that fulfills the three aforementioned conditions is said to be an *admissible* bilinear map.

Basic Algorithmic Problems

We recall the definitions of well-known mathematical Diffie-Hellman problems. These problems are defined over the multiplicative cyclic group $\mathbb{G}_1$ of prime order p.

Discrete Logarithm (DL) Problem. We define the Discrete Logarithm (DL) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic group of λ-bit prime order p, where $\lambda \in \mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a \in \mathbb{G}_1$, the problem is to output a, for $a \in_R \mathbb{Z}_p$.

An algorithm $\mathscr{A}$ has advantage ε in solving the DL problem if

$$Pr[\mathscr{A}(g, g^a) = a] \geq \varepsilon.$$

The DL assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the DL problem in $\mathbb{G}_1$.

Computational Diffie-Hellman (CDH) Problem. We define the Computational Diffie-Hellman (CDH) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic group of λ-bit prime order p, where $\lambda \in \mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, g^b \in \mathbb{G}_1$, the problem is to output g^{ab}, for $a, b \in_R \mathbb{Z}_p$.

An algorithm $\mathscr{A}$ has advantage ε in solving the CDH problem if

$$Pr[\mathscr{A}(g, g^a, g^b) = g^{ab}] \geq \varepsilon.$$

The CDH assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the CDH problem in $\mathbb{G}_1$.

Decisional Diffie-Hellman (DDH) Problem. We define the Decisional Diffie-Hellman (DDH) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic group of λ-bit prime order p, where $\lambda \in \mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, g^b \in \mathbb{G}_1$, the problem is to decide either $Z = g^{ab}$ or $Z = g^z \in_R \mathbb{G}_1$, for $a, b, z \in_R \mathbb{Z}_p$.

An algorithm $\mathscr{A}$ has advantage ε in solving the DDH problem if

$$|Pr[\mathscr{A}(g, g^a, g^b) = g^{ab}] - Pr[\mathscr{A}(g, g^a, g^b) = g^z]| \geq \varepsilon.$$

The CDH assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the CDH problem in $\mathbb{G}_1$.

Consequences and Issues

The MOV Reduction. In 1993, Menezes et al. [155] showed that the DL problem in the group $\mathbb{G}_1$ is no harder than the DL problem in the group $\mathbb{G}_T$. To be more precise, we consider two elements $g, h \in \mathbb{G}_1$ as an instance of the DL problem in $\mathbb{G}_1$ such that g and h have order p. The goal is to find an element $a \in \mathbb{Z}_p$ such that $h = g^a$. From the properties satisfied by a symmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, we observe the following:

1. By bilinearity of e, we know that $e(h, g) = e(g, g)^a$.

2. By non-degeneracy of e, both $e(g, g)$ and $e(h, g)$ have order p in $\mathbb{G}_T$.

Therefore, we reduce the DL problem in $\mathbb{G}_1$ to a DL problem in $\mathbb{G}_T$. Note that we should select the security parameter λ in such a way that the DL is hard in $\mathbb{G}_T$, so we get that the DL hard in $\mathbb{G}_1$.

The Easiness of the Decision Diffie-Hellman (DDH) Problem. The Decision Diffie-Hellman (DDH) problem [34] in $\mathbb{G}_1$ is to decide either the output is equal to g^{ab} or g^c given g, g^a, g^b, where $a, b, c \in_R \mathbb{Z}_p$ and $g \in_R \mathbb{G}_1$. Joux and Nguyen [126] pointed out that the DDH problem is easy in $\mathbb{G}_1$. To see why, note that $c = ab \mod p \Leftrightarrow e(g, g^c) = e(g^a, g^b)$ given $g, g^a, g^b, g^c \in \mathbb{G}_1$, where $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a symmetric bilinear map.

The Computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$ is to output g^{ab} given g, g^a, g^b where $a, b \in_R \mathbb{Z}_p$ and $g \in_R \mathbb{G}_1$. This problem is still hard in $\mathbb{G}_1$. Joux and Nguyen [126] provided symmetric bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ where the CDH problem is believed to be hard in $\mathbb{G}_1$ even though the DDH problem is easy in $\mathbb{G}_1$.

Bilinear Diffie-Hellman (BDH) Problem

In this section, we give details about the Bilinear Diffie-Hellman (BDH) problem in $(\mathbb{G}_1, \mathbb{G}_T)$ [41]. Since the DDH problem is easy in $\mathbb{G}_1$, the DDH assumption cannot be used to build cryptosystems in the group $\mathbb{G}_1$ and prove them secure. Instead, the security of cryptosystems can be based on a variant of the CDH assumption, called the Bilinear Diffie-Hellman (BDH) assumption.

The Bilinear Diffie-Hellman (BDH) problem is as follows. As above, we consider the two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of λ-bit prime order p as well as the admissible symmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ and the generator $g \in \mathbb{G}_1$. Given (g, g^a, g^b, g^c), the BDH problem is to output $e(g,g)^{abc} \in \mathbb{G}_T$ for some $a, b, c \in_R \mathbb{Z}_p$.

An algorithm $\mathscr{A}$ has advantage ε in solving the BDH problem in $(\mathbb{G}_1, \mathbb{G}_T)$ if

$$Pr[\mathscr{A}(g, g^a, g^b, g^c) = e(g,g)^{abc}] \geq \varepsilon,$$

where the probability is over the random choice of $a, b, c \in \mathbb{Z}_p$, the random choice of $g \in \mathbb{G}_1$, and the random bits of $\mathscr{A}$. The BDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$ if no PPT algorithm $\mathscr{A}$ has advantage non-negligible ε in solving the BDH problem in $(\mathbb{G}_1, \mathbb{G}_T)$.

BDH Parameter Generator $\mathscr{G}$. Let a randomized algorithm $\mathscr{G}$ be a BDH parameter generator with the following properties:

1. $\mathscr{G}$ takes a security parameter $\lambda \in \mathbb{N}$.
2. $\mathscr{G}$ runs in polynomial time in λ.
3. $\mathscr{G}$ outputs a prime number p, the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_T$ and the description of an admissible symmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Let $\mathscr{G}(\lambda) = (p, \mathbb{G}_1, \mathbb{G}_T, e)$ denote $\mathscr{G}$'s output.

We recall that the security parameter λ is used to determine the size of the λ-bit prime p. For instance, p could be a random λ-bit prime. Moreover, the description of the group $\mathbb{G}_1$ (respectively of the group $\mathbb{G}_T$) is supposed to contain polynomial time (in λ) algorithms to generate the group action in $\mathbb{G}_1$ (respectively in $\mathbb{G}_T$) and to contain a generator g of $\mathbb{G}_1$ (respectively a generator $e(g,g)$ of $\mathbb{G}_T$). The generator g of $\mathbb{G}_1$ (respectively the generator $e(g,g)$ of $\mathbb{G}_T$) allows to output uniformly random elements in $\mathbb{G}_1$ (respectively in $\mathbb{G}_T$). Similarly, the description of e is assumed to embed a polynomial time algorithm for computing e.

The security parameter is $\lambda \in \mathbb{N}$ such that $\lambda > 2$. The BDH parameter generator $\mathcal{G}$ selects a random λ-bit prime p and outputs the smallest prime q such that $q = 2 \mod 3$, $p \mid q+1$ and $p^2 \nmid q+1$. Let $q = lp+1$ for an integer l. $\mathbb{G}_1$ is the subgroup of order p of the group of points on the supersingular elliptic curve E defined by the equation $y^2 = x^3 + 1$ over $\mathbb{F}_q$, and $\mathbb{G}_T$ is the subgroup of order p of $\mathbb{F}^*_{q^2}$. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be the modified Weil pairing as defined previously. The BDH parameter generator $\mathcal{G}$ is believed to satisfy the BDH assumption asymptotically. Nevertheless, values q and p have to be chosen carefully to make the BDH problem sufficiently hard in practice. First, one should start to get the DL problem sufficiently hard in $\mathbb{G}_1$. As mentioned above, the DL problem in $\mathbb{G}_1$ is efficiently reducible to DL in $\mathbb{G}_T$ [155, 88]. Therefore, the DL computation in $\mathbb{F}^*_{q^2}$ is sufficient for the DL computation in $\mathbb{G}_1$. In the reality, primes q that are at least 512-bits long are required to ensure the security of the DL assumption in $\mathbb{F}^*_{q^2}$ since the group size will be at least 1024-bits long. In other words, the aforementioned BDH parameter generator $\mathcal{G}$ can be used with primes p that are longer than 512-bits.

Definition of the BDH Problem. $\mathcal{G}$ is a BDH parameter generator as defined above. An algorithm $\mathcal{A}$ is said to have advantage ε in solving the BDH problem for $\mathcal{G}$ if for any sufficiently large λ, we get:

$$Pr[\mathcal{A}(p,\mathbb{G}_1,\mathbb{G}_T,e,g,g^a,g^b,g^c) = e(g,g)^{abc} | (p,\mathbb{G}_1,\mathbb{G}_T,e) \leftarrow \mathcal{G}(\lambda), g \in \mathbb{G}_1, a,b,c \in \mathbb{Z}_p] \geq \varepsilon.$$

$\mathcal{G}$ is said to satisfy the BDH assumption if for any randomized PPT algorithm $\mathcal{A}$, we have that the advantage of $\mathcal{A}$ is a negligible function. We say that the BDH problem is hard in groups generated by $\mathcal{G}$ when $\mathcal{G}$ satisfies the BDH assumption.

Hardness of the BDH Problem. The BDH problem in $(\mathbb{G}_1,\mathbb{G}_T)$ is currently said to be no harder than the CDH problem in $\mathbb{G}_1$ or $\mathbb{G}_T$. This means that an algorithm for the CDH problem in $\mathbb{G}_1$ or $\mathbb{G}_T$ is sufficient to solve the BDH problem in $(\mathbb{G}_1,\mathbb{G}_T)$. However, the converse is not true and still an open problem: is an algorithm for the BDH problem sufficient to solve the CDH problem in $\mathbb{G}_1$ or in $\mathbb{G}_T$?

The isomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_T$ induced by the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ are supposed to be one-way functions. In other words, for $h \in \mathbb{G}_1$, let the isomorphism $f_h : \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be defined as $f_h(g) = e(g,h)$.

Suppose that there is one of these isomorphisms that is invertible. Thus, this gives us that the BDH problem is easy in $(\mathbb{G}_1,\mathbb{G}_T)$. In addition, if an efficient algorithm that inverts f_h for some value h exists, then an efficient algorithm that solves the DDH problem in $\mathbb{G}_T$ does exist. In all our protocols, the DDH problem is assumed to be hard in the group $\mathbb{G}_T$, and so all the isomorphisms $f_h : \mathbb{G}_1 \rightarrow \mathbb{G}_T$ induced by the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ are assumed to be one-way functions.

Extension to Asymmetric Bilinear Pairings Nothing changes with what was written above, except that the BDH assumption in $(\mathbb{G}_1,\mathbb{G}_T)$ has to be modified to get the co-BDH aassumption in $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T)$. The co-Bilinear Diffie-Hellman (co-BDH) problem is as follows. As above, we consider the three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of λ-bit prime order p as well as the admissible asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and the generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Given $(g_1, g_1^a, g_1^b, g_1^c, g_2, g_2^a, g_2^b, g_2^c)$, the co-BDH problem is to output $e(g_1,g_2)^{abc} \in \mathbb{G}_T$ for some $a,b,c \in_R \mathbb{Z}_p$.

An algorithm $\mathscr{A}$ has advantage ε in solving the co-BDH problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ if

$$Pr[\mathscr{A}(g_1, g_1^a, g_1^b, g_1^c, g_2, g_2^a, g_2^b, g_2^c) = e(g_1, g_2)^{abc}] \geq \varepsilon,$$

where the probability is over the random choice of $a, b, c \in \mathbb{Z}_p$, the random choices of $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, and the random bits of $\mathscr{A}$. The BDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ if no PPT algorithm $\mathscr{A}$ has advantage non-negligible ε in solving the BDH problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

We assume that the co-BDH assumtpion is accepted, and instead of supersingular curves, we can use elliptic curves over $\mathbb{F}_q$ for $q > 3$ proposed by Miyaji et al. [163]. These non-supersingular curves $E/\mathbb{F}_q$ have the property that if $p||E(\mathbb{F}_q)|$ then $E[p] \subseteq E(\mathbb{F}_{q^6})$. Remember that $E[p]$ is the group containing all point in E of order dividing p.

These curves can be used as follows. Let $\mathbb{G}_2$ to be a cyclic subgroup of $E(\mathbb{F}_q)$ of order p and $\mathbb{G}_1$ to be a different cyclic subgroup of $E(\mathbb{F}_{q^6})$ of the same order p. We obsserve that the bilinear map e can be the Weil or Tate pairings on $\mathbb{G}_1 \times \mathbb{G}_2$ as defined previously. Another option is to let $\mathbb{G}_1$ be a subgroup of order p of $E(\mathbb{F}_q)$ and $\mathbb{G}_2$ be a different subgroup of $E(\mathbb{F}_{q^6})$ of the same order.

Diffie-Hellman Problems

We recall the definition of mathematical Diffie-Hellman problems derived from the DDH and CDH problems, that we require to prove the security of the schemes presented in this thesis. These problems are defined over the multiplicative cyclic group $\mathbb{G}_1$ of prime order p.

s-Diffie-Hellman Inversion (DHI) Problem. Mitsunari et al. [161] presented this problem by calling it the s-weak Diffie-Hellman (wDH) problem in $\mathbb{G}_1$. Boneh and Boyen [37] showed that the s-Diffie-Hellman Inversion (DHI) assumption implies the s-Generalized Diffie-Hellman (GenDH) assumption [213, 169, 31]. In other words, cryptosystems relying on the s-GenDH assumption can instead rely on the s-DHI assumption since this appears to be a more natural complexity assumption.

We define the s-Diffie-Hellman Inversion (DHI) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic group of λ-bit prime order p, where $\lambda \in \mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, \cdots, g^{a^s}$, the problem is to output $g^{1/a}$, for $a \in_R \mathbb{Z}_p^*$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-DHI problem if

$$Pr[\mathscr{A}(g, g^a, \cdots, g^{a^s}) = g^{1/a}] \geq \varepsilon.$$

The s-DHI assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-DHI problem in $\mathbb{G}_1$.

s-Strong Diffie-Hellman (SDH) Problem. Boneh and Boyen [35] introduced the s-Strong Diffie-Hellman (SDH) problem in $\mathbb{G}_1$. To prove the s-SDH assumption, the authors provided a lower bound on the computational complexity of the s-SDH problem for generic groups following Shoup's work [208].

We define the s-Strong Diffie-Hellman (SDH) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic group of λ-bit prime order p, where $\lambda \in \mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, \cdots, g^{a^s}$, the problem is to output $(b, g^{\frac{1}{a+b}})$, for $a, b \in_R \mathbb{Z}_p$, $a+b \neq 0 \mod p$, and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-SDH problem if

$$Pr[\mathscr{A}(g, g^a, \cdots, g^{a^s}) = (b, g^{\frac{1}{a+b}})] \geq \varepsilon.$$

The s-SDH assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-SDH problem in $\mathbb{G}_1$.

Strong Decisional Diffie-Hellman (SDDH) Problem. Pfitzmann and Sadeghi [179] first gave the Strong Decisional Diffie-Hellman (SDDH) problem.

We define the Strong Decisional Diffie-Hellman (SDDH) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic groups of λ-bit prime order p, where $\lambda \in \mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, g^b, g^{1/b}$, the problem is to decide either $Z = g^{ab}$ or $Z = g^z \in_R \mathbb{G}_1$, for $a, z \in_R \mathbb{Z}_p$ and $b \in_R \mathbb{Z}_p^*$.

An algorithm $\mathscr{A}$ has advantage ε in solving the SSDH problem if

$$|Pr[\mathscr{A}(g, g^a, g^b, g^{1/b}) = g^{ab}] - Pr[\mathscr{A}(g, g^a, g^b, g^{1/b}) = g^z]| \geq \varepsilon.$$

The SSDH assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the SSDH problem in $\mathbb{G}_1$.

s-Diffie-Hellman Exponent (DHE) Problem. The s-Diffie-Hellman Exponent (DHE) probem in $\mathbb{G}_1$ was proposed in [239, 131, 49].

We define the s-Diffie-Hellman Exponent (DHE) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic group of λ-bit prime order p, where $\lambda \in \mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}$, the problem is to output $g^{a^{s+1}}$, for $a \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-DHE problem if

$$Pr[\mathscr{A}(g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}) = g^{a^{s+1}}] \geq \varepsilon.$$

The s-DHE assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-DHE problem in $\mathbb{G}_1$.

Truncated Version. To prove the security of one of the schemes presented in this thesis, we need a truncated form of the s-DHE problem. More precisely, the s-DHE assumption still holds in $\mathbb{G}_1$ by giving $g, g^a, \cdots, g^{a^s}$ (the values $g^{a^{s+2}}, \cdots, g^{a^{2s}}$ are missing).

s-Decisional Diffie-Hellman Exponent (DDHE) Problem. We define the s-Decisional Bilinear Diffie-Hellman Exponent (DDHE) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic group of λ-bit prime order p, where $\lambda \in \mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}$, the problem is to decide either $Z = g^{a^{s+1}}$ or $Z = g^z \in_R \mathbb{G}_1$, for $a, z \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-DDHE problem if

$$|Pr[\mathscr{A}(g,g^a,\cdots,g^{a^s},g^{a^{s+2}},\cdots,g^{a^{2s}})=g^{a^{s+1}}]-Pr[\mathscr{A}(g,g^a,\cdots,g^{a^s},g^{a^{s+2}},\cdots,g^{a^{2s}})=g^z]|\geq\varepsilon.$$

The s-DDHE assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-DDHE problem in $\mathbb{G}_1$.

Variant Version. To prove the security of one of the schemes presented in this thesis, we need a variant form of the s-DDHE problem. More precisely, the s-DDHE assumption still holds in $\mathbb{G}_1$ by giving $g_1,g_1^a,\cdots,g_1^{a^s}\in\mathbb{G}_1$ and $g_2,g_2^a\in\mathbb{G}_2$ (the values $g_1^{a^{s+2}},\cdots,g_1^{a^{2s}}\in\mathbb{G}_1$ are missing and the values $g_2,g_2^a\in\mathbb{G}_2$ are given).

(f,s)-Diffie-Hellman Exponent (DHE) Problem. Guo et al. [110] first proposed the (f,s)-Diffie-Hellman Exponent (DHE) problem in $\mathbb{G}_1$. The (f,s)-Diffie-Hellman Exponent (DHE) problem is a modified version of the s-DHE problem [49].

We define the (f,s)-Diffie-Hellman Exponent (DHE) problem as follows. Let $\mathbb{G}_1$ be a multiplicative cyclic group of λ-bit prime order p, where $\lambda\in\mathbb{N}$ is the security parameter. Let g be a generator of $\mathbb{G}_1$. Given $g,g^a,\cdots,g^{a^s}$, the problem is to output $(f(x),g^{f(a)})$ for $a\in_R\mathbb{Z}_p$, $s\in_R\mathbb{N}$ and $f(x)\in\mathbb{Z}_p[x]$ is a s'-degree polynomial function for $s'>s$.

An algorithm $\mathscr{A}$ has advantage ε in solving the (f,s)-DHE problem if

$$Pr[\mathscr{A}(g,g^a,\cdots,g^{a^s})=(f(x),g^{f(a)})]\geq\varepsilon.$$

The (f,s)-DHE assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the (f,s)-DHE problem in $\mathbb{G}_1$.

Symmetric External Diffie-Hellman (SXDH) Problem. The Symmetric External Diffie-Hellman (SXDH) problem is an extension of the DDH problem in $(\mathbb{G}_1,\mathbb{G}_2)$ such that $\mathbb{G}_1\neq\mathbb{G}_2$. The SXDH assumption holds if and only the DDH assumption holds in both in $\mathbb{G}_1$ and $\mathbb{G}_2$. The problem was first suggested in [202, 39] and formally enunciated in [22].

We define the Symmetric External Diffie-Hellman (SXDH) problem as follows. Let $\mathbb{G}_1,\mathbb{G}_2$ be two multiplicative cyclic groups of λ-bit prime order p, where $\lambda\in\mathbb{N}$ is the security parameter. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. The problem can be split into two subproblems.

Given g_1,g_1^a,g_1^b, the first subproblem is to decide either $Z_1=g_1^{ab}$ or $Z_1=g_1^{z_1}\in_R\mathbb{G}_1$, for $a,b,z_1\in_R\mathbb{Z}_p$.

Given g_2,g_2^a,g_2^b, the second subproblem is to decide either $Z_2=g_2^{ab}$ or $Z_2=g_2^{z_2}\in_R\mathbb{G}_2$, for $a,b,z_2\in_R\mathbb{Z}_p$.

An algorithm $\mathscr{A}$ has advantage ε in solving the SXDH problem if

$$|Pr[\mathscr{A}(g_1,g_1^a,g_1^b)=g_1^{ab}]-Pr[\mathscr{A}(g_1,g_1^a,g_1^b)=g_1^{z_1}]|\geq\varepsilon$$

and

$$|Pr[\mathscr{A}(g_2,g_2^a,g_2^b)=g_2^{ab}]-Pr[\mathscr{A}(g_2,g_2^a,g_2^b)=g_2^{z_2}]|\geq\varepsilon.$$

The SXDH assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the SXDH problem in $\mathbb{G}_1$ and in $\mathbb{G}_2$.

Bilinear Diffie-Hellman Problems

We give the mathematical bilinear Diffie-Hellman problems derived from the BDH problem, that we require to prove the security of the schemes presented in this thesis. These problems are defined over three multiplicative cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p (with possibly $\mathbb{G}_1 = \mathbb{G}_2$) and with the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Note that we first express the problem definitions based on symmetric bilinear pairings, and we sometimes extend them to the asymmetric bilinear pairing context.

Decisional Bilinear Diffie-Hellman (DBDH) Problem. The hardness of the Decisional Bilinear Diffie-Hellman (DBDH) assumption enables to prove the security of the identity-based encryption scheme in [41].

We define the Decisional Bilinear Diffie-Hellman (DBDH) problem as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathscr{G}(\lambda) \to (p, \mathbb{G}_1, \mathbb{G}_T, e)$ that outputs the description of symmetric bilinear groups of λ-bit prime order p as well as an admissible symmetric bilinear map e. Let g be a generator of $\mathbb{G}_1$. Given g, g^a, g^b, g^c, the problem is to decide either $Z = e(g,g)^{abc}$ or $Z = e(g,g)^z \in_R \mathbb{G}_T$, for $a,b,c,z \in_R \mathbb{Z}_p$.

An algorithm $\mathscr{A}$ has advantage ε in solving the DBDH problem if

$$|Pr[\mathscr{A}(g, g^a, g^b, g^c) = e(g,g)^{abc}] - Pr[\mathscr{A}(g, g^a, g^b, g^c) = e(g,g)^z]| \geq \varepsilon.$$

The DBDH assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the DBDH problem in $(\mathbb{G}_1, \mathbb{G}_T)$.

Asymmetric Bilinear Pairings. We now give the definition of the DBDH problem in the asymmetric pairing context. The Decisional Bilinear Diffie-Hellman (DBDH) problem is as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathscr{G}(\lambda) \to (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ that outputs the description of asymmetric bilinear groups of λ-bit prime order p as well as an admissible asymmetric bilinear map e. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. Given g_1, g_1^a, g_1^b, g_1^c and g_2, g_2^a, g_2^b, g_2^c, the problem is to decide either $Z = e(g_1,g_2)^{abc}$ or $Z = e(g_1,g_2)^z \in_R \mathbb{G}_T$, for $a,b,c,z \in_R \mathbb{Z}_p$.

An algorithm $\mathscr{A}$ has advantage ε in solving the DBDH problem if

$$\begin{aligned}&|Pr[\mathscr{A}(g_1, g_1^a, g_1^b, g_1^c, g_2, g_2^a, g_2^b, g_2^c) = e(g_1,g_2)^{abc}]\\ &-Pr[\mathscr{A}(g_1, g_1^a, g_1^b, g_1^c, g_2, g_2^a, g_2^b, g_2^c) = e(g_1,g_2)^z]| \geq \varepsilon.\end{aligned}$$

The DBDH assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the DBDH problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

Truncated Version. To prove the security of one of the schemes presented in this thesis, we need a truncated form of the DBDH problem. More precisely, the DBDH assumption still holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ by giving $g_1, g_1^a, g_1^b, g_2, g_2^b, g_2^c$ (the values g_1^c and g_2^a are missing).

s-Bilinear Diffie-Hellman Inversion (BDHI) Problem. Boneh and Boyen [37] introduced the s-Bilinear Diffie-Hellman Inversion (BDHI) problem over the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$.

We define the s-Bilinear Diffie-Hellman Inversion (BDHI) problem as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathcal{G}(\lambda) \rightarrow (p, \mathbb{G}_1, \mathbb{G}_T, e)$ that outputs the description of symmetric bilinear groups of λ-bit prime order p as well as an admissible symmetric bilinear map e. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, \cdots, g^s$, the problem is to output $e(g,g)^{1/a}$, for $a \in_R \mathbb{Z}_p^*$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-BDHI problem if

$$Pr[\mathscr{A}(g, g^a, \cdots, g^s) = e(g,g)^{1/a}] \geq \varepsilon.$$

The s-BDHI assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-BDHI problem in $(\mathbb{G}_1, \mathbb{G}_T)$.

s-Bilinear Diffie-Hellman Exponent (BDHE) Problem. Boneh et al. [38] first defined the s-Bilinear Diffie-Hellman Exponent (BDHE) problem over the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$.

We define the s-Bilinear Diffie-Hellman Exponent (BDHE) problem as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathcal{G}(\lambda) \rightarrow (p, \mathbb{G}_1, \mathbb{G}_T, e)$ that outputs the description of symmetric bilinear groups of λ-bit prime order p as well as an admissible symmetric bilinear map e. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, g^b$, the problem is to output $e(g,g)^{a^{s+1} \cdot b}$, for $a, b \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-BDHE problem if

$$Pr[\mathscr{A}(g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, g^b) = e(g,g)^{a^{s+1} \cdot b}] \geq \varepsilon.$$

The s-BDHE assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_T)$.

Asymmetric Bilinear Pairings. We now give the definition of the s-BDHE problem in the asymmetric pairing context. The $s+1$-Bilinear Diffie-Hellman Exponent (BDHE) problem is as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathcal{G}(\lambda) \rightarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ that outputs the description of asymmetric bilinear groups of λ-bit prime order p as well as an admissible asymmetric bilinear map e. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. Given $g_1, g_1^a, \cdots, g_1^{a^s}, g_1^{a^{s+2}}, \cdots, g_1^{a^{2s}}, g_1^b$ and $g_2, g_2^a, \cdots, g_2^{a^s}, g_2^{a^{s+2}}, \cdots, g_2^{a^{2s}}, g_2^b$, the problem is to output $Z = e(g_1, g_2)^{a^{s+1} \cdot b}$, for $a, b \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-BDHE problem if

$$Pr[\mathscr{A}(g_1, g_1^a, \cdots, g_1^{a^s}, g_1^{a^{s+2}}, \cdots, g_1^{a^{2s}}, g_1^b, g_2, g_2^a, \cdots, g_2^{a^s}, g_2^{a^{s+2}}, \cdots, g_2^{a^{2s}}, g_2^b) = e(g_1, g_2)^{a^{s+1} \cdot b}] \geq \varepsilon.$$

The s-BDHE assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-BDHE problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

s-Decisional Bilinear Diffie-Hellman Exponent (DBDHE) Problem. We define the s-Decisional Bilinear Diffie-Hellman Exponent (DBDHE) problem as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathcal{G}(\lambda) \rightarrow (p, \mathbb{G}_1, \mathbb{G}_T, e)$ that outputs the description of symmetric bilinear groups of λ-bit prime order p as well as an admissible symmetric bilinear map e. Let g be a generator of $\mathbb{G}_1$. Given $g, g^a, \cdots, g^{a^s}$, $g^{a^{s+2}}, \cdots, g^{a^{2s}}, g^b$, the problem is to decide either $Z = e(g,g)^{a^{s+1}\cdot b}$ or $Z = e(g,g)^z \in_R \mathbb{G}_T$, for $a, b, z \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-DBDHE problem if

$$\begin{aligned}&|Pr[\mathscr{A}(g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, g^b) = e(g,g)^{a^{s+1}\cdot b}]\\&-Pr[\mathscr{A}(g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, g^b) = e(g,g)^z]| \geq \varepsilon.\end{aligned}$$

The s-DBDHE assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-DBDHE problem in $(\mathbb{G}_1, \mathbb{G}_T)$.

Asymmetric Bilinear Pairings. We now give the definition of the s-DBDHE problem in the asymmetric pairing context. The s-Decisional Bilinear Diffie-Hellman Exponent (DBDHE) problem is as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathscr{G}(\lambda) \to (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ that outputs the description of asymmetric bilinear groups of λ-bit prime order p as well as an admissible asymmetric bilinear map e. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. Given $g_1, g_1^a, \cdots, g_1^{a^s}, g_1^{a^{s+2}}, \cdots, g_1^{a^{2s}}, g_1^b$ and $g_2, g_2^a, \cdots, g_2^{a^s}, g_2^{a^{s+2}}, \cdots, g_2^{a^{2s}}, g_2^b$, the problem is to decide either $Z = e(g_1,g_2)^{a^{s+1}\cdot b}$ or $Z = e(g_1,g_2)^z \in_R \mathbb{G}_T$, for $a, b, z \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-DBDHE problem if

$$\begin{aligned}&|Pr[\mathscr{A}(g_1, g_1^a, \cdots, g_1^{a^s}, g_1^{a^{s+2}}, \cdots, g_1^{a^{2s}}, g_1^b, g_2, g_2^a, \cdots, g_2^{a^s}, g_2^{a^{s+2}}, \cdots, g_2^{a^{2s}}, g_2^b) = e(g_1,g_2)^{a^{s+1}\cdot b}]\\&-Pr[\mathscr{A}(g_1, g_1^a, \cdots, g_1^{a^s}, g_1^{a^{s+2}}, \cdots, g_1^{a^{2s}}, g_1^b, g_2, g_2^a, \cdots, g_2^{a^s}, g_2^{a^{s+2}}, \cdots, g_2^{a^{2s}}, g_2^b) = e(g_1,g_2)^z]| \geq \varepsilon.\end{aligned}$$

The s-DBDHE assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-DBDHE problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

Truncated Version. To prove the security of one of the schemes presented in this thesis, we need a truncated form of the s-DBDHE problem. More precisely, the s-DBDHE assumption still holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ by giving $g_1, g_1^a, \cdots, g_1^{a^s}, g_1^{a^{s+2}}, \cdots, g_1^{a^{2s}}, g_1^b \in \mathbb{G}_1$ and $g_2, g_2^a, \cdots, g_2^{a^s}, g_2^b \in \mathbb{G}_2$ (the values $g_2^{a^{s+2}}, \cdots, g_2^{a^{2s}} \in \mathbb{G}_2$ are missing).

s-Decisional Parallel Bilinear Diffie-Hellman Exponent (DPBDHE) Problem Waters [237] first gave the definition of the s-Decisional Parallel Bilinear Diffie-Hellman Exponent (DPBDHE) problem and proved the hardness of the assumption in the generic group model. Note that this assumption can be viewed as a generalization of the BDHE problem.

We define the s-Decisional Parallel Bilinear Diffie-Hellman Exponent (DPBDHE) problem as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathscr{G}(\lambda) \to (p, \mathbb{G}_1, \mathbb{G}_T, e)$ that outputs the description of symmetric bilinear groups of λ-bit prime order p as well as an admissible symmetric bilinear map e. Let g be a generator of $\mathbb{G}_1$. Given g and the tuple Y containing the following elements

$$\begin{aligned}&g^c, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}\\&\forall j \in [1,s], \quad g^{c\cdot b_j}, g^{a/b_j}, \cdots, g^{a^s/b_j}, g^{a^{s+2}/b_j}, \cdots, g^{a^{2s}/b_j}\\&\forall j,k \in [1,s], k \neq j, \quad g^{a\cdot c\cdot b_k/b_j}, \cdots, g^{a^s\cdot s\cdot b_k/b_j}\end{aligned}$$

the problem is to decide either $Z = e(g,g)^{a^{s+1}\cdot c}$ or $Z \in_R \mathbb{G}_T$, for $a, b_1, \cdots, b_s, c \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-DPBDHE problem if

$$Pr[\mathscr{A}(g,Y) = e(g,g)^{a^{s+1}\cdot c}] \geq \varepsilon.$$

The s-DPBDHE assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-DPBDHE problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

s-type Problem Rouselakis and Waters [195] first studied the hardness of the s-type assumption. This s-type assumption in $\mathbb{G}_1$ is similar to the DPBDHE assumption [237].

We define the s-type problem as follows. Given a security parameter $\lambda \in \mathbb{N}$, there is a group generator algorithm $\mathscr{G}(\lambda) \to (p, \mathbb{G}_1, \mathbb{G}_T, e)$ that outputs the description of symmetric bilinear groups of λ-bit prime order p as well as an admissible symmetric bilinear map e. Let g be a generator of $\mathbb{G}_1$. Given g and the tuple Y containing the following elements

$$\begin{array}{l}
g^c \\
\forall i,j \in [1,s], \quad g^{a^i}, g^{b_j}, g^{c\cdot b_j}, g^{a^i\cdot b_j}, g^{a^i/b_j^2} \\
\forall i \in [1,2s], j \in [1,s], i \neq s+1, \quad g^{a^i/b_j} \\
\forall i \in [1,2s], j,k \in [1,s], j \neq k, \quad g^{a^i b_j/b_k^2} \\
\forall i \in [1,2s], j,k \in [1,s], j \neq k, \quad g^{c\cdot a^i b_j/b_k}, g^{c\cdot a^i b_j/b_k^2}
\end{array}$$

the problem is to decide either $Z = e(g,g)^{a^{s+1}\cdot c}$ or $Z \in_R \mathbb{G}_T$, for $a, b_1, \cdots, b_s, c \in_R \mathbb{Z}_p$ and $s \in_R \mathbb{N}$.

An algorithm $\mathscr{A}$ has advantage ε in solving the s-type problem if

$$Pr[\mathscr{A}(g,Y) = e(g,g)^{a^{s+1}\cdot c}] \geq \varepsilon.$$

The s-type assumption holds if no PPT algorithm $\mathscr{A}$ has advantage non-negligible in solving the s-type problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

Relations between the Algorithmic Problems of Group Theory

Given two problems P_1 and P_2, we denote by $P_1 \to P_2$ if the problem P_2 can be solved in polynomial time with polynomially many queries to the oracle solving the problem P_1. We recall the acronyms of the above algorithmic problems:

- DL: Discrete Logarithm;
- CDH: Computational Diffie-Hellman;
- DDH: Decisional Diffie-Hellman;
- SXDH: Symmetric External Diffie-Hellman;
- DHI: Diffie-Hellman Inversion;
- GenDH: Generalized Diffie-Hellman;

- SDH: Strong Diffie-Hellman;
- BDHI: Bilinear Diffie-Hellman Inversion;
- BDHE: Bilinear Diffie-Hellman Exponent;
- DPBDHE: Decisional Parallel Bilinear Diffie-Hellman Exponent.

In Figure 2.1, we illustrate the relations among the algorithmic problems [38, 59].

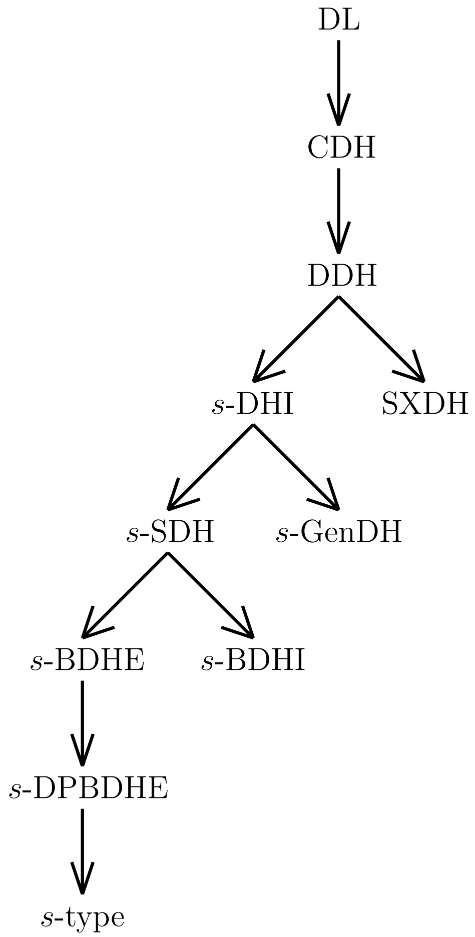


Figure 2.1: Relations between the various problems defined in the previous section.

Chapter 3

Cryptographic Primitives and Security Notions

3.1 Public-Key Cryptography

In a public-key encryption setting, cryptosystems comprise two keys: a public key that is known to everyone and a private key that is only known to the recipient of the message. For instance, let Alice and Bob be respectively a sender and a recipient. Alice encrypts the message that she wants to forward to Bob using Bob's public key. Then, Bob uses his private key to decrypt it. Follow some properties:

1. Public and private keys are linked. More precisely, for a public key used to encrypt a message, only the corresponding private key can be used to decrypt it.

2. Given a public key, it should be impossible in practice to deduct the corresponding private key.

Public-key encryption offers various advantages. Unlike symmetric encryption (where the private key is used for both encryption and decryption), the distribution of the public keys used for encryption is easy and the publication of such keys is available. In addition, digital signatures can be designed in a public-key setting. Digital signatures enable a user that receives a message to verify that this message is really sent from another particular user, whereas this user can detect if the message was modified in transit. Finally, signing a message by using a digital signature is an approval of this message that the user sending it cannot deny.

Nevertheless, public-key encryption also present several disadvantages. A first difficulty encountered in public-key cryptosystems is that we have to know the public key of a recipient to encrypt a message for him/her. Thus, public keys have to be registered and made available to all the users. Moreover, these keys have to be authenticated before use to ensure that they belong to the correct users. Then, public-key encryption is slower and requires more computer resources than symmetric encryption. Finally, if an attacker is able to successfully guess the private key of a user, then the messages received by this user can all be read, while if this user loses his/her private key, then s/he cannot decrypt the received ciphertexts.

3.1.1 Public-Key Encryption and Public-Key Encryption with Keyword Search

Public-Key Encryption

A public-key encryption (PKE) scheme is composed of the following four algorithms [66]:

- $\mathsf{Setup}(\lambda) \rightarrow params$. On input a security parameter λ, output the public parameters $params$.
- $\mathsf{KeyGen}(params) \rightarrow (ek, dk)$. On input the public parameters $params$, output an encryption (public) key ek and a decryption (private) key dk.
- $\mathsf{Encrypt}(params, ek, m) \rightarrow C$. On input the public parameters $params$, the encryption key ek and a message m, output a ciphertext C.
- $\mathsf{Decrypt}(params, dk, C) \rightarrow m/\perp$. On input the public parameters $params$, the decryption key dk and a ciphertext C, output the message m for a valid ciphertext; output $\perp$ otherwise.

Correctness. We require that a public-key encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $params \leftarrow \mathsf{Setup}(\lambda)$, $(ek, dk) \leftarrow \mathsf{Keygen}(params)$ and $C \leftarrow \mathsf{Encrypt}(params, ek, m)$, we have that $\mathsf{Decrypt}(params, dk, C) = m$.

Public-Key Encryption with Keyword Search

Public-key encryption with keyword search (PEKS) enables a server to search for a keyword from a collection of encrypted information contents, given a trapdoor provided by a receiver.

PEKS was designed as a solution to the problem of searching encrypted information using a public-key setting. In more details, the data is private, stored in their encrypted form on databases and organised in a random way that is not controlled by the receiver.

Existing Work. Boneh et al. [40] introduced the notion of public-key encryption with keyword search (PEKS) in order to solve the problem of searching on encrypted information in a public-key environment. They provided a scheme that is secure against adaptive chosen-keyword attacks in the random oracle model. Then, Baek et al. [20] combined the PEKS scheme presented in [40] and a variation of the ElGamal encryption scheme [75]. Byun et al. [47] pointed out vulnerabilities present in existing PEKS schemes due to the fact that keywords might not have high min-entropy and receivers might choose well-known keywords to search information (off-line keyword-guessing attacks).

Thereafter, Bellare et al. [25] presented database encryption techniques that achieve fast search (in logarithmic time) while provably reaching strong privacy. Fuhr and Paillier [89] introduced the concept of searchable encryption which enables decryption, since PEKS schemes do not allow the receiver to decrypt keywords. In parallel, Gu et al. [106] proposed a new PEKS scheme based on bilinear pairings in the random oracle model. Their scheme is computationally consistent and there is no pairing operation involved in the encryption procedure.

Subsequently, Baek et al. [21] improved the concept of PEKS by noticing several issues that were not considered in [40]. They defined the notion of secure-channel-free PEKS (SCF-PEKS) and gave a construction in the random oracle model. The same year, Abdalla et al. [1] studied the consistency in PEKS schemes. Consistency refers to the extent to which false positives are produced. Fang et al. [81] constructed the first SCF-PEKS scheme that does not require random oracle and that is proven secure according to the security model defined in [20]. The same year, Rhee et al. [187] defined the concept of PEKS with a designated tester (dPEKS) that is similar to SCF-PEKS. They improved the security model given in [20] in order to enhance the adversary's ability.

Later, Rhee et al. [188] introduced the notion of trapdoor indistinguishability that suffices to oppose keyword-guessing attacks. They constructed a dPEKS scheme proven secure against keyword-guessing attacks. Rhee et al. [186] gave generic transformations to construct a dPEKS scheme using IBE schemes. Yau et al. [242] stressed a keyword guessing attack by an outsider on existing dPEKS schemes [187, 188]. They showed that their attack is generic and applies to all existing dPEKS schemes that claim to be secure against keyword-guessing attacks by an outsider. Then, Fang et al. [82] pointed out the security vulnerabilities found in the existing schemes [20, 81, 187, 188] and gave a SCF-PEKS scheme that is secure against chosen-keyword attacks, chosen-ciphertext attacks and keyword-guessing attacks without random oracle.

Recently, Boldyreva and Chenette [33] defined the notion of efficiently fuzzy searchable encryption (EFSE) in order to solve the problem of fuzzily searching on encrypted information in symmetric key systems. Informally, a receiver may mispell keywords or provide noisy information in the query and a server should be still able to locate the information containing the requested keywords.

Definition. A public-key encryption with keyword search (PEKS) scheme is composed of the following five algorithms [40]:

- $\mathsf{Setup}(\lambda) \to params$. On input a security parameter λ, output the public parameters $params$.
- $\mathsf{KeyGen}(params) \to (pk, sk)$. On input the public parameters $params$, output an poublic key ek and a private key sk.
- $\mathsf{Encrypt}(params, pk, w) \to C$. On inputs the public parameters $params$, the public key pk and a keyword w, output a ciphertext C.
- $\mathsf{TrapGen}(params, sk, w') \to C$. On inputs the public parameters $params$, the private key sk and a keyword w', output a trapdoor T.
- $\mathsf{Test}(params, C, T) \to true/false\}$. On inputs the public parameters $params$, a ciphertext C for a keyword w and a trapdoor T for a keyword w', output $true$ if $w = w'$; output $false$ otherwise.

Correctness. We require that a public-key encryption with keyword search scheme is correct if for any $\lambda \in \mathbb{N}$, $params \leftarrow \mathsf{Setup}(\lambda)$, $(pk, sk) \leftarrow \mathsf{Keygen}(params)$, $C \leftarrow \mathsf{Encrypt}(params, pk, w)$ and $T \leftarrow \mathsf{TrapGen}(params, sk, w')$, we have that $\mathsf{Decrypt}(params, dk, C) = true$.

3.1.2 Identity-Based Encryption, Ciphertext-Policy Attribute-Based Encryption and Derivatives

Identity-Based Encryption

Identity-based encryption (IBE) allows a party to encrypt a message using the recipient's identity as a public key. The ability to use identities as public keys avoids the need to distribute public-key certificates. This can be very useful in applications such as email where the reicever is often off-line and unable to present a public-key certificate while the sender encrypts a message.

The main advantage of IBE is that the public keys of the recipients do not need to be obtained since the recipients' identities are used for their generation and so for the encryption. Observe that the recipient does not even need to be in the possession of the private key related to his/her identity when the ciphertext is produced.

Observe that in a public-key setting, the users generate their public and private keys locally, publish the public key and keep the private key secret. Conversely, in the identity-based setting, the public and private keys are generated by a trusted authority, that is an inherent key escrow, i.e. this authority may decrypt the user's ciphertexts and/or may issue signatures on behalf of these users.

We recall that the identity used for the public key is some information that all the users originally know, for instance an email address. Nevertheless, if the corresponding private key is compromised, then a solution is to choose an new identity (e.g. a new email address), but this does not seem practical. Therefore, an identity should contain additional information, for instance a key expiration date. However, the related public key is no longer well known by all the users and meet public-key setting disadvantages (i.e. distribution and publication of the public keys).

Existing Work. Shamir [206] introduced the idea of identity-based encryption (IBE) and Boneh and Franklin [41] presented the first secure and efficient scheme based on bilinear maps. Meanwhile, Cocks [63] constructed a identity-based encryption system based on quadratic residues. Canetti et. al. [51] defined a weaker model of security for identity-based encryption, called selective-ID model, in which the adversary must first declare which identity it wishes to be challenged on before the parameters are generated. The authors gave a system provably secure in this selective-ID model without random oracles. Boneh and Boyen [37] provided a more efficient scheme that is secure in the selective-ID model, in the standard model. Finally, Boneh and Boyen [36] proposed a scheme that is fully secure without random oracles. However, their construction is too inefficient to be of practical use. Then, Waters [235] described an efficient identity-based encryption system that is fully secure in the standard model.

The notion of hierarchical identity-based encryption (HIBE) was introduced by Horwitz and Lynn [117]. In a HIBE system, users' identities are arranged in a hierarchy and a party can use its own private key to issue a private key to any of its descendants. Thus, HIBE is an IBE extension in which key delegation is included. The first HIBE construction was given by Gentry and Silverberg [96]; the scheme was proved secure in the random oracle model. Selectively secure systems in the standard model were then proposed in [51, 37, 38]. Later, Gentry [94] presented a security proof for an IBE system outside of the partitioning paradigm; however at the cost of employing a q-based assumption. Then, Gentry and Halevi [95] extended the techniques to provide the first fully secure HIBE system to allow a hierarchy of polynomial depth.

Definition. A identity-based encryption (IBE) scheme is composed of the following four algorithms [206]:

- $\mathsf{Setup}(\lambda) \to (params, msk)$. On input a security parameter λ, output the public parameters *params* and the master secret key *msk*.
- $\mathsf{KeyGen}(params, msk, id) \to sk_{id}$. On inputs the public parameters *params*, the master secret key *msk* and an identity *id*, output a private key sk_{id}, such that sk_{id} is associated with the identity *id*.
- $\mathsf{Encrypt}(params, id, m) \to C_{\mathbb{AS}}$. On inputs the public parameters *params*, an identity *id* and a message *m*, output a ciphertext C.
- $\mathsf{Decrypt}(params, id, sk_{id}, C) \to m/\perp$. On inputs the public parameters *params*, an identity *id*, the corresponding private key sk_{id} and a ciphertext C, output m for a valid ciphertext; output $\perp$ otherwise.

Correctness. We require that a identity-based encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda)$, $sk_{id} \leftarrow \mathsf{KeyGen}(params, msk, id)$ and $C \leftarrow \mathsf{Encrypt}(params, id, m)$, we have that $\mathsf{Decrypt}(params, id, sk_{id}, C) = m$.

Ciphertext-Policy Attribute-Based Encryption

Attribute-based encryption (ABE) allows parties to encrypt and decrypt messages based on recipient attributes. This can be seen as a generalisation of IBE and can effectively increase the flexibility of data sharing such that only parties satisfying specific policy are allowed to access the data.

The advantage to using ABE over IBE is that a document can be stored on an untrusted storage server, instead of relying on a trusted one, in order to make authentication verifications to deliver this document.

Two main disadvantages are met in ABE. Unfortunately, ABE is non-efficient and does not offer a mechanism for attribute revocation. More precisely, the computational costs for encryption and key generation are due to the complexity of the access policy or the number of attributes. Then, revocation is more challenging in an attribute-based setting, since that each attribute may belong to several users, while in a public-key setting, a public and private key pair is uniquely associated to one user. In ABE, note that the attributes should be revoked, unlike the users or the keys. In addition, problems from dealing with the keys are encountered, such as key coordination, key escrow and key revocation.

Existing Work. Sahai and Waters [201] introduced the idea of attribute-based encryption (ABE). The notion of key-policy ABE (KP-ABE) emerged in [104]. In a KP-ABE system, the ciphertexts are associated to an attribute set and each of the private keys is related to an access policy over the attributes. Bethencourt et al. [30] proposed the first ciphertext-policy ABE (CP-ABE) in which the ciphertexts are related with an access policy and each of the private keys corresponds to a set of attributes. CP-ABE and KP-ABE selectively secure constructions followed in [181, 30, 60, 176, 103, 237]. Except for [176] (in which negation is possible), the ABE schemes only work for non monotonic access structures.

Thereafter, Waters [237] proposed the first fully secure CP-ABE systems in the standard model. Other fully secure constructions in the standard model were given by Lewko et al. [140], using the dual pairing vector space framework. However, these schemes are less efficient that the ones in [237]. Meanwhile, Lewko and Waters [142] proposed the first large universe KP-ABE construction in the standard model, using composite order groups. Based on the techniques initiated by Okamoto and Takashima [175], Lewko [139] formalized the first large universe KP-ABE scheme in prime order groups. Recently, Rouselakis and Waters [195] proposed two large universe ABE constructions in the prime order group setting. The schemes are proven selectively secure in the standard model under two q-based assumptions.

Subsequently, Lewko and Waters [143] suggested a CP-ABE scheme that is proven fully secure while achieving similar efficiency of selectively secure ABE systems. Attrapadung et al. [16] presented the first KP-ABE schemes allowing for non-monotonic access structures with constant ciphertext size. Moreover, Attrapadung et al. [15] offered the first ABE scheme allowing for truly expressive access structures and with constant ciphertext size. The authors constructed a CP-ABE scheme with constant-size ciphertexts for threshold access policies. They also provided a KP-ABE construction supporting non-monotonic access structures with short ciphertexts.

The concept of an ABE setting with multiple central authorities was addressed by Chase [53]. However, the scheme relied on a central authority and was limited to expressing a strict “AND” policy over a predetermined set of authorities. Chase and Chow [54] achieved to remove the central authority using a distributed pseudorandom function; however, the same limitations of an “AND” policy of a determined set of authorities remained.

Definition. A ciphertext-policy attribute-based encryption (CP-ABE) scheme is composed of the following four algorithms [201]:

- $\mathsf{Setup}(\lambda, \mathcal{U}) \rightarrow (params, msk)$. On inputs a security parameter λ and an attribute universe $\mathcal{U}$, output the public parameters *params* and the master secret key *msk*.
- $\mathsf{KeyGen}(params, msk, S) \rightarrow sk_S$. On inputs the public parameters *params*, the master secret key *msk* and an attribute set S, output a private key sk_S, such that sk_S is associated with the attribute set S.
- $\mathsf{Encrypt}(params, \mathbb{AS}, S, m) \rightarrow C_{\mathbb{AS}}$. On inputs the public parameters *params*, an access structure $\mathbb{AS}$ for attributes over $\mathcal{U}$, a set of attributes S satisfying the access structure $\mathbb{AS}$ and a message m, output a ciphertext $C_{\mathbb{AS}}$.

  We assume that the access structure $\mathbb{AS}$ is included in the ciphertext $C_{\mathbb{AS}}$.
- $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) \rightarrow m/\perp$. On inputs the public parameters *params*, an attribute set S, the corresponding private key sk_S and a ciphertext $C_{\mathbb{AS}}$, output m if S satisfies $\mathbb{AS}$; output $\perp$ otherwise, indicating either $C_{\mathbb{AS}}$ is invalid or S does not satisfy $\mathbb{AS}$.

Correctness. We require that a ciphertext-policy attribute-based encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda, \mathcal{U})$, $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$ and $C_{\mathbb{AS}} \leftarrow \mathsf{Encrypt}(params, \mathbb{AS}, S, m)$, and if S satisfies $\mathbb{AS}$, we have that $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) = m$.

On-line/Off-line Ciphertext-Policy Attribute-Based Encryption

On-line/Off-line ABE (OO-ABE) is a natural extension of ABE that splits the computation for the encryption and the user key generation into two phases:

- a preparation phase that does most of the work: encrypting a message or creating a private key before the message or the pair (attribute list,access control policy) that will be used is known.
- a finalisation phase can then rapidly assemble an ABE ciphertext or key when the elements become known.

OO-ABE implies that not only the message is unknown during the preparation phase (as for classic on-line/off-line encryption), but also the attribute lists and the access policies.

We recall that one drawback from ABE is that encryption and key generation are too inefficient for some applications. On-line/off-line attribute-based setting appears to be a solution to overcome the above issue, by dividing the computation of encryption and key generation into the preparation phase and the finalisation phase.

An application of OO-ABE is for mobile device technology. Indeed, imagine that the preparation work is performed such that the phone is connected to a power source (thus, the device has the necessary computational resources), then the finalisation phase with the ABE operations is performed such that the phone is not longer plugged to the source while the battery of this phone is not significantly consumed.

Existing Work. Even et al. [78] introduced the notion of on-line/off-line cryptography in the context of signatures. Later, Shamir and Tauman [207] generalized the notion by using chameleon hash functions [134]. Guo et al. [109] applied the concept for IBE and obtained an off-line encryption system for IBE. Recently, Hohenberger and Waters [115] developped new techniques for OO-ABE encryption and gave solutions in both key-policy context and ciphertext-policy context.

Definition. A on-line/off-line ciphertext-policy attribute-based encryption (OO-CP-ABE) scheme is composed of the following five algorithms [115]:

- $\mathsf{Setup}(\lambda, \mathscr{U}) \to (params, msk)$. On inputs a security parameter λ and an attribute universe $\mathscr{U}$, output the public parameters *params* and the master secret key *msk*.
- $\mathsf{KeyGen}(params, msk, S) \to sk_S$. On inputs the public parameters *params*, the master secret key *msk* and an attribute set S, output a private key sk_S, such that sk_S is associated with the attribute set S.
- $\mathsf{OffEncrypt}(params, m) \to IntC$. On inputs the public parameters *params* and a message m, output an intermediate ciphertext $IntC$.
- $\mathsf{OnEncrypt}(params, \mathbb{AS}, S, IntC) \to C_{\mathbb{AS}}$. On inputs the public parameters *params*, an access structure $\mathbb{AS}$ for attributes over $\mathscr{U}$, a set of attributes S satisfying the access structure $\mathbb{AS}$ and an intermediate ciphertext $IntC$, output a ciphertext $C_{\mathbb{AS}}$.

  We assume that the access structure $\mathbb{AS}$ is included in the ciphertext $C_{\mathbb{AS}}$.

- $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) \to m/\perp$. On inputs the public parameters *params*, an attribute set S, the corresponding private key sk_S and a ciphertext $C_{\mathbb{AS}}$, output m if S satisfies $\mathbb{AS}$; output $\perp$ otherwise, indicating either $C_{\mathbb{AS}}$ is invalid or S does not satisfy $\mathbb{AS}$.

Correctness. We require that a on-line/off-line ciphertext-policy attribute-based encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda, \mathscr{U})$, $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$, $IntC \leftarrow \mathsf{OffEncrypt}(params, m)$ and $C_{\mathbb{AS}} \leftarrow \mathsf{OnEncrypt}(params, \mathbb{AS}, S, IntC)$, and if S satisfies $\mathbb{AS}$, we have that $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) = m$.

3.1.3 Broadcast Encryption and Membership Encryption

Broadcast Encryption

Broadcast encryption (BE) allows a party to encrypt a message to a set of recipients such that only the recipients within this set can decrypt it. Observe that the set of recipients is chosen by the party at the time of encryption. In BE cryptosystems, any subset of recipients can be included in a broadcast, however successful decryption of encrypted messages is only possible for recipients included in the broadcast using their own private keys.

In context with large number of recipients, BE is particularly efficient. Moreover, such primitive has applications in secure database system, digital right management (DRM) and group communications.

In a broadcast-encryption setting, a master secret key is set at the beginning of the protocol, and allows to generate the recipients' public and private keys. Thus recipients can be added through the protocol without the need to reset the protocol. Nevertheless, this also means that there is a central authority (CA) holding this master secret key, while PKE does not need such requirement.

One drawback is that the ciphertext has to embed information on the set of recipients used for its generation. This means that a recipient knows the composition of the set since it has to use his/her private key as well as the public keys of the other recipients belonging to the set in order to decrypt the ciphertext. Observe that successful decryption by colluding unauthorised recipients should be impossible when desiging a BE scheme.

Moreover, BE is efficient and fast since the process works from one encryptor to a set of authorised decryptors, while PKE only allows a one-to-one process. Another advantage is for the key revocation: this is easy to revoke a user by not including him/her in any set used for the generation of ciphertexts, and thus his/her private key becomes useless.

Existing Work. Broadcast encryption (BE) was introduced by Fiat and Naor [84]. Their scheme is a private-key scheme proved secure against an upper bounded number of colluders. Fully collusion secure private-key BE was first proposed by Naor et al. [167] and public-key BE was first formalized [73]. In [167, 73], schemes are constructed based on the subset cover framework. Later, Boneh et al. [42] presented the first fully collusion-resistant public-key BE where ciphertexts have constant size, while in all the previous schemes, the size of the ciphertext was linear in the size of the target set. The authors in [42] proposed two schemes that are proved selectively CPA and CCA secure, respectively.

Thereafter, a dynamic BE system was proposed in [71] where its security is only partially adaptive. In addition, the scheme can be seen a revocation one such that the set of revoked users is selected at the time of encryption, and any user outside of this set is able to decrypt. Delerablée [70] suggested an identity-based BE (IBBE) construction that is proven selectively CPA secure.

Adaptive security was first proposed by Gentry and Waters [97]. The authors gave several systems achieving adaptive CPA security in the random oracle model, in particular BE schemes and IBBE schemes. They also managed to get constant-size ciphertexts for their constructions.

Lewko et al. [141] gave a revocation scheme and identity-based revocation scheme where private keys have small size. Waters [236] offered an efficient BE using dual system encryption framework. In [141, 236], the systems are secure under static assumptions, whereas the previous schemes are proved secure based on q-based assumptions.

More recently, the first adaptive CCA secure BE schemes were suggested by Phan et al. [180], where both the private keys and the ciphertexts achieve constant size.

Definition. A broadcast encryption (BE) scheme is composed of the following four algorithms [84]:

- $\mathsf{Setup}(\lambda, s) \to (params, msk)$. On inputs a security parameter λ and the total number of users s, output the public parameters $params$ and the master secret key msk.

- $\mathsf{KeyGen}(params, msk) \to \{sk_i\}_{i\in[1,s]}$. On input the public parameters $params$ and the master secret key msk, output the user private keys $sk_1, \cdots, sk_s$.

- $\mathsf{Encrypt}(params, S) \to (Hdr, K)$. On inputs the public parameters $params$ and a subset $S \subseteq [1,s]$, output a pair (Hdr, K) where Hdr is the header, referred as the broadcast ciphertext, and K is a symmetric key chosen from a finite key set.

  Let m be a message to be broadcast that should be decipherable precisely by the recipients in S. Let C_m be the encryption of m under the symmetric key K. The broadcast consists of (S, Hdr, C_m). The pair (S, Hdr) is often called the *full header* and C_m is often called the *broadcast body*.

- $\mathsf{Decrypt}(params, Hdr, S, i, sk_i) \to K/\perp$. On inputs the public parameters $params$, the header Hdr, the subset $S \subseteq [1,s]$, a user $i \in [1,s]$ and its private key sk_i, output K if $i \in S$; output $\perp$ otherwise.

  Then, the user i can use K to decrypt the broadcast body C_m and obtain the message body m.

Correctness. We require that a broadcast encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda, s)$, $\{sk_i\}_{i\in[1,s]} \leftarrow \mathsf{Keygen}(params)$ and $(Hdr, K) \leftarrow \mathsf{Encrypt}(params, S)$, and if $i \in S$, we have that $\mathsf{Decrypt}(params, Hdr, S, i, sk_i) = K$.

Membership Encryption

Let $P(G)$ denote a group token generated from a group G of users and a secret token S. Membership encryption (ME) implies that decryption satisfies the privacy-preserving group membership $A \in G$ given the token $P(G)$.

We suppose that the encryption takes a user A and the group token $P(G)$ as inputs. Thus, the decryption succeeds if and only if the recipient knows the group G of users and the secret token S, and the statement $A \in G$ is true.

A membership encryption can be converted into a membership proof. Indeed, given A and $P(G)$, decrypting a membership encryption with success means that $A \in P(G)$. In addition, a membership proof $A \in P(G)$ from a membership encryption is non-transferable, since the membership proof $A \in P(G)$ does not need to be published, and thus the privacy of $P(G)$ is not compromised. Another advantage is that membership encryption is efficient since it combines two steps into one: a membership proof that is implicitly checked and the encryption of a message. Finally, observe that membership encryption can be used for oblivious transfer protocol.

Existing Work. Privacy-preserving membership proof enables to prove that a user A belongs to a group G of users while the privacy is protected. Follow two directions taken by membership proof:

- Set membership proof for a user $A \in G$ given a token $P(A)$ [64, 50, 48]. The token $P(A)$ and the users in G are known by a verifier. The goal for a prover is to show that a user in $P(A)$ is in G without disclosing the identity of this user to the verifier. Note that the privacy is preserved for the involved user.
- Accumulator for witness for a user $A \in G$ given a token $P(G)$ [29, 23, 173, 108, 49, 17, 110]. The token $P(G)$ and a user A are known by a verifier. The goal for a prover is to show that A belongs to the group of users in $P(G)$ without disclosing the identities of the other users in G to the verifier. Note that the privacy is preserved for the non-involved users.

ME was introduced by Guo et al. [110] as an application for accumulator for witness. Their provably secure scheme contains the following features:

- the group token $P(G)$ has constant size with the maximum number accountability on users and does not depend on the number of users in G;
- the upper bound number of users in $P(G)$ is accountable;
- the ciphertext has constant size with the maximum number accountability on users and is linearly dependent on the length of security parameter.

Definition. A membership encryption (ME) scheme is composed of the following five algorithms [110]:

- $\mathsf{Setup}(\lambda, s, \{A_i\}_{i\in[1,s]}) \to (params, msk)$. On inputs a security parameter λ, an integer s and all the attributes $\{A_1, \cdots, A_s\}$, output the public parameters *params* and the master secret key *msk*.
- $\mathsf{GroupGen}(params, msk, G) \to (P(G), sk)$. On inputs the public parameters *params*, the master secret key *msk* and a group attribute $G = \{A_i\}_{i\in[1,k]}$ for $k \in [1, s]$, output a group token $P(G)$ and a private key *sk*.
- $\mathsf{Verify}(params, P(G), k) \to true/false$. On inputs the public parameters *params*, a group token $P(G)$ and an integer k, output *true* if the attribute number in $P(G)$ satisfies $|P(G)| \leq k$; output *false* otherwise.
- $\mathsf{Encrypt}(params, A, P(G), m) \to C$. On inputs the public parameters *params*, an attribute A, a group token $P(G)$ and a message m, output a ciphertext C.

- $\mathsf{Decrypt}(params, A, G, sk, C) \rightarrow m/\perp$. On inputs the public parameters *params*, the attribute A, the group attribute G, the private key sk and a ciphertext C, output the message m for a valid ciphertext; output $\perp$ otherwise.

Correctness. We require that a membership encryption scheme is correct if for any $\lambda, s \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda, s, \{A_i\}_{i\in[1,s]}$, $(P(G), sk) \leftarrow \mathsf{GroupGen}(params, msk, G)$ and $C \leftarrow \mathsf{Encrypt}(params, A, P(G), m)$, and if $A \in G$, we have that $\mathsf{Decrypt}(params, A, G, sk, C) = m$.

3.1.4 Proxy Re-Encryption and Derivatives

Proxy Re-Encryption

Proxy re-encryption (PRE) allows a proxy to transform a ciphertext computed under Alice's public key into one that can be opened by Bob's private key. An application of such primitive can be described as follows: Alice wants to temporarily forward her encrypted emails to Bob without giving him her private key. Thus, Alice, called the delegator, can ask a proxy to re-encrypt her emails into a format that Bob, called the delegatee, can decrypt using his own private key. Since the proxy is untrusted, PRE enables Alice to not provide her private key to the proxy.

PRE offers various properties including:

1. *Directionality:* unidirectional PRE allows re-encryption in one way while bi-directional PRE enables re-encryption in both ways.

2. *Interactivity:* Re-encryption keys are generated with the intervention of a third party and/or with some interactions between Alice and Bob.

3. *Transitivity:* The proxy can re-delegate decryption rights.

4. *Transferability:* The proxy and any set of colluding delegatees can re-delegate decryption rights.

5. *Transparency:* Neither the sender of ciphertexts nor any of the delegatees are aware of the existence of the proxy.

PRE schemes can decide to satsify some of the above properties and not the other ones. More important, a property can be met at the price of another one. Therefore, different advantages and disadvantages appear in function of the property choices.

Existing Work. Mambo and Okamoto [150] introduced the concept of decryption rights delegation. No long after, Blaze et al. [32] formalized the notion of "atomic proxy cryptography" where a semi-trusted proxy computes a function that converts ciphertexts for Alice into ciphertexts for Bob without seeing the underlying plaintext. Their scheme is bidirectional: ciphertexts can be diverted from Alice to Bob and from Bob to Alice. Morever, the delegation in this scheme is transitive, meaning that the proxy alone can create delegation rights between two parties that have never agreed on this.

Jakobsson [123] proposed a quorum-based protocol in which the proxy is divided into sub-components, each controlling a share of the re-encryption key. Thus, the keys of the delegator are safe as long as some of the proxies are honest. Zhou et al. [250] used a similar way to construct their re-encryption protocol.

Thereafter, Ivan and Dodis [120] suggested precise definitions of bidirectional and unidirectional proxy functions. The authors presented an unidirectional proxy encryption by sharing the user's private key between two parties. They also achieved to overcome the issue of the proxy alone by assigning new delegation rights. However, their systems do no change ciphertexts for Alice into ciphertexts for Bob such that Bob can decrypt them with his own private key. Instead, decryption is delegated by requiring Bob to store additional secrets.

Ateniese et al. [14] improved the re-encryption schemes by using bilinear maps and obtained that the master secret key of the delegator is safeguarded from a colluding proxy and delegatee. They constructed three unidirectional PRE schemes proven CPA secure. More recently, Canetti and Hohenberger [52] achieved CCA security for PRE systems. Simultaneously, Green and Ateniese [105] gave an identity-based PRE (IB-PRE) scheme that achieves CCA security.

Definition. A (unidirectional) proxy re-encryption (PRE) scheme is composed of the following six algorithms [32]:

- $\mathsf{Setup}(\lambda) \to params$. On input a security parameter λ, outptut the public parameters $params$.
- $\mathsf{KeyGen}(params) \to (pk, sk)$. On inputs the public parameters $params$, output the public and private key pair (pk, sk).
- $\mathsf{Encrypt}(params, pk_A, m) \to C_A$. On inputs the public parameters $params$, a public key pk_A and a message m, output a ciphertext C_A.
- $\mathsf{Decrypt}(params, sk_A, C_A) \to m / \perp$. On inputs the public parameters $params$, a private key sk_A and a ciphertext C_A, output m for a valid ciphertext; output $\perp$ otherwise.
- $\mathsf{RekeyGen}(params, pk_A, sk_A, pk_B, sk_B) \to rk_{A\to B}$. On inputs the public parameters $params$, a public key pk_A and the corresponding private key sk_A, and another public key pk_B and the corresponding private key sk_B, output the re-encryption key $rk_{A\to B}$.

  The input sk_B is sometimes omitted; when this happens, we say that ReKeyGen is non-interactive since the user B does not need to be involved in the generation of the re-encryption keys. The input sk_A may in some cases be replaced by the tuple $(rk_{A\to C}, sk_C)$.
- $\mathsf{ReEncrypt}(params, rk_{A\to B}, C_A) \to C_B$. On inputs the public parameters $params$, the re-encryption key $rk_{A\to B}$ and a ciphertext C_A, output another ciphertext C_B.

Correctness. We require that a proxy re-encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $params \leftarrow \mathsf{Setup}(\lambda)$, $(pk_A, sk_A) \leftarrow \mathsf{Keygen}(params)$ and $(pk_B, sk_B) \leftarrow \mathsf{Keygen}(params)$,

- Given $C_A \leftarrow \mathsf{Encrypt}(params, pk_A, m)$, we have that $\mathsf{Decrypt}(params, sk_A, C_A) = m$.
- Given $rk_{A\to B} \leftarrow \mathsf{RekeyGen}(params, pk_A, sk_A, pk_B, sk_B)$ and $C_B \leftarrow \mathsf{ReEncrypt}(params, rk_{A\to B}, C_A)$, we have that $\mathsf{Decrypt}(params, sk_B, C_B) = m$.

Ciphertext-Policy Attribute-Based Proxy Re-Encryption

Ciphertext-policy attribute-based proxy re-encryption (CP-AB-PRE) extends the notion of PRE by permitting a semi-trusted proxy to transform a ciphertext under an access policy to another ciphertext encrypting with the same plaintext under another access policy.

Since CP-AB-PRE is a combination of PRE and CP-ABE, advantages and disadvantages of both primitives are encountered. We recall the ABE is derived from IBE and enhances the flexibility of message sharing such that only users satisfying specific policy are able to recover the message. In CP-ABE, private keys are labeled with attribute sets and ciphertexts are associated with access structures that specify which kinds of private keys the receiver has to retain. We also recall that PRE extends PKE to support the delegation of decryption rights.

Existing Work. Liang et al. [146] introduced the primitive ciphertext-policy attribute-based proxy re-encryption (CP-AB-PRE). The authors extended the CP-ABE scheme given in [60] to obtain a CP-AB-PRE system, in which "AND" gates on positive and negative attributes are supported. Mizuno and Doi [164] offered a hybrid PRE construction such that it can bridge ABE and IBE: ciphertexts generated in the ABE setting can be converted to ciphertexts which can be decrypted in the context of IBE. Later, Luo et al. [149] gave a CP-AB-PRE scheme such that "AND" gates on multivalued and negative attributes are supported. All the aforementionned papers only obtained CPA secure CP-AB-PRE systems. Liang et al. [145] proposed the first CP-AB-PRE construction that is proved CCA secure and supports any monotonic access policy.

Definition. A (single-hop unidirectional) ciphertext-policy attribute-based proxy re-encryption (CP-AB-PRE) scheme is composed of the following six algorithms [146]:

- $\mathsf{Setup}(\lambda, \mathscr{U}) \rightarrow (params, msk)$. On inputs a security parameter λ and an attribute universe $\mathscr{U}$, output the public parameters *params* and the master secret key *msk*.

- $\mathsf{KeyGen}(params, msk, S) \rightarrow sk_S$. On inputs the public parameters *params*, the master secret key *msk* and an attribute set S, output a private key sk_S, such that sk_S is associated with the attribute set S.

- $\mathsf{Encrypt}(params, \mathbb{AS}, m) \rightarrow C_{\mathbb{AS}}$. On inputs the public parameters *params*, an access structure $\mathbb{AS}$ for attributes over $\mathscr{U}$ and a message m, output a ciphertext $C_{\mathbb{AS}}$.

  We assume that the access structure $\mathbb{AS}$ is included in the ciphertext $C_{\mathbb{AS}}$.

- $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) \rightarrow m/\perp$. On inputs the public parameters *params*, an attribute set S, the corresponding private key sk_S and a ciphertext $C_{\mathbb{AS}}$, output m if S satisfies $\mathbb{AS}$; output $\perp$ otherwise, indicating either $C_{\mathbb{AS}}$ is invalid or S does not satisfy $\mathbb{AS}$.

- $\mathsf{RekeyGen}(params, S, sk_S, \mathbb{AS}') \rightarrow rk_{S \rightarrow \mathbb{AS}'}$. On inputs the public parameters *params*, an attribute set S, the corresponding private key sk_S and an access structure $\mathbb{AS}'$ for attributes over $\mathscr{U}$, output a re-encryption key $rk_{S \rightarrow \mathbb{AS}'}$ that can be used to transform a ciphertext under $\mathbb{AS}$ to another ciphertext under $\mathbb{AS}'$ such that S satisfies $\mathbb{AS}$.

  Note that $\mathbb{AS}$ and $\mathbb{AS}'$ are disjoint: for any attribute x satisfying $\mathbb{AS}$, x does not satisfy $\mathbb{AS}'$.

- $\mathsf{ReEncrypt}(params, rk_{S \to \mathbb{AS}'}, C_{\mathbb{AS}}) \to C_{\mathbb{AS}'}$. On inputs the public parameters $params$, the re-encryption key $rk_{S \to \mathbb{AS}'}$ and a ciphertext $C_{\mathbb{AS}}$, output another ciphertext $C_{\mathbb{AS}'}$ if S satisfies $\mathbb{AS}$; output $\perp$ indicating either $C_{\mathbb{AS}}$ is invalid or S does not satisfy $\mathbb{AS}$.

Correctness. We require that a ciphertext-policy attribute-based encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda, \mathscr{U})$, $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$ and $sk_{S'} \leftarrow \mathsf{KeyGen}(params, msk, S')$,

- Given $C_{\mathbb{AS}} \leftarrow \mathsf{Encrypt}(params, \mathbb{AS}, m)$, if S satisfies $\mathbb{AS}$, we have that $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) = m$.
- Given $rk_{S \to \mathbb{AS}'} \leftarrow \mathsf{RekeyGen}(params, S, sk_S, \mathbb{AS}')$ and $C_{\mathbb{AS}'} \leftarrow \mathsf{ReEncrypt}(params, rk_{S \to \mathbb{AS}'}, C_{\mathbb{AS}})$, if S' satisfies $\mathbb{AS}'$, we have that $\mathsf{Decrypt}(params, S', sk_{S'}, C_{\mathbb{AS}'}) = m$.

3.1.5 Certificate-Based Encryption

The notion of certificate-based encryption (CBE) involves a certificate, that can be seen as a signature, to act not only as a certificate but also as a decryption key. Thus, a recipient needs both its private key and the most recent certificate, received from a certificate authority (CA), to decrypt a message. In other words, CBE is inspired by IBE through implicit certification and by PKE in that there is no escrow.

In a certificate-based setting, certificate revocation lists (CRL) or certificate status protocols can be used to check whether a private key has been compromised.

One drawback is that the CA can always produce valid certificates for users and thus sign on their behalf, meaning that users' decryption capabilities is affected. In addition, CBE equires a public-key infrastructure (PKI) that can increase the cost of setting up the protocol.

Existing Work. Micali [158] proposed a public-key infrastructure (PKI) system, called "Novomodo". The system involves a certificate authority (CA), one or more directories to distribute the certification information and users. The efficiency of "Novomodo" is better than the ones obtained by the PKI certificate revocation list (CRL) and online certificate status protocol. Following Micali's solution, Naor and Nissim [168] and Aiello et al. [3] simultaneously presented hash-based systems. Both of these protocols require binary trees to reduce the computation of the CA and the communication between the CA and a directory.

Gentry [93] introduced the concept of certificate-based encryption (CBE) that can be used to construct an efficient PKI while requiring less infrastructure than for the aforementionned systems. This primitive constrains a user to use both its private key and an up-to-date certificate delivered from some CA to decrypt the ciphertext. The author constructed a CBE scheme that was proved CCA secure in the random oracle model. Thereafter, Sur et al. [217] proposed the first multi-receiver CBE (MR-CBE) system to avoid the built-in key escrow problem while preserving the inferred certification of the multi-receiver IBE (MR-IBE). However, no formal security proof was provided. More recently, Fan et al. [79] improved the previous scheme and obtained an anonymous MR-CBE scheme. They defined proper security models and gave formal CPA security and user anonymity proofs in the random oracle model.

Definition. A certificate-based encryption (CBE) scheme is composed of the following six algorithms [93]:

- $\mathsf{Setup}(\lambda_1,t) \rightarrow (params,msk)$. On inputs a security parameter λ_1 and the total number of time periods t, output the public parameters $params$ and the master secret key msk.

  The public parameters $params$ include a description of a string space $\mathscr{S}$.

- $\mathsf{KeyGen}(\lambda_2,t) \rightarrow (pk,sk)$. On inputs a security parameter λ_2 and (optionally) the total number of time periods t, output the user public and private key pair (pk,sk).

- $\mathsf{Upd1}(params,msk,pk,l,n) \rightarrow Cert'_l$. On inputs the public parameters $params$, the master secret key msk, the user public key pk, a time period l and a string $n \in \mathscr{S}$, output a certificate $Cert'_l$ at the start of l.

- $\mathsf{Upd2}(params,l,Cert'_l,Cert_{l-1}) \rightarrow Cert_l$. On inputs the public parameters $params$, a time period l, a certificate $Cert'_l$ and (optionally) a certificate $Cert_{l-1}$, output a certificate $Cert_l$ at the start of l.

- $\mathsf{Encrypt}(params,pk,m,l,n) \rightarrow C$ On inputs the public parameters $params$, the user public pk, a message m, a time period l and a string $n \in \mathscr{S}$, output a ciphertext C.

- $\mathsf{Decrypt}(params,sk,Cert_l,C,l) \rightarrow m/\perp$ On inputs the public parameters $params$, the user public key pk, the certificate $Cert_l$, a ciphertext C and a time period l, output m if $Cert_l$ is a valid certificate; output $\perp$ otherwise.

Correctness. We require that a certificate-based encryption scheme is correct if for any $\lambda_1,\lambda_2,t \in \mathbb{N}$, $(params,msk) \leftarrow \mathsf{Setup}(\lambda_1,t)$, $(pk,sk) \leftarrow \mathsf{KeyGen}(\lambda_2,t)$, $Cert'_l \leftarrow \mathsf{Upd1}(params, msk,pk,l,n)$, $Cert_l \leftarrow \mathsf{Upd2}(params,l,Cert'_l,Cert_{l-1})$ and $C \leftarrow \mathsf{Encrypt}(params,pk,m,l,n)$, we have that $\mathsf{Decrypt}(params,sk,Cert_l,C,l) = m$.

3.1.6 Digital Signatures

Digital signatures (DS) are electronic tokens that create binding between a user and a message, in order to validate and authenticate the message. Validating a message means certifying the contents of this message, and authenticating a message denotes certifying the sender of the document.

DS encounter various properties. First, they enable the recipient of a message to verify the sender (authenticity of the sender). Then, a digitally signed messagecannot be altered without invalidating the signature (authenticity of the message). Finally, a DS is used as a a sign of acknowledgement of a message (non-repudiation). Therefore, if a sender has digitally signed a message, s/he cannot deny this message.

Since a DS cannot be altered, committing fraud and forging signature are not possible. Given a DS of a message, we prove that this message is a valid one, and thus the recipient is ensured that the message is not forged or fake. DS enahnces security but is an additional cost when combined with another primitive.

Related Work. Signatures used to be a conventional way for authentication. Hellman and Diffie introduced the notion of public-key cryptography that developped the first practical method of distributing cryptographic keys over an unprotected public network [72].

Definition. The notion of digital signature (DS) was first proposed by Diffie and Hellman [72]. A digital signature scheme consists of the following algorithms:

- $\mathsf{Setup}(\lambda) \rightarrow params$. On input a security parameter λ, output the public parameters $params$.

- $\mathsf{KeyGen}(params) \rightarrow (vk, sk)$. On input the public parameters $params$, output a verifying (public) key vk and a signing (private) key sk.

- $\mathsf{Sign}(params, sk, m) \rightarrow \sigma$. On inputs the public parameters $params$, the signing key sk and a message m, output a signature σ.

- $\mathsf{Verify}(params, vk, m, \sigma) \rightarrow true/false$. On input the public parameters $params$, the verifying key vk, the message m and the signature σ, output $true$ if σ is a valid signature; output $false$ otherwise.

Correctness. We require that a digital signature scheme is correct if for any security parameter $\lambda \in \mathbb{N}$, $params \leftarrow \mathsf{Setup}(\lambda)$, $(vk, sk) \leftarrow \mathsf{Keygen}(params)$ and $\sigma \leftarrow \mathsf{Sign}(params, sk, m)$, we have that $\mathsf{Verify}(params, vk, m, \sigma) = true$.

3.1.7 Provable Data Possession

A provable data possession (PDP) allows a client to check the integrity of its data stored at an untrusted server without retrieving the entire document.

A basic PDP can be improved by adding various features, including:

1. *Data dynamics:* Basic PDP work on static data, such that the client is not able to update his/her stored data. Thus, one extension of basic PDP is to allow data operations, such as data insertion, data deletion and data modification.

2. *Reduced computational complexity:* The complexities generated by data integrity verifications should be reduced and even constant on both client's and server's sides, since auditing the untrusted server is crucial. Given constant complexities, the client can perform data integrity verifications regularly with less required computational resources.

3. *Block-less verification:* The data are divided into blocks and these blocks are then uploaded on the server. In order to check the stored data integrity, the client sends a challenge on some data blocks to the server. The property of block-less verification enables the challenged blocks to be efficienctly and securely retrieved by the client (as the verifier) during the audit of the server.

4. *Unboundedness:* Since integrity of the stored data is critical, the client should not be limited on the number of times that s/he can process the data integrity verifications. Indeed, the client should be able to audit the server as much as s/he wants or needs.

5. *Communication or network overhead:* A PDP should reduce the network communication overhead as much as possible, since data integrity verifications imply exchanges of parts of the data.

6. *Public verifiability:* Basic PDP require that the client himself/herself audits the server to check the data integrity. The property of public verifiability allows everyone to proceed the data integrity verifications.

7. *Privacy preservation:* When the data integrity verification is proceeded by a third party auditor (that is not the client), then PDP must ensure that this third party auditor learns nothing on the challenged data blocks.

The above features are desirable or not regarding of which problem the PDP primitive should solve. Note that some properties can be satisfied at the price of other ones.

Existing Work. Ateniese et al. [12] introduced the notion of provable data possession (PDP) and presented a scheme in which the verification metadata are homomorphic authenticators. The scheme is only designed for static data and the precomputation of the verification metadata imposes heavy computation overhead. Thereafter, Ateniese et al. [13] proposed symmetric key-based schemes to improve the scalability and the efficiency of the audit process compared to the scheme given in [12]. In addition, these shemes partially support dynamic data operations (updates, deletions and appends at block level); however, they are privately verifiable and limited in number of verification requests.

Subsequently, several works were presented following the models given in [12, 13]. Wang et al. [230] combined homomorphic authenticators based on the Boneh-Lynn-Shacham signature scheme [43] with Merkle hash trees [157] to achieve a public auditing protocol with fully dynamic data. Yu et al. [243] joined the techniques of ABE, PRE and lazy re-encryption (LRE) [19, 18] to obtain a scheme that guarantees fine-grainedness, scalability and data confidentiality of the access control. Hao et al. [112] designed a dynamic public auditing system based on RSA cryptosystem. However, the authors did not provide any proof of security; their scheme was shown not to be secure with respect to data privacy in [245]. Consequently, Yu et al. [245] improved the scheme given [112] that is proved to be data private. Erway et al. [77] proposed a fully dynamic PDP (DPDP) scheme based on rank-based authenticated dictionaries [102]. Unfortunately, their construction is very inefficient. Zhu et al. [252] used index hash tables to sustain fully dynamic data and achieved to construct a zero-knowledge PDP. Zhu et al. [251] created a dynamic audit service based on fragment structure, random sampling and index hash tables that supports timely anomaly detection. Wang et al. [229] proposed a system to ensure the correctness of user data stored on multiple servers by requiring homomorphic tokens and erasure codes in the auditing process. Le and Markopoulou [137] constructed an efficient dynamic remote data integrity checking scheme based on homomorphic message authentication code (MAC) [99, 100] and CPA secure encryption. Note that this scheme is specifically designed for network coding based storage cloud. Wang et al. [228] gave a flexible distributed storage integrity auditing protocol utilizing a homomorphic token and distributed erasure-coded data.

Later, Wang et al. [225] designed a privacy-preserving protocol, called “Oruta”, that allows public auditing on shared data stored in the cloud. They exploited ring signatures to compute the verification information needed to audit the integrity of shared data. The scheme allows public auditing and identity privacy; nevertheless, it fails to support large groups and traceability. In a parallel work, Wang et al. [224] presented a privacy-preserving auditing system, called “Knox”, for data stored in the cloud and shared among a large number of users in the group. The authors used group signatures [55] to construct homomorphic authenticators. The scheme permits identity privacy, large number of users and traceability but is only for private auditing. Note that in [225, 224], the identity of the user signing each block in the shared data is kept private from a third party auditor (TPA), while this TPA is still able to verify the integrity of the shared data without retrieving the entire document. However, Yu et al. [247] investigated the active adversary attacks in existing auditing protocols for shared data in the cloud, including the two identity privacy preserving auditing systems “Oruta” [225] and “Knox” [224], along with the above distributed storage integrity auditing system [228]. More precisely, the authors showed that these schemes become insecure when active adversaries are involved in the cloud storage (i.e. they can alter the cloud data without being detected by the auditor in the verification phase). Yang et al. [241] presented a survey of the previous works on data auditing. Subsequently, Wang et al. [227] proposed another privacy-preserving scheme with public auditing. Nevertheless, Fan et al. [80] showed that this scheme cannot guarantee that information is not leaked since indistinguishability is not achieved. The authors then gave a new definition of data privacy based on indistinguishability, along with an improved version of the aforementioned scheme [227] that satisfies the new security model.

Wang et al. [226] suggested a protocol, called “Panda”, that achieves user revocation by using proxy re-signature[32]. Their scheme also reaches public verifiability, data dynamicity (by using index hash tables) and batch auditing. However, Yu et al. [246] showed that the system “Panda” is not secure since a server can hide data loss without being detected. The authors in [246] proposed a solution to overcome the issue that keeps the properties of “Panda”.

Definition. A provable data possession (PDP) scheme is composed of the following four algorithms [12]:

- $\mathsf{KeyGen}(\lambda) \rightarrow (pk, sk)$. On input the security parameter λ, output the public and private key pair (pk, sk).

- $\mathsf{TagGen}(pk, sk, m) \rightarrow T_m$. On inputs the public key pk, the private sk and a document m, output a verification metadata T_m.

  The client splits the document m into data blocks and stores all of them in an ordered collection $\mathbb{F}$ and the corresponding verification metadata T_m in an ordered collection $\mathbb{E}$. It forwards these two collections to a server and deletes them from its local storage.

- $\mathsf{GenProof}(pk, F, chal, \Sigma) \rightarrow \nu$. On inputs the public key pk, an ordered collection $F \subset \mathbb{F}$ of data blocks, a challenge $chal$ and an ordered collection $\Sigma \subset \mathbb{E}$ which are the verification metadata corresponding to the data blocks in F, output a proof of data possession ν for the data blocks in F that are determined by the challenge $chal$.

- $\mathsf{CheckProof}(pk, sk, chal, \nu) \rightarrow treu/false$. On inputs the public key pk, the private key sk, the challenge $chal$ and the proof of data possession ν, output $true$ if ν is a correct proof of data possession for the data blocks determined by $chal$; output $false$ otherwise.

Correctness. We require that a provable data possession scheme is correct if for any $\lambda \in \mathbb{N}$, $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$, $T_m \leftarrow \mathsf{TagGen}(pk, sk, m)$ and $\nu \leftarrow \mathsf{GenProof}(pk, F, chal, \Sigma)$, we have that $\mathsf{CheckProof}(pk, sk, chal, \nu) = true$.

Public Verifiability

Public verifiability implies that everyone can check that the proof of data possession ν generated by the server is a correct one, meaning that the private key sk is no longer required in the algorithm $\mathsf{CheckProof}$. For instance, a third party auditor (TPA), on behalf of the client, can be asked to verify that the server correctly stores the documents. The algorithm $\mathsf{CheckProof}$ is modified as follows [204]:

- $\mathsf{CheckProof}(pk, chal, \nu) \to true/false$. On inputs the public key pk, the challenge $chal$ and the proof of data possession ν, output $true$ if ν is a correct proof of data possession for the data blocks determined by $chal$; output $false$ otherwise.

Correctness. We require that a provable data possession scheme is correct if for any $\lambda \in \mathbb{N}$, $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$, $T_m \leftarrow \mathsf{TagGen}(pk, sk, m)$ and $\nu \leftarrow \mathsf{GenProof}(pk, F, chal, \Sigma)$, we have that $\mathsf{CheckProof}(pk, chal, \nu) = true$.

Remarks. A first challenge $chal_C$ might be generated by the client and forwarded to the TPA. Then, the TPA, on behalf of the client, generates a challenge $chal$ based on $chal_C$ and sends it to the server. If the client wants to check the integrity of its document without the help of the TPA, then $chal_C = chal$.

Dynamicity

Dynamicity implies that the client can ask the server to perform data operations on data blocks, namely insertion, deletion and modification. The client can then check that the server has correctly done the data operations. A dynamic provable data possession (DPDP) scheme is composed of the four aforementioned algorithms KeyGen, $\mathsf{Tag\ Gen}$, $\mathsf{GenProof}$ and $\mathsf{CheckProof}$, along with the two following algorithms [77]:

- $\mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info) \to (\mathbb{F}', \mathbb{E}', \nu')$. On inputs the public key pk, the previous collection $\mathbb{F}$ of all the data blocks, the previous collection $\mathbb{E}$ of all the corresponding verification metadata, and the data operation details $info$ given by the client, output the updated verification metadata collection $\mathbb{F}'$, the updated verification metadata collection $\mathbb{E}'$, and the updating proof ν'.

  The element $info$ specifies the data operation to be performed (insertion, deletion or modification), along with other information like the rank where the operation has to be performed (if the data blocks are ordered), the data block itself and the corresponding metadata.

- $\mathsf{CheckOp}(pk, sk, \nu') \to true/false$. On inputs the public key pk, the private key sk and the updating proof ν', output $true$ if ν' is a correct updating proof; otherwise it outputs $false$.

Correctness. We require that a provable data possession scheme is correct if for any $\lambda \in \mathbb{N}$, $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$, $T_m \leftarrow \mathsf{TagGen}(pk, sk, m)$, and

- if $(\mathbb{F}', \mathbb{E}', \nu') \leftarrow \mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info)$, we have that $\mathsf{CheckOp}(pk, sk, \nu') = true$;
- if $\nu \leftarrow \mathsf{GenProof}(pk, F, chal, \Sigma)$, we have that $\mathsf{CheckProof}(pk, sk, chal, \nu) = true$.

Remarks. Public verifiability can apply for DPDP schemes. In this case, the private key *sk* is not taken as input in both algorithms CheckOp and CheckProof.

3.2 Cryptographic Tools

3.2.1 Hash Functions

A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is an algorithm that can map data of arbitrary length to data of a fixed length n. In this thesis, we require that hash functions have to be collision resistant. A family of collision resistant hash functions is a set of hash functions with the following properties [68]:

1. There is a PPT algorithm, which on input a security parameter λ selects uniformly and randomly a member of the family with the given value attached.
2. All functions in the family are computable in polynomial time.
3. The problem of finding $x \neq y$ such that $h(x) = h(y)$ for a given h in the family is computationally impossible to solve.

The random oracle model, introduced by Bellare and Rogaway [27] as a paradigm for designing efficient protocols, assumes that all parties including the adversary have access to a public, truly random hash function H. In practice, this ideal hash function H is instantiated as a concrete cryptographic hash function. Hence, from a theoretical perspective, a security proof in the random oracle model is only a heuristic indication of the security of the system. When instantiated with a specific hash function, this model is extremely useful for designing simple, efficient and highly practical solutions for many problems. On the contrary, in the standard model, i.e. without random oracles, there is no idealized oracle access, and the security is proven using only the standard complexity assumptions. Hence, from a security point of view, a proof in the standard model is preferable to a proof in the random oracle model.

Target Collision Resistant Hash Functions

Target collision resistant (TCR) hash functions were introduced by Cramer and Shoup [66].

A TCR hash function H guarantees that given a random element x which is from the valid domain of H, a PPT adversary $\mathscr{A}$ cannot find $y \neq x$ such that $H(x) = H(y)$. We let $Adv_{H,\mathscr{A}}^{TCR} = Pr[(x,y) \leftarrow \mathscr{A}(\lambda) : H(x) = H(y),\ x \neq y,\ x,y \in \mathbb{D}]$ be the advantage of $\mathscr{A}$ in successfully finding collisions from a TCR hash function H, where $\mathbb{D}$ is the valid input domain of H and λ is the security parameter. If a hash function is chosen from a TCR hash function family, then $Adv_{H,\mathscr{A}}^{TCR}$ is negligible.

Waters' Hash Function

Let $H : \{0,1\}^n \rightarrow \mathbb{G}_1$ be the hash function used in Waters' IBE and Signature schemes [235]. First, pick at random $n+1$ exponents $e_0, e_1, \cdots, e_n \in_R \mathbb{Z}_p$ and then compute $h_i = g^{e_i}$ for $i \in [0,n]$. Let $h = (h_0, h_1, \cdots, h_n) \in \mathbb{G}_1^{n+1}$ be the public description of the hash function H.

The algebraic hash function $H : \{0,1\}^n \rightarrow \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as the product $h(w) = e_0 + \sum_{i=1}^{n}(e_i \cdot w_i)$ and $H(w) = h_0 \cdot \prod_{i=0}^{n}(h_i^{w_i}) = g_1^{h(w)}$. The part $h(w)$ can be seen as the private key that protects the keyword.

In [235], Waters compared his IBE scheme with the Boneh and Boyen's IBE system [35] in terms of security. Let u be a random element in $\mathbb{G}_1$ and id be an identity. In Boneh and Boyen's scheme, id is seen as an element in $\mathbb{Z}_p$ whereas in Waters' scheme, $id = (id_1, \cdots, id_n) \in \{0,1\}^n$. Boneh and Boyen evaluated the element $u \cdot g_1^{id}$ while Waters considered the element $u \cdot \prod_{i \in \mathscr{S}} id_i$ where $\mathscr{S} \subset [1,n]$ is the set of all i for which $id_i = 1$. This difference makes that Waters' scheme is fully secure whereas Boneh and Boyen's scheme is only selectively secure.

3.2.2 Access Structure

The definition of access structure (AS) was introduced by Beimel [24].

Let $\{P_1, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{AS} \subseteq 2^{\{P_1, \cdots, P_n\}}$ is monotone if the following implication is satisfied for all B, C: if $B \in \mathbb{AS}$ and $B \subseteq C$, then $C \in \mathbb{AS}$. An (monotone) access structure is a (monotone) collection $\mathbb{AS} \subseteq 2^{\{P_1, \cdots, P_n\}} \setminus \{\emptyset\}$. We define the sets in $\mathbb{AS}$ as the *authorised sets* (we say also that these sets satisfy $\mathbb{AS}$) and the sets not in $\mathbb{AS}$ as the *unauthorised sets*.

In this thesis, the role of the parties will be taken by the attributes. Thus, the access structure $\mathbb{AS}$ will contain the authorised sets of attributes.

3.2.3 Linear Secret-Sharing Scheme

Beimel first defined the linear secret-sharing scheme (LSSS) [24].

Let Π be a secret-sharing scheme (SSS) over a set of parties P. Π is said *linear* over $\mathbb{Z}_p$ if:

1. The shares for each party form a vector over $\mathbb{Z}_p$.
2. There is a $l \times n$ matrix M (the share-generating matrix for Π). For all $i \in [1,l]$, the i-th row of M, written as M_i, is labeled by a party $\rho(i)$, where $\rho : [1,l] \rightarrow P$. Let $v = (s, r_2, \cdots, r_n)$ be a column vector, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \cdots, r_n \in \mathbb{Z}_p$ be randomly chosen. Thus, $M \cdot v$ is the vector of l shares of the secret s according to Π. The share $(M \cdot v)_i$ belongs to the party $\rho(i)$.

A linear SSS (LSSS) satisfies the following *linear reconstruction property*. Let Π be a LSSS for the access structure $\mathbb{AS}$, $S \in \mathbb{AS}$ be any authorised set (i.e. S satisfies $\mathbb{AS}$), and $I = \{i\}_{\rho(i) \in S} \subset [1,l]$. Therefore, there exist constants $\{w_i\}_{i \in I}$ in $\mathbb{Z}_p$ satisfying the following implication: if $\{\lambda_i\}_{i \in I}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} w_i \cdot \lambda_i = s$.

Convention. The vector $(1,0,\cdots,0)$ is the target vector for any LSSS. Let I be a set of rows of M. If I is an authorised set, then $(1,0,\cdots,0)$ is in the span of I. If I is a unauthorised set, then $(1,0,\cdots,0)$ is not in the span of I. There is a vector w such that $w \cdot (1,0,\cdots,0) = -1$ and $\forall i \in I$, we have that $w \cdot M_i = 0$.

3.3 Security Models

We succinctly present the security models for public-key cryptosystems. The schemes presented in this thesis will be proven secure according to the attacks defined below through games based on the hardness of the Diffie-Hellman problems. The goal of an attacker,that we call an *adversary*, in a given security game is to obtain information which reduces the security of a scheme. Note that a security model will be clearly exposed for each protocol given in this thesis.

3.3.1 Semantic security

Goldwasser and Micali [101] defined the notion of semantic security to capture the intuition that an adversary should not be able to get any piece of information about a plaintext given the resulting ciphertext. Plaintext are any messages, documents, file contents, etc, seen in plain (i.e. not encrypted). More precisely, even if the adversary has seen a ciphertext (i.e. the encrypted form of the plaintext), it cannot compute anything about the corresponding plaintext (and it could not computed it by itself beforehand). Such requirement should hold even if the adversary has some pieces of information about the plaintext.

Nevertheless, secrecy is guaranteed if and only if the adversary is completely passive; this means that it can only eavesdrop. In other words, semantic security does not offer any guarantee of secrecy if the adversary can trigger an active attack; this means that it can inject messages or influence the behavior of parties in the network.

3.3.2 Chosen-Plaintext Attack (CPA)

A chosen-plaintext attack (CPA) is a security model such that an adversary can obtain the ciphertexts for arbitrary plaintexts. More precisely, the adversary is allowed to interact with an encryption oracle viewed as a black box to query a ciphertext as an encryption of a given plaintext. CPA security does not enable an adversary to obtain decryption of ciphertexts of its choice (as in a chosen-ciphertext attack security).

3.3.3 Chosen-Ciphertext Attack (CCA)

Rackoff and Simon [183] introduced the notion of security against adaptive chosen-ciphertext attacks (CCA2) to deal with active adversarys. More precisely, if an adversary can inject plaintexts into a network, then these plaintexts may actually be ciphertexts, and so the adversary might be able to extract some information about the corresponding plaintexts through its interactions with the parties in the network. The security model enables an adversary to get decryptions of its choice by calling a decryption oracle. Note that given a challenge ciphertext, the adversary should not obtain any piece of information about the corresponding plaintext by restricting the adversary's power. In other words, it cannot give the challenge ciphertext to the decryption oracle but some ciphertexts that might be related to the challenge ciphertext.

Naor and Yung [170] first gave the security model in terms of (non-adaptive) chosen-ciphertext attacks (CCA1) where the adversary has access to the decryption oracle only prior to obtaining the challenge ciphertext. As above, the adversary's goal is to get information about the encrypted plaintext.

More precisly, CCA1 enables an adversary to obtain the plaintexts for arbitrary ciphertexts under an unknown key. The adversary is allowed to interact with a decryption oracle viewed as a black box to query a plaintext as a decryption of a given ciphertext. Thus, the adversary can try to recover the hidden private key used for decryption. Then, CCA2 differs from CCA1 in that the adversary forwards ciphertexts to be decrypted to the decryption oracle and uses the resulting plaintexts to send next ciphertexts. The adversary's goal is to progressively obtain pieces of information about plaintexts or even about the decryption (private) key.

Observe that in the public-key encryption context, CCA2 applies when ciphertexts are malleable; this means that a ciphertext can be changed in certain ways to give a predictable effect on its decryption and the resulting plaintext.

3.3.4 Non-Malleability

If an adversary is able to modify a ciphertext into another ciphertext which decrypts a related plaintext, then we say that the encryption is malleable. In other words, by using the ciphertext of a plaintext m and without necessarily knowing m, we can create another ciphertext that is the encryption of another plaintext $F(m)$, where F denotes a known function.

Dolev, Dwork, and Naor [74] first presented the notion of non-malleability. In this context, the adversary has access to a decryption oracle and submits a challenge ciphertext. The adversary's goal is to generate a ciphertext as the encryption of a plaintext such that this plaintext should be related to the one with its encryption equal to the challenge ciphertext.

Non-malleability and security against adaptive chosen-ciphertext attack were shown to be equivalent [26].

3.3.5 Indistinguishability

Indistinguishability (regarding the ciphertext) in a cryptosystem ensure that an adversary will not be able to distinguish two different ciphertexts based on the plaintext they encrypt.

Indistinguishability under CPA is equivalent to semantic security, and so is the first requirement for public-key cryptosystems to be proven secure. Indistinguishability under CCA1 or CCA2 is a stronger requirement that can be reached by some cryptosystems.

We say that a cryptosystem is secure in terms of indistinguishability if no adversary can determine what is the chosen plaintext with probability significantly greater than the random guessing probability (which is equal to $1/2$), given a ciphertext that encrypts a plaintext randomly chosen between two plaintexts submitted by the adversary. In other words, we want to ensure that the adversary should not be capable to do better than if it guesses randomly.

If an adversary successfully distinguishes the chosen ciphertext with a probability significantly greater than $1/2$, then we say that the adversary has advantage in distinguishing the ciphertext, and thus the scheme is not secure in terms of indistinguishability. We observe that the above definition includes the fact that the adversary should not learn any information from the challenge ciphertext.

Goldwasser and Micali [101, 159, 98, 100] proved that semantic security is equivalent to the notion of indistinguishability. These models imply security even when the adversary can trigger a CPA to obtain encryptions of plaintexts that it has chosen.

Indistinguishability under CPA (IND-CPA)

Indistinguishability under CPA (IND-CPA) is defined by the following game between an adversary and a challenger. For schemes based on computational security, the adversary is designed as a PPT Turing machine, meaning that it must complete the game and output a guess μ' within a polynomial number of time steps. We use the notations from the definition of a public-key encryption scheme in Section 3.1.1.

1. The challenger generates a key pair (ek, dk) based on some security parameter λ and gives ek to the adversary. The challenger keeps dk secret.
2. The adversary may perform any number of queries to the encryption oracle based on arbitrary plaintexts, or it may perform any number of other operations.
3. Eventually, the adversary submits two distinct chosen plaintexts (messages) m_0 and m_1 of equal length to the challenger.
4. The challenger selects a bit $\mu \in_R \{0,1\}$ and sends the challenge ciphertext $C = \mathsf{Encrypt}\ (params, ek, m_\mu)$ to the adversary.
5. The adversary performs again any number of additional queries and operations.
6. Finally, the adversary outputs a guess $\mu' \in \{0,1\}$ for μ.

We define the advantage of the adversary $\mathscr{A}$ in winning the above game as $Pr[\mu = \mu'] - 1/2$. A scheme is IND-CPA secure if no adversary has a non-negligible advantage in winning the above game.

Indistinguishability under CCA (IND-CCA1, IND-CCA2)

Indistinguishability under CCA (IND-CCA1 for the non-adaptive case and IND-CCA2 for the adaptive case) uses a definition similar to IND-CPA, except that the adversary is given access to a decryption oracle which decrypts arbitrary ciphertexts at the adversary's request and returns the corresponding plaintexts, in addition to have access to an encryption oracle.

In the non-adaptive definition, the adversary is allowed to query the decryption oracle only up until it receives the challenge ciphertext. In the adaptive definition, the adversary may continue to query to the decryption oracle even after it has received a challenge ciphertext, with the restriction that it cannot query the challenge ciphertext itself for decryption (otherwise, the definition would be trivial).

1. The challenger generates a key pair (ek, dk) based on some security parameter λ and gives ek to the adversary. The challenger keeps dk secret.
2. The adversary may perform any number of queries to the encryption oracle based on arbitrary plaintexts and to the decryption oracle based on arbitrary ciphertexts, or it may perform any number of other operations.

3. Eventually, the adversary submits two distinct chosen plaintexts (messages) m_0 and m_1 of equal length to the challenger.

4. The challenger selects a bit $\mu \in_R \{0,1\}$ and sends the challenge ciphertext $C = \mathsf{Encrypt}\ (params, ek, m_\mu)$ back to the adversary.

5. The adversary performs again any number of additional queries and operations.

6. Finally, the adversary outputs a guess $\mu' \in \{0,1\}$ for μ.

We define the advantage of the adversary $\mathscr{A}$ in winning the above game as $Pr[\mu = \mu'] - 1/2$. A scheme is IND-CCA1/IND-CCA2 secure if no adversary has a non-negligible advantage in winning the above game.

N.B. In our security models, we write A-IND-CCA as a shorthand for IND-CCA2, where the letter "A" stands for "adaptive".

Selective Indistinguishability under CCA (S-IND-CCA). A third model is well presented in the literature, called selective indistinguishability under CCA. In this security model, the adversary is given access to an encryption oracle and a decryption oracle, but with the constraint to submit the challenge at the beginning of the game played between the challenger and the adversary, before the challenger sets the keys used in the protocol. The challenger answers to the adversary's queries according to this challenge.

3.3.6 Relations between the Security Models

The notation $P_1 \Rightarrow P_2$ means that property P_1 implies property P_2, the notation $P_1 \Leftrightarrow P_2$ means that properties P_1 and P_2 are equivalent, and the notation $P_1 \not\Rightarrow P_2$ means that property P_1 does not necessarily imply property P_2. Let NM-CPA define non-malleability under chosen-plaintext attack and NM-CCA2 refer to non-malleability under adaptive chosen-ciphertext attack. The following list recalls some (non) implications and equivalences:

- IND-CPA $\Leftrightarrow$ semantic security under CPA.

- NM-CPA $\Rightarrow$ IND-CPA.

- NM-CPA $\not\Rightarrow$ IND-CCA2.

- NM-CCA2 $\Leftrightarrow$ IND-CCA2.

Chapter 4

Broadcasting and Sharing Electronic Health Records among the Involved Users

Electronic health records (EHRs) are a powerful tool to help the hospital staff members to obtain the necessary information to medically intervene. Therefore, hospital staff members naturally wish to be able to broadcast and share among them the medical information. Nevertheless, an implementation of the above situation is not straightforward. Indeed, we have to ensure that the hospital staff members can efficiently and easily proceed when dealing with EHRs among them, as well as that the security of the EHRs is maintained since some members and outsiders might misbehave.

We present two primitives to enable the hospital staff members to broadcast and share among them EHRs and other medical documents. The first one allows a user to broadcast medical information to everyone, such as only a receiver able to prove his/her belonging to group specially chosen for this broadcast can recover the information The second one enables a user to broadcast medical information to everyone, such as only a receiver belonging to a group specified for this broadcast and being granted with an up-to-date license delivered by a health legislator can recover the information.

Observe that the two above primitives require more than just forwarding encrypted EHRs to a group of selected hospital staff members. Indeed, the hospital staff members have to bring extra evidence of either their belonging to the group or certifying that they are authorised as of today.

4.1 Broadcast Encryption with Membership

Let us consider the following scenario. A medical institute has the responsibility to distribute private keys to the staff members of a hospital. These components will enable the hospital staff members to deal with the medical institute on retrieving patients' EHRs and other medical documents in the future.

Regularly, the hospital delivers a membership list to the medical institute. This list contains the identities and possibly extra information of some staff members working at the hospital and can be seen as an authorisation to access medical information. For instance, the list might contain only the identities of the medical staff members from the traumatology service (including surgeons, anesthetists, nurses, physiotherapists, ...). We assume that the list does not reveal the identities or any related information of the authorised hospital staff members to the medical institute. More precisely, only the hospital gets the personal information of the staff members that it has selected, along with the number of these staff members that should not be greater than a bound chosen beforehand.

Then, based on this hidden list, the medical institute encrypts some requested EHRs and medical documents that the authorised hospital staff members need to access and retrieve. Note that the resulting encrypted documents are forwarded to all the hospital staff members, i.e. the staff members authorised by the list and the non-authorised staff members. Upon receiving the encrypted documents, a hospital staff member will be able to successfully decrypt using his/her private key if and only if s/he is an authorised staff member of the aforementioned list. We depict such scenario in Figure 4.1.

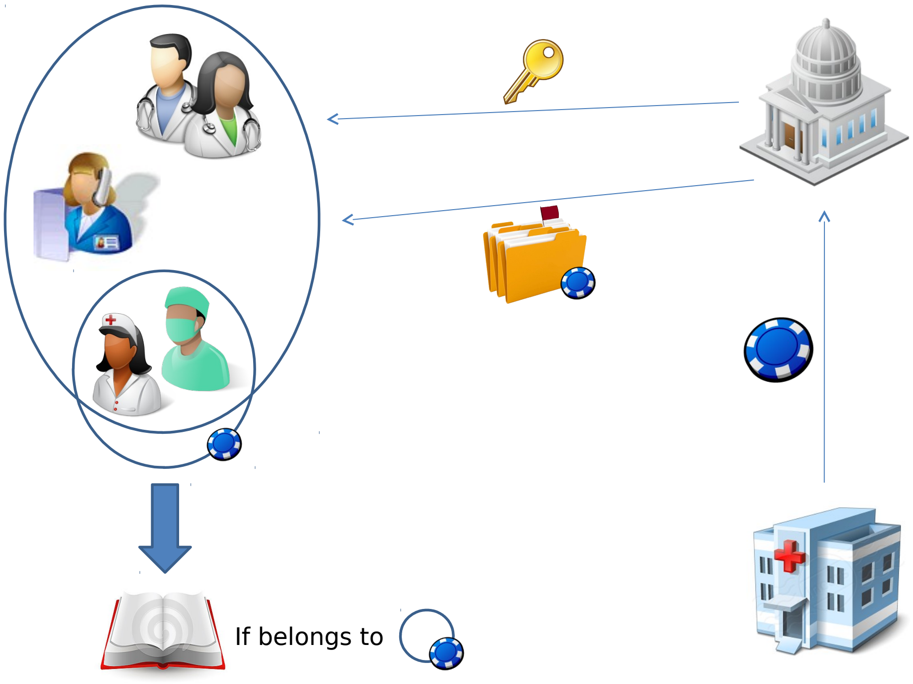


Figure 4.1: Broadcast Encryption with Membership. A medical institute, acting as a group manager and a broadcaster, computes the public and private key pairs for all the hospital staff members. Then, a hospital chooses a group G of hospital staff members and delivers a token $P(G)$ to the medical institute. The latter used $P(G)$ to compute a ciphertext C that encrypts a message m regarding the group G, and forwards it to all the hospital staff members. A hospital staff member can recover m if and only if s/he belongs to G.

4.1.1 Contributions

We introduce the notion of broadcast encryption with membership (BEM) to implement the above scenario. In a BEM system, a sender generates the public and private key pairs for all the users. A not-fully-trusted third party selects some users and hides their identities in a token that is given to the sender. The latter then encrypts a message according to this token and broadcasts the resulting ciphertext for all the users. Only the selected users are enable to successfully decrypt the ciphertext.

We provide a BEM construction that is inspired by Gentry-Waters broadcast encryption (BE) scheme [97] and by Guo et al.'s membership encryption (ME) scheme [110], with some subtle changes. We should note that a trivial combination between these two schemes will result to an insecure scheme. We also prove that the BEM construction is semi-statically IND-CCA secure, preserves the privacy and guarantees the property of maximum number accountability in the standard model.

Semi-static IND-CCA security model comes from Gentry and Waters' paper [97]. The authors proved their BE scheme secure regarding this model. In this model, an adversary must commit to a group G at the beginning of the security game played with a challenger. The adversary can only attack any group $G^* \subseteq G$. Note that the adversary has more flexibility than in the static IND-CCA security model. Gentry and Waters [97] also proposed a way to transform a semi-statically secure BE scheme into an adaptively secure BE scheme. We apply such transformation to our BEM scheme.

4.1.2 Protocol Definition

A broadcast encryption with membership (BEM) scheme is composed of the folowing six algorithms:

- $\mathsf{Setup}(\lambda, s) \rightarrow (params, msk)$. On inputs a security parameter λ and an integer s, output the public parameters $params$ and the master secret key msk.

- $\mathsf{KeyGen}(params, msk, \{A_i\}_{i\in[1,s]}) \rightarrow \{(pk_i, sk_i)\}_{i\in[1,s]}$. On input the public parameters $params$, the master secret key msk and all the users $\{A_i\}_{i\in[1,s]}$, output the user public and private key pairs $(pk_1, sk_1), \cdots, (pk_s, sk_s)$.

- $\mathsf{GroupGen}(params, msk, G, \{pk_i\}_{i\in G}) \rightarrow P(G)$. On inputs the public parameters $params$, the master secret key msk, a user group $G = \{A_i\}_{i\in[1,k]}$ for $k \in [1, s]$ and the public keys $pk_1, \cdots, pk_k$ of the users in G, output a group token $P(G)$.

- $\mathsf{Verify}(params, P(G), k) \rightarrow true/false$. On inputs the public parameters $params$, a group token $P(G)$ and an integer $k < s$, output $true$ if the user number in $P(G)$ satisfies $|P(G)| \leq k$; output $false$ otherwise.

- $\mathsf{Encrypt}(params, P(G), m) \rightarrow C$. On inputs the public parameters $params$, a group token $P(G)$ and a message m, output a ciphertext C.

- $\mathsf{Decrypt}(params, A_i, (pk_i, sk_i), G, C) \rightarrow m/\perp$. On inputs the public parameters $params$, a user A_i for $i \in [1, s]$, the public and private key pair (pk_i, sk_i) of this user A_i, the user group G and a ciphertext C, output the message m if $A_i \in G$; output $\perp$ otherwise.

Correctness. We require that a broadcast encryption with membership scheme is correct if for any $\lambda, s \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda, s)$, $\{(pk_i, sk_i)\}_{i\in[1,s]} \leftarrow \mathsf{KeyGen}(params, msk, \{A_i\}_{i\in[1,s]})$, $P(G) \leftarrow \mathsf{GroupGen}(params, msk, G, \{pk_i\}_{i\in G})$ and $C \leftarrow \mathsf{Encrypt}(params, P(G), m)$, and if $true \leftarrow \mathsf{Verify}(params, P(G), k)$ and $A_i \in G$, we have that $\mathsf{Decrypt}(params, A_i, (pk_i, sk_i), G, C) = m$.

4.1.3 Security Models

Semi-static Security Model

Recent BE systems [97, 180, 147] achieve adaptive IND-CCA security in the standard model. In the corresponding security game, the adversary is allowed to see the public parameters and the user public keys, and then adaptively requests user private keys before choosing the challenge user group G^* that it wishes to attack.

Nevertheless, in this section, we consider a weaker security definition, that is semi-static IND-CCA security. This notion was introduced by Gentry and Waters [97] to prove the security of their BE scheme. They also provided a generic technique to transform a semi-statically secure BE to an adaptively secure scheme. We note that their transformation is also applicable to BEM schemes, and so from the semi-statically secure BEM system presented below, we are able to extend it to a BEM system that is adaptively secure. We provide the transformation from semi-static security to adaptive security at the end of this section.

In the semi-static security game, the adversary must commit to a group G of users in an initialization phase that precedes the setup phase. The adversary cannot query a private key for any user $A_i \in G$, and it must choose a challenge group G^* for the challenge ciphertext such that $G^* \subseteq G$.

In the following, we define the security notion for semi-static indistinguishability against chosen-ciphertext attacks (IND-CCA). A BEM scheme is semi-statically IND-CCA secure if no PPT adversary $\mathscr{A}$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}$ as follows:

Initialization. The adversary gives to the challenger a group $G \subseteq \{A_i\}_{i\in[1,s]}$ of users that it wishes to be challenged on.

Setup. The challenger runs the algorithms Setup and KeyGen to obtain the public parameters and the public keys of users, and sends these elements to the adversary.

Query Phase 1. The adversary issues private key queries for users in $\{A_i\}_{i\in[1,s]} \setminus G$. The challenger answers by sending back to $\mathscr{A}$ the private keys $\{sk_i\}_{\{A_i\}_{i\in[1,s]}\setminus G} \leftarrow \mathsf{KeyGen}(params, msk, \{A_i\}_{i\in[1,s]} \setminus G)$.

Challenge. The adversary chooses a challenge group $G^* \subseteq G$, such that all the private key queries correspond to users $A_i \notin G^*$. The challenger runs the algorithm $\mathsf{GroupGen}$ with input the group G^* and obtains the group token $P(G^*)$. $\mathscr{B}$ then encrypts a message m under $P(G^*)$ to obtain the ciphertext C_0. Let C_1 be a random ciphertext from the ciphertext space. It sets $\mu \in_R \{0,1\}$ and gives $C^* = C_\mu$ to the adversary.

Guess. The adversary outputs a guess $\mu' \in \{0,1\}$ for μ, and wins the game if $\mu' = \mu$.

We define the advantage of the adversary $\mathscr{A}$ as $Adv^{SemiS}_{BEM,\mathscr{A}}(\lambda) = |Pr[\mu' = \mu] - 1/2|$. A BED scheme is semi-statically IND-CCA secure if for any PPT adversary, its advantage $Adv^{SemiS}_{BEM,\mathscr{A}}(\lambda)$ is a negligible function in λ.

Privacy Model

We consider the following requirements. Let the broadcaster be the one who encrypts a message m according to a group G of users and distributes the resulting ciphertext to all the users. The broadcaster knows the identities of all the users $A_1, \cdots, A_s$ since it generates their public and private key pairs (meaning that it runs the algorithm KeyGen). However, it does not know the identities of the users in the group $G = \{A_i\}_{i\in[1,k]}$ since it does not select them (meaning that the algorithm $\mathsf{GroupGen}$ is run by another party). Instead, the broadcaster only knows the maximum number of users in G since it gives the constraint that k has to be smaller than s (meaning that it runs the algorithm Verify).

The term "privacy" here does not refer to the complete anonymity towards the users. Nevertheless, we emphasize that the users chosen to be in G are ensured that their identities are kept secret from the broadcaster.

We require that a BEM system preserves the privacy of group users. Essentially, given a group token $P(G)$ where G represents a group of users, and two user groups $G_0 = \{A_i\}_{i\in[1,k_0]}$ and $G_1 = \{A'_i\}_{i\in[1,k_1]}$, it is computationally hard to decide whether $G = G_0$ or $G = G_1$.

In the following, we define the security notion for privacy. A BEM scheme is privately secure if no PPT adversary $\mathscr{A}$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}$ as follows:

Setup. The challenger runs the algorithms Setup and KeyGen to generate the public parameters *params* and the public keys $pk_1, \cdots, pk_s$, and sends these elements to the adversary.

Challenge. The adversary gives the challenger two user groups $G_0 = \{A_i\}_{i\in[1,k]}$ and $G_1 = \{A'_i\}_{i\in[1,k]}$ for $k < s$. The challenger responds by choosing a bit $\mu \in_R \{0,1\}$ at random, and by generating the token $P(G_\mu)$ for the user group G_μ. Finally, $\mathscr{B}$ sends $P(G_\mu)$ to the adversary.

Guess. The adversary outputs a guess $\mu' \in \{0,1\}$ of μ, and wins the game if $\mu' = \mu$.

We define the advantage of the adversary $\mathscr{A}$ as $Adv^{Privacy}_{BEM,\mathscr{A}}(\lambda) = |Pr[\mu' = \mu] - 1/2|$. A BEM scheme preserves the privacy of the group users if for any PPT adversary, its advantage $Adv^{Privacy}_{BEM,\mathscr{A}}(\lambda)$ is a negligible function in λ. We say the BEM scheme unconditionally preserves the privacy of the group users if $Adv^{Privacy}_{BEM,\mathscr{A}}(\lambda) = 0$.

Maximum Number Accountability Model

The property of maximum number accountability ensures that given a group token $P(G)$ for a user group G of cardinality equal to s, the broadcaster is guaranteed that the encrypted contents will only be decipherable by a maximum number s of users. Intuitively, a BEM system verifies the property of maximum number accountability if it is computationally hard to generate a group token $P(G)$ for G such that $|G| = s$, and the algorithm Verify outputs $|G| \leq k$ for $k < s$ chosen by the broadcaster.

In the following, we define the security notion for maximum number accountability. A BEM scheme guarantees the property of maximum number accountability if no PPT adversary $\mathscr{A}$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}$ as follows:

Challenge. The challenger runs the algorithms Setup and KeyGen to generate the public parameters *params* and the public keys $pk_1, \cdots, pk_s$. It also chooses a value $k < s$, and sends all these elements to the adversary.

Win. The adversary outptus $(P(G), G)$ and wins the game if G contains s users but the algorithm Verify outputs $true$, meaning that $|G| \leq k$.

We define the advantage of the adversary as $Adv_{BEM,\mathscr{A}}^{MaxNubAcc}(\lambda)$. A BEM system guarantees the property of maximum number accountability if for any PPT adversary, its advantage $Adv_{BEM,\mathscr{A}}^{MaxNubAcc}(\lambda)$ is a negligible function in λ.

4.1.4 Construction

Let $\mathscr{G}$ be an algorithm that, on input the security parameter λ, outputs a pairing group tuple $(p, \mathbb{G}_1, \mathbb{G}_T, e, g)$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are two multiplicative cyclic groups of prime order p, the map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is bilinear and g is a generator of $\mathbb{G}_1$.

- $\mathsf{Setup}(\lambda, s) \to (params, msk)$. First, run $(p, \mathbb{G}_1, \mathbb{G}_T, e, g) \leftarrow \mathscr{G}(\lambda)$. Then, choose $\alpha, \beta, \gamma \in_R \mathbb{Z}_p$ and $h \in_R \mathbb{G}_1$ at random. Compute $e(g^\gamma, g)$, $u_i = h^{\gamma\alpha^i}$ and $v_i = h^{\gamma\beta\alpha^i}$ for $i \in [0, s]$.

  The public parameters $params$ are set as $(p, \mathbb{G}_1, \mathbb{G}_T, e, g, h, e(g^\gamma, g), u_0, \cdots, u_s, v_0, \cdots, v_s)$ and the master secret key msk is set as (α, β, γ).

- $\mathsf{KeyGen}(params, msk, \{A_i\}_{i\in[1,s]}) \to \{(pk_i, sk_i)\}_{i\in[1,s]}$. Randomly choose $x_i, s_i \in_R \mathbb{Z}_p$ and $f_i \in_R \mathbb{G}_1$. Compute the private key $sk_i = (d_{i0}, \cdots, d_{is},)$ as follows:

  $$d_{i0} = g^{-s_i}, \quad d_{ii} = g^\gamma f_i^{s_i}, \text{ and for } i \neq j, \quad d_{ij} = f_j^{s_i}.$$

  Set the public key pk_i as $(x_i + \alpha, f_i)$.

- $\mathsf{GroupGen}(params, msk, G, \{pk_i\}_{i\in G}) \to P(G)$. Randomly choose $t \in_R \mathbb{Z}_p$ and compute $P(G)$ as

  $$\begin{aligned} P(G) &= (w_1, w_2, w_3, w_4, w_5, w_6) \\ &= (u_0^{t\cdot\prod_{A_i\in G}(x_i+\alpha)}, v_0^{t\cdot\prod_{A_i\in G}(x_i+\alpha)}, v_{s-k}^{t\cdot\prod_{A_i\in G}(x_i+\alpha)}, \prod_{A_i\in G} f_i^t, g^t, e(g^\gamma, g)^t) \\ &= (h^{\gamma t\cdot\prod_{A_i\in G}(x_i+\alpha)}, h^{\gamma\beta t\cdot\prod_{A_i\in G}(x_i+\alpha)}, h^{\gamma\beta t\alpha^{s-k}\cdot\prod_{A_i\in G}(x_i+\alpha)}, \prod_{A_i\in G} f_i^t, g^t, e(g^\gamma, g)^t) \end{aligned}$$

- $\mathsf{Verify}(params, P(G), k) \to true/false$. First, verify that $w_2 = w_1^\beta$ by checking $e(w_1, v_0) = e(w_2, u_0)$. If the last equality holds, and if $e(w_2, u_s) = e(w_3, u_k)$, then accept $|G| \leq k$ and output $true$; otherwise, output $false$.

- $\mathsf{Encrypt}(params, P(G), m) \to C$. First, verify that $w_2 = w_1^\beta$ by checking $e(w_1, v_0) = e(w_2, u_0)$. If the last equality holds, then proceed; otherwise, abort. Then, randomly choose $r \in_R \mathbb{Z}_p$. Given a message m, compute the ciphertext $C = (C_0, C_1, C_2, C_3)$ as follows:

  $$\begin{aligned} C_0 &= w_1^r = h^{r\gamma t\cdot\prod_{A_i\in G}(x_i+\alpha)} \\ C_1 &= w_5^r = g^{rt} \\ C_2 &= w_4^r = \prod_{A_i\in G} f_i^{rt} \\ C_3 &= m \cdot w_6^r \cdot e(C_0, g) = m \cdot e(g, g)^{rt\gamma} \cdot e(h^{r\gamma t\cdot\prod_{A_i\in G}(x_i+\alpha)}, g) \end{aligned}$$

- $\mathsf{Decrypt}(params, A_i, (pk_i, sk_i), G, C) \to m/\perp$. First, check the cardinality of the group G. If $|G| \le k$, then proceed; otherwise, abort. Then, compute the pairing

$$\begin{aligned} e_1 &= e(d_{ii} \cdot \prod_{A_j \in G \setminus \{A_i\}} d_{ij}, C_1) \cdot e(d_{i0}, C_2) \\ &= e(g^{\gamma} \cdot f_i^{s_i} \cdot \prod_{A_j \in G \setminus \{A_i\}} f_j^{s_i}, g^{rt}) \cdot e(g^{-s_i}, \prod_{A_j \in G} f_j^{rt}) \\ &= e(g^{\gamma}, g^{rt}) \cdot e(\prod_{A_j \in G} f_j^{s_i}, g^{rt}) \cdot e(g^{-s_i}, \prod_{A_j \in G} f_j^{rt}) \\ &= e(g^{\gamma}, g^{rt}) \end{aligned}$$

Retrieve m by computing

$$\begin{aligned} & e_1^{-1} \cdot C_3 \cdot e(h^{\gamma \cdot \prod_{A_i \in G}(x_i+\alpha)}, C_1) \\ = \; & e(g^{\gamma}, g^{rt})^{-1} \cdot m \cdot e(g^{\gamma}, g)^{rt} \cdot e(C_0, g) \cdot e(h^{\gamma \cdot \prod_{A_i \in G}(x_i+\alpha)}, C_1) \\ = \; & m \cdot e(h^{r\gamma t \cdot \prod_{A_i \in G}(x_i+\alpha)}, g) \cdot e(h^{\gamma \cdot \prod_{A_i \in G}(x_i+\alpha)}, g^{rt}) \\ = \; & m \end{aligned}$$

4.1.5 Security Proofs

Semi-static Security Proof

Intuitively, an adversary $\mathscr{A}$ must recover $e(g^{\gamma}, g)^{rt}$ in order to decrypt the ciphertext C. To do this, $\mathscr{A}$ must pair $C_1 = g^{rt}$ from the ciphertext with the components from some private keys sk_i of a user $A_i \in G^*$, where G^* is the challenge group chosen by $\mathscr{A}$. This will result in the desired value $e(g^{\gamma}, g)^{rt}$, which is hidden by $val = e(\prod_{A_i \in G^*} f_i^b, g^{rt})$ for an unknown exponent $b \in \mathbb{Z}_p$. This value val can cancel out if and only if the user has the correct key components. Since the hiding value val is randomized with the exponent b from the private keys of users $A_i \in G^*$, the attack of $\mathscr{A}$ will be successful with negligible probability.

Theorem. Suppose that the s-DBDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$, then the BEM scheme is semi-statically IND-CCA secure in the standard model.

Suppose there is an adversary $\mathscr{A}$ who can break the semi-statically IND-CCA security of the BEM scheme with advantage $Adv_{BEM,\mathscr{A}}^{SemiS}(\lambda) \ge \varepsilon$. We then construct a challenger $\mathscr{B}$ that can decide whether Z is either equal to $e(g,g)^{a^{s+1} \cdot b}$ or to a random element in $\mathbb{G}_T$. The simulator $\mathscr{B}$ plays the semi-statically IND-CCA game with $\mathscr{A}$ as follows.

$\mathscr{B}$ takes as inputs $(p, \mathbb{G}_1, \mathbb{G}_T, e, g) \leftarrow \mathscr{G}(\lambda)$ and a s-DBDHE instance $(g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, g^b, Z)$ where Z is either equal to $e(g,g)^{a^{s+1} \cdot b}$ or to a random element in $\mathbb{G}_T$.

Initialization. $\mathscr{A}$ commits to a group $G \subseteq \{A_i\}_{i \in [1,s]}$ of users.

Setup. First, $\mathscr{B}$ randomly selects $y_0, \cdots, y_s \in_R \mathbb{Z}_p$, and computes $f_j = g^{y_j}$ for $A_j \in G$ and $f_j = g^{y_j + a^j}$ for $A_j \in \{A_i\}_{i \in [1,s]} \setminus G$. In addition, $\mathscr{B}$ implicitly sets $\gamma = y_0 \cdot a^{s+1}$. It picks at random $h, v_i \in_R \mathbb{G}_1$ and $\alpha, \tau \in_R \mathbb{Z}_p$, and sets $u_i = (g^b)^{\tau \cdot \alpha^i}$ for $i \in [0, s]$. It sets the public parameters $params$ as $(p, \mathbb{G}_1, \mathbb{G}_T, e, g, h, e(g^{\gamma}, g), u_0, \cdots, u_s, v_0, \cdots, v_s)$, where $e(g^{\gamma}, g)$ can be computed as $e(g^a, g^{a^s})^{y_0}$.

Second, for $i \in [1,s]$, it calculates $x_i + \alpha$ for a random exponent $x_i \in_R \mathbb{Z}_p$. It set the public keys pk_i as $(x_i + \alpha, f_i)$. Finally, $\mathscr{B}$ sends $params$ and $\{pk_i\}_{i\in[1,s]}$ to $\mathscr{A}$.

Query Phase 1. $\mathscr{A}$ is allowed to query private keys for users in $\{A_i\}_{i\in[1,s]} \setminus G$. To answer the query, $\mathscr{B}$ randomly selects $z_i \in_R \mathbb{Z}_p$, computes $s_i = z_i - y_0 \cdot a^{s+1-i}$, and sets the private keys sk_i as $(d_{i0}, \cdots, d_{is})$, where

$$d_{i0} = g^{-s_i}, \quad d_{ii} = g^{\gamma} \cdot f_i^{s_i}, \text{ and for } j \neq i, \quad d_{ij} = f_j^{s_i}.$$

$\mathscr{B}$ can compute all these components from the problem instance. For instance,

$$d_{ii} = g^{\gamma} \cdot f_i^{s_i} = g^{y_0 \cdot a^{s+1} + (y_i + a^i)(z_i - y_0 \cdot a^{s+1-i})}$$

can be computed since the term a^{s+1} in the exponent cancels out.

Challenge. $\mathscr{A}$ chooses a group $G^* \subseteq G$ and selects two messages m_0 and m_1 of equal length. $\mathscr{B}$ picks at random a bit $\mu \in_R \{0,1\}$, and sets $b = rt$ for unknown exponents $r,t \in \mathbb{Z}_p$. We observe that $\prod_{A_i \in G^*}(x_i + \alpha) = \sum_{A_i \in G^*} X_i \alpha^i$ for $X_i \in \mathbb{Z}_p$. The challenger lets the challenge ciphertext C^* be (C_0, C_1, C_2, C_3), where $C_0 = \prod_{A_i \in G^*} u_i^{X_i}$, $C_1 = g^b$, $C_2 = \prod_{A_i \in G^*} f_i^b$ and $C_3 = m_\mu \cdot Z^{y_0} \cdot e(C_0, g)$. It sends C^* to $\mathscr{A}$.

$\mathscr{B}$ can compute these components from the problem instance as C_1 and C_3 come directly from it. Moreover, since $\mathscr{B}$ knows the logarithm $DL_g(f_i)$ for all $A_i \in G^*$, it can compute C_2 as $\prod_{A_i \in G^*} f_i^b = \prod_{A_i \in G^*} g^{y_i \cdot b} = g^{b \cdot \sum_{A_i \in G^*} y_i}$. The challenger can also compute C_0 as $\prod_{A_i \in G^*} u_i^{X_i} = \prod_{A_i \in G^*} (g^{b\tau \cdot \alpha^i})^{X_i} = g^{b\tau(\sum_{A_i \in G^*} \alpha^i \cdot X_i)}$.

Guess. Finally, $\mathscr{A}$ outputs a guess μ' for μ. The adversary wins if $\mu' = \mu$.

Analysis. The public and private keys are appropriately distributed since γ and the values $DL_g(f_i)$ and s_i are uniformly random and independent.

When $Z = e(g,g)^{a^{s+1} \cdot b}$, the ciphertext C^* is generated according to the same distribution as in the real scheme. Indeed, the component C_3 is equal to $m_\mu \cdot e(g,g)^{\gamma \cdot b} \cdot e(C_0, g)$, where $\gamma = y_0 \cdot a^{s+1}$, and thus the challenge ciphertext is valid under the randomness $b = rt$. Then, the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$.

When $Z \in_R \mathbb{G}_T$, the ciphertext $C^* = (C_0, C_1, C_2, C_3)$ is generated as follows: the component C_3 is equal to $m_\mu \cdot Z \cdot e(C_0, g)$ for $Z \in_R \mathbb{G}_T$. Thus, we get $Pr[\mu' = \mu] = \frac{1}{2}$.

It follows that we have $Adv_{\mathscr{B},\mathbb{G}_T}^{DBDHE}(\lambda) \geq \varepsilon$.

Privacy Proof

We prove that given $P(G)$, it is hard to know the users in $G_0 = \{A_i\}_{i\in[1,k]}$. To do so, we consider another user group $G_1 = \{A_i'\}_{i\in[1,k]}$ such that $|G_0| = |G_1|$. We show that $P(G_0)$ is a group token for both G_0 and G_1.

Theorem. Suppose that the DL assumption holds in $\mathbb{G}_1$, then the BEM scheme is privacy preserving in the standard model.

Let $\mathscr{A}$ be an adversary who breaks the property of unconditional privacy with advantage $Adv_{BEM,\mathscr{A}}^{UncPriv}(\lambda)$. Then, we construct a challenger $\mathscr{B}$ that solves the DL problem in $\mathbb{G}_1$ by interacting with $\mathscr{A}$ as follows.

$\mathscr{B}$ takes as inputs $(p, g, \mathbb{G}_1, \mathbb{G}_T, e) \leftarrow \mathscr{G}(\lambda)$ and a DL instance containing the tuple (g, g^t).

Setup. The challenger runs the algorithms Setup and KeyGen to generate the public parameters *params* and the public keys $pk_1, \cdots, pk_s$, and sends these elements to the adversary.

Challenge. The adversary gives the challenger two user groups G_0 and G_1 such that $|G_0| = |G_1| = k < s$. The challenger then chooses a bit $\mu \in_R \{0,1\}$ at random, and computes the token $P(G_\mu)$ for the user group G_μ as follows:

$$\begin{aligned} P(G_\mu) &= (w_1, w_2, w_3, w_4, w_5, w_6) \\ &= (u_0^{t\cdot\prod_{A_i\in G_\mu}(x_i+\alpha)}, v_0^{t\cdot\prod_{A_i\in G_\mu}(x_i+\alpha)}, v_{s-k}^{t\cdot\prod_{A_i\in G_\mu}(x_i+\alpha)}, \prod_{A_i\in G_\mu} f_i^t, g^t, e(g^\gamma, g)^t) \\ &= (h^{\gamma t\cdot\prod_{A_i\in G_\mu}(x_i+\alpha)}, h^{\gamma\beta t\cdot\prod_{A_i\in G_\mu}(x_i+\alpha)}, h^{\gamma\beta t\alpha^{s-k}\cdot\prod_{A_i\in G_\mu}(x_i+\alpha)}, \prod_{A_i\in G_\mu} f_i^t, g^t, \\ &\quad e(g^\gamma, g)^t) \end{aligned}$$

where $t \in_R \mathbb{Z}_p$. Finally, $\mathscr{B}$ sends $P(G_\mu)$ to the adversary.

Guess. The adversary outputs a guess $\mu' \in \{0,1\}$ of μ, and wins the game if $\mu' = \mu$.

Analysis. If the adversary can provide the random exponent t chosen by $\mathscr{B}$, then it can provide the group G_μ given $P(G_\mu)$. Since the adversary selects G_0 and G_1, it can generate the token $P(G_\nu)$, for $\nu \in \{0,1\}$, if it knows the randomness t. Thus, if $P(G_\nu) = P(G_\mu)$, then the adversary knows that $\mu = \nu$; otherwise, it knows that $\mu = 1 - \nu$.

Therefore, guessing μ is as guessing t from $P(G_\mu)$. In other words, finding t from $w_5 = g^t$ is as solving the DL problem in $\mathbb{G}_1$.

Maximum Number Accountability Proof

Intuitively, for users in $G = \{A_i\}_{i\in[1,k]}$ for $k < s$, we have

$$P(G) = (w_1, w_2, w_3, w_4, w_5, w_6) = (w_1, w_1^\beta, w_1^{\beta\alpha^{s-k}}, w_4, w_5, w_6).$$

From the algorithm Verify, the exponent in w_3 contains α^{s-k} which is known by the verifier. We have that $w_1 = h^{\gamma t\cdot\prod_{A_i\in G}(x_i+\alpha)}$, where the polynomial $\prod_{A_i\in G}(x_i + x)$ is at most of degree k. Otherwise, computing w_3 requires $h^{\gamma\beta\alpha^{s+1}}, \cdots, h^{\gamma\beta\alpha^{s'}}$ for $s' > s$ and these components are not generated during the algorithm Setup.

Theorem. Suppose that the (f,s)-DHE problem holds in $\mathbb{G}_1$, then the BEM scheme guarantees the property of maximum number accountability in the standard model.

Let $\mathscr{A}$ be an adversary who breaks the property of maximum number accountability with advantage $Adv_{BEM,\mathscr{A}}^{MaxNubAcc}(\lambda)$. Then, we construct a challenger $\mathscr{B}$ that solves the (f,s)-DHE problem in $\mathbb{G}_1$ by interacting with $\mathscr{A}$ as follows.

$\mathscr{B}$ takes as inputs $(p, g, \mathbb{G}_1, \mathbb{G}_T, e) \leftarrow \mathscr{G}(\lambda)$ and a (f,s)-DHE instance containing the tuple $(g, g^a, \cdots, g^{a^s})$.

Challenge. $\mathscr{B}$ generates the public components as follows. It first chooses $\beta, \gamma \in_R \mathbb{Z}_p$ at random, and implicitly makes α equal to a. It then randomly picks $y \in_R \mathbb{Z}_p$, and sets $h = g^y$. Thus, we obtain:

$$\begin{aligned} e(g^\gamma, g) &= e(g,g)^\gamma \\ u_i &= h^{\gamma\alpha^i} = (g^{a^i})^{y\gamma} \\ v_i &= h^{\gamma\beta\alpha^i} = (g^{a^i})^{y\beta\gamma} \end{aligned}$$

For $i \in [1,s]$, the challenger randomly chooses $s_i, x_i \in_R \mathbb{Z}_p$ and $f_i \in_R \mathbb{G}_1$. It implicitly sets the public key pk_i as $(x_i + \alpha, f_i) = (x_i + a, f_i)$ and the private key sk_i as $(d_{i0}, \cdots, d_{is})$, where $d_{i0} = g^{-s_i}$, $d_{ii} = g^\gamma f_i^{s_i}$ and for $j \neq i$, $d_{ij} = f_j^{s_i}$.

Note that all the terms are computable from the problem instance. $\mathscr{B}$ sends the public parameters and the public keys to $\mathscr{A}$.

Win. $\mathscr{A}$ outputs $(P(G), G)$ where $G = \{A_i\}_{i\in[1,k]}$.

Analysis. Suppose that the algorithm Verify accepts $|G| \leq k < s$ and outputs *true*. Let $P(G) = (w_1, w_2, w_3, w_4, w_5, w_6)$. If the algorithm Verify outputs $|G| = k' < k$, then we can write $P(G)$ as:

$$P(G) = (w_1, w_2, w_3, w_4, w_5, w_6) = (w_1, w_1^\beta, w_1^{\beta\alpha^{s-k}}, w_4, w_5, w_6).$$

Moreover, we get $w_1 = h^{t\cdot\prod_{A_i\in G}(\alpha+x_i)} = g^{y\cdot t\cdot\prod_{A_i\in G}(a+x_i)}$. Finally, $\mathscr{B}$ sets $f(x) = y\beta t \cdot x^{s-k'}\prod_{A_i\in G}(x_i + x)$, which is a $(s+k-k')$-degree polynomial function in $\mathbb{Z}_p[x]$, and outputs $(f(x), w_3)$ as the solution to the (f,s)-DHE problem.

Remarks. Note that the elements w_1, w_2, w_3 are involved in the verification of the group size, while the elements w_1, w_4, w_5, w_6 are involved in the generation of the ciphertext. Therefore, the component w_1 is common in both $P(G)$ and C and allows us to avoid the following situation. Imagine that w_1 is only computed for $P(G)$ but not used for C, meaning that there is no common component. Thus, a dealer might compute w_1, w_2, w_3 for less than k selected users while it might calculate w_4, w_5, w_6 for more than k selected users, without being noticed by the broadcaster. In addition, during the decryption, a user is able to check the size of the group G: if $|G| > k$, then the dealer's cheating behaviour is detected. This means that the broadcaster does not have any control on observing cheating behaviours and has to wait for users' notices to realize that the dealer is not acting honestly.

By involving w_1 in both the verification of the group size and the generation of the ciphertext, we enable the broadcaster to keep full control on dealers' behaviour.

4.1.6 From Semi-Static Security to Adaptive Security

We recall that in a semi-static security game, the adversary has to limit its queries before the setup phase. This requirement is not found in an adaptive security game, where the adversary can "adaptively" query a challenger. Gentry and Waters [97] showed how to extend a semi-statically secure BE scheme into an adaptively secure BE scheme. To obtain such transformation, they applied the two-key methodology. In a two-key scheme, a trusted authority generates two private keys for a user, while it forwards only one of the two keys to this user. Another user has to encrypt twice a message, i.e. one encryption for each key. Note that the encryptor does not know which key was delivered to the first user.

Let the tuple $\mathsf{BEM}_{SS} = (\mathsf{Setup}_{SS}, \mathsf{KeyGen}_{SS}, \mathsf{GroupGen}_{SS}, \mathsf{Verify}_{SS}, \mathsf{Encrypt}_{SS}, \mathsf{Decrypt}_{SS})$ be a semi-statically secure BEM scheme. Let EDSym be a semantically secure symmetric scheme, where $\mathsf{EncryptSym}$ and $\mathsf{DecryptSym}$ are the two algorithms for symmetric encryption and symmetric decryption respectively. We construct an adaptively secure BEM scheme $\mathsf{BEM}_A = (\mathsf{Setup}_A, \mathsf{KeyGen}_A, \mathsf{GroupGen}_A, \mathsf{Verify}_A, \mathsf{Encrypt}_A, \mathsf{Decrypt}_A)$ as follows:

- $\mathsf{Setup}_A(\lambda, s) \rightarrow (params, msk)$. Run $(params, msk') \leftarrow \mathsf{Setup}_{SS}(\lambda, 2s)$. Pick at random $\zeta \in_R \{0,1\}^s$. Set $msk = (msk', \zeta)$.

- $\mathsf{KeyGen}_A(params, msk, \{A_i\}_{i\in[1,s]}) \rightarrow \{(pk_i, sk_i)\}_{i\in[1,s]}$. Run $\{(pk_i, sk_i')\}_{i\in[1,s]} \leftarrow \mathsf{KeyGen}_{SS}(params, msk, \{A_{2i-\zeta_i}\}_{i\in[1,s]})$. Set $sk_i = (sk_i', \zeta_i)$ for $i \in [1,s]$.

- $\mathsf{GroupGen}_A(params, msk, G, \{pk_i\}_{i\in G}) \rightarrow P(G)$. Create a random set of $|G|$ bit as follows: $\tau_i \in_R \{0,1\}$ for $i \in G$ and then, $\tau = \{\tau_i\}_{i\in G}$. Let $G_0 = \{2i - \tau_i\}_{i\in G}$ and $G_1 = \{2i - (1 - \tau_i)\}_{i\in G}$. Run $P(G_0) \leftarrow \mathsf{GroupGen}_{SS}(params, msk, G_0, \{pk_i\}_{i\in G_0})$ and $P(G_1) \leftarrow \mathsf{GroupGen}_{SS}(params, msk, G_1, \{pk_i\}_{i\in G_1})$. Set $P(G) = (P(G_0), P(G_1), \tau)$.

- $\mathsf{Verify}_A(params, P(G), k) \rightarrow true/false$. Run $true/false \leftarrow \mathsf{Verify}_{SS}(params, P(G_0), \lceil\frac{k}{2}\rceil)$ and $true/false \leftarrow \mathsf{Verify}_{SS}(params, P(G_1), \lceil\frac{k}{2}\rceil)$. Note that if k is odd, the maximum number becomes $k+1$.

- $\mathsf{Encrypt}_A(params, P(G), m) \rightarrow C$. Parse $P(G)$ as $(P(G_0), P(G_1), \tau)$. Pick at random η_0, η_1 from the message space. Run $\mathsf{Encrypt}_{SS}(params, P(G_0), \eta_0) \rightarrow C_0$ and $\mathsf{Encrypt}_{SS}(params, P(G_1), \eta_1) \rightarrow C_1$. Then, run $C_0' \leftarrow \mathsf{EncryptSym}(params, \eta_0, m)$ and $C_1' \leftarrow \mathsf{EncryptSym}(params, \eta_1, m)$ (η_0 and η_1 play the roles of the symmetric keys). Set $C = (C_0, C_0', C_1, C_1', \tau)$.

- $\mathsf{Decrypt}_A(params, A_i, (pk_i, sk_i), G, C) \rightarrow m/\perp$. Set G_0 and G_1 as above. Parse C as $(C_0, C_0', C_1, C_1', \tau)$ and sk_i as (sk_i', ζ_i). Run $\eta_{\zeta_i\oplus\tau_i} \leftarrow \mathsf{Decrypt}_{SS}(params, A_i, (pk_i, sk_i), G_{\zeta_i\oplus\tau_i}, C_{\zeta_i\oplus\tau_i})$. Then, run $m \leftarrow \mathsf{DecryptSym}(params, \eta_{\zeta_i\oplus\tau_i}, C'_{\zeta_i\oplus\tau_i})$.

Theorem. Suppose that the BEM scheme $\mathsf{BEM}_{SS} = (\mathsf{Setup}_{SS}, \mathsf{KeyGen}_{SS}, \mathsf{GroupGen}_{SS}, \mathsf{Verify}_{SS}, \mathsf{Encrypt}_{SS}, \mathsf{Decrypt}_{SS})$ is semi-statically secure, then the BEM scheme $\mathsf{BEM}_A = (\mathsf{Setup}_A, \mathsf{KeyGen}_A, \mathsf{GroupGen}_A, \mathsf{Verify}_A, \mathsf{Encrypt}_A, \mathsf{Decrypt}_A)$ is adaptively secure (given that EDSym is a semantically secure symmetric scheme).

Let $\mathscr{A}$ be an adaptive adversary that attacks the adaptive BEM scheme BEM_A. Then, there exists four challengers $\mathscr{B}_1$, $\mathscr{B}_2$, $\mathscr{B}_3$ and $\mathscr{B}_4$ that play the security game with $\mathscr{A}$.

The proof is constructed as a sequence of games. Let E_i be the event that $\mathscr{A}$ wins the game $Game_i$, for $i \in [0,4]$.

$Game_0$. The game $Game_0$ is identical to the adaptive IND-CCA security game. Therefore, we obtain that $|Pr[E_0] - 1/2| = Adv^{A-IND-CCA}_{\mathsf{BEM}_A,\mathscr{A}}(\lambda)$.

$Game_1$. The game $Game_1$ is identical to the game $Game_0$, except that the challenger $\mathscr{C}_1$ randomly chooses $P(G_0)$ when constructing the challenge ciphertext. There exists a challenger $\mathscr{B}_1$ that breaks the security of the scheme BEM_{SS} as follows. First, $\mathscr{B}_1$

picks at random $\zeta \in_R \{0,1\}^s$ and sets the group $\tilde{G}$ as the set $\{2i-(1-\zeta_i)\}_{i\in[1,s]}$. It then forwards $\tilde{G}$ to $\mathscr{C}_1$, which gives back the public parameters *params*. $\mathscr{B}_1$ then sends *params* to the adversary $\mathscr{A}$.

When the adversary $\mathscr{A}$ makes queries on the private keys of the BEM scheme BEM_A, for $i \in [1,s]$, the challenger $\mathscr{B}_1$ makes queries on the private keys of the BEM scheme BEM_{SS}, for $2i-\zeta_i$. The challenger $\mathscr{C}_1$ answers by sending back sk_i' to $\mathscr{B}_1$, and then $\mathscr{B}_1$ answers by sending back $sk_i = (sk_i', \zeta_i)$ to $\mathscr{A}$.

Later, $\mathscr{A}$ asks for a challenge ciphertext on a group $G^* \subseteq [1,s]$. $\mathscr{B}_1$ computes $\tau = \{\tau_i = 1-\zeta_i\}_{i\in G^*}$ and then, sets $G_0 = \{2i-\tau_i\}_{i\in G^*}$ and $G_1 = \{2i-(1-\tau_i)\}_{i\in G^*}$. It also asks for a challenge ciphertext on G_0 to the challenger $\mathscr{C}_1$. $\mathscr{C}_1$ answers by choosing at random a bit $\mu \in_R \{0,1\}$ and by giving back $(C_0)_\mu \leftarrow \mathsf{Encrypt}_{SS}(params, (P(G_0))_\mu, (\eta_0)_\mu)$ and $(\eta_0)_\mu$. Then, the challenger $\mathscr{B}_1$ picks at random η_1 from the message space, and runs $\mathsf{Encrypt}_{SS}(params, P(G_1), \eta_1) \rightarrow C_1$. $\mathscr{B}_1$ also chooses two messages of equal length m_0 and m_1 and picks at random a bit $\mu^\dagger \in_R \{0,1\}$. In addition, $\mathscr{B}_1$ runs $C_0' \leftarrow \mathsf{EncryptSym}(params, (\eta_0)_\mu, m_{\mu^\dagger})$ and $C_1' \leftarrow \mathsf{EncryptSym}(params, \eta_1, m_{\mu^\dagger})$. Finally, the challenger $\mathscr{B}_1$ sets $C = ((C_0)_\mu, C_0', C_1, C_1', \tau)$ and gives it to the adversary $\mathscr{A}$.

Eventually, $\mathscr{A}$ gives a bit $\mu' \in \{0,1\}$ as a guess for μ. If $\mu' = \mu^\dagger$, then $\mathscr{B}_1$ sends 0 to $\mathscr{C}_1$; otherwise, it sends 1 to $\mathscr{C}_1$.

If $\mu = 0$, then the adversary's view is as in the game $Game_0$. The private keys sent by $\mathscr{B}_1$ are appropriately distributed. The element τ is uniformly random from the adversary's view, since the private key queries made by $\mathscr{A}$ only reveal the values ζ_i for $i \notin G^*$. Moreover, $(C_0)_0$ (generated with input $(P(G_0))_0$) is correctly generated, and the depedent values are also correctly generated. If $\mu = 1$, then the adversary's view is as in the game $Game_1$. Therefore, $|Pr[E_1] - Pr[E_0]| \leq Adv^{SS-IND-CCA}_{\mathsf{BEM}_{SS},\mathscr{B}_1}(\lambda)$.

$Game_2$. The game $Game_2$ is identical to the game $Game_1$, except that the challenger $\mathscr{C}_2$ randomly chooses $P(G_1)$ when constructing the challenge ciphertext. The analysis is similar to the one above, and so there exists a challenger $\mathscr{B}_1$ that breaks the security of the scheme BEM_{SS}. Following the above methodology, we can conclude that $|Pr[E_2] - Pr[E_1]| \leq Adv^{SS-IND-CCA}_{\mathsf{BEM}_{SS},\mathscr{B}_2}(\lambda)$.

$Game_3$. The game $Game_3$ is identical to the game $Game_2$, except that the challenger $\mathscr{C}_3$ randomly chooses a message m_0 from the message space. Then, the challenger computes $C_0' \leftarrow \mathsf{EncryptSym}(params, \eta_0, m_0)$. There exists a challenger $\mathscr{B}_3$ constructed as an adversary that attacks the semantic security of EDSym. Thus, we get that $|Pr[E_3] - Pr[E_2]| = Adv^{SemSec}_{\mathsf{EDSym},\mathscr{B}_3}(\lambda)$.

$Game_4$. The game $Game_4$ is identical to the game $Game_3$, except that the challenger $\mathscr{C}_4$ randomly chooses a message m_1 from the message space. Then, the challenger computes $C_1' \leftarrow \mathsf{EncryptSym}(params, \eta_1, m_1)$. There exists a challenger $\mathscr{B}_4$ constructed as an adversary that attacks the semantic security of EDSym. Thus, we get that $|Pr[E_4] - Pr[E_3]| = Adv^{SemSec}_{\mathsf{EDSym},\mathscr{B}_4}(\lambda)$.

Since the ciphertext C is independent of m_μ, for $\mu \in \{0,1\}$, and so of μ, we obtain that $|Pr[E_4] - 1/2| = 0$.

We conclude by giving the upper bound of the adversary's advantage:

$$\begin{aligned} Adv^{A-IND-CCA}_{\mathsf{BEM}_{A},\mathscr{A}}(\lambda) &\leq Adv^{SS-IND-CCA}_{\mathsf{BEM}_{SS},\mathscr{B}_1}(\lambda) + Adv^{SS-IND-CCA}_{\mathsf{BEM}_{SS},\mathscr{B}_2}(\lambda) \\ &\leq Adv^{SemSec}_{\mathsf{EDSym},\mathscr{B}_3}(\lambda) + Adv^{SemSec}_{\mathsf{EDSym},\mathscr{B}_4}(\lambda) \end{aligned}$$

4.1.7 Performance

In the following table, we evaluate the efficiency of our BEM symmetric pairing-based scheme. We use results of cryptographic operation implementations (group exponentiations of a random group element with a random exponent and pairing operations on random group elements) using the MIRACL framework [203, 196] for a 128-bit security level. All the following experiments are based on a dual core IntelR XeonR CPU W3503@2.40GHz with 2.0GB RAM running Ubuntu R10.04. The elliptic curve utilised for all the benchmarks was the super-singular symmetric curve $y^2 = x^3 + 1 \mod p$ with embedding degree 2 for suitable primes p.

We do not take into account multiplication/division in $\mathbb{G}_1$ and $\mathbb{G}_T$ as well as exponentiation in $\mathbb{G}_T$ as these operations have timings negligible compared to the ones for exponentiation in $\mathbb{G}_1$ and pairing operation.

	Group Exponentiation in $\mathbb{G}_1$	Pairing
Time/operation	34.4	176.6
Setup	6.983.2	176.6
KeyGen	$6,880$	
GroupGen	$3,577.6$	176.6
Verify		353.2
Encrypt	103.3	353.2
Decrypt		176.6

Table 4.1: Timings for our BEM symmetric pairing-based system. Times are in milliseconds. We assume that there are $s = 100$ users.

We note that the total time in the algorithms Setup and KeyGen is substantial, but we recall that these algorithms should be run only once to generate the public parameters and the static private keys for all the users. In the algorithm GroupGen, it requires $3,754.2$ milliseconds since the group G contains up to $s = 100$ users. Finally, in the algorithms Verify, Encrypt and Decrypt, it takes $353,2$ milliseconds, 456.5 milliseconds and 176.6 milliseconds respectively, mainly due to the cost of pairing computations: these results remain the same for every execution of the protocol, since the number of group exponentiation and pairing operation is constant.

4.1.8 Conclusion

We introduced the notion of broadcast encryption with membership (BEM). We presented a realistic scenario taking place in a e-Health context that requires such primitive. Furthermore, we defined a BEM scheme along with the security models. Subsequently, we presented a BEM construction which is provably semi-statically IND-CCA secure, privacy preserving and maximum number accountability secure in the standard model. Finally, we gave a possible way to transform a semi-statically IND-CCA secure BEM scheme into an adaptively secure BEM scheme.

4.2 Certificate-Based Broadcast Encryption

EHRs have become of paramount interest to health and security communities, due to their size as well as their security issues and sensitivity. We propose a framework that enables secure communication among medical staff members (general practitioners (GP), surgeons, anesthetists, gynaecologists, etc). The communication channel enabled here will allow medical staff members to communicate among themselves as well as to review the patients' EHRs and other medical documents. Additionally, it allows the hospital to broadcast any important sensitive information to the medical staff members working in that hospital, and this information will only be made available to them. The issue is that the staff members have their own rights in a hospital, delivered by a medical institute and other health legislators. Therefore, our framework should be able to specify which are the medical staff members that have access to the documents by updating their license as authorised in that hospital. To generalize this scenario, we decouple several entities involved in this scenario, namely the hospital (as the document owner), the medical institute and the other health legislators (as the certifiers) who grant the license for the medical staff members (as the users) to work in that hospital.

This scenario is depicted in Figure 4.2. We note that the hospital would be the entity that generates the encrypted documents for the medical staff members, which includes all important information that the latters need to access. Additionally, any other hopsital staff members (i.e. the staff members that do not belong to the medical staff, such as the administration staff members and the paramedical staff members) and the patients should also be able to send any information to the medical staff in that hospital. For instance, it represents the case when a patient provides his/her X-ray or his/her blood test results to the hospital.

4.2.1 Contributions

Based on the above scenario, we present a new cryptographic notion called certificate-based broadcast encryption (CBBE). In this system, there are four entities, namely the group manager, a group of certifiers, a group of users and the rest of the universe. Following the aforementioned scenario, the group manager represents the hospital, the certifiers represent a medical institute and other health legislators who grant the rights to some medical staff members to work in the given hospital. The group of users are the main players in this setting and represent the medical staff. The rest of the universe includes the patients and the other hospital staff members (that do not belong to the medical staff) and any other players who are not captured in the above description.

We also provide sound security models to capture this scenario. We request that a CBBE system should be secure against chosen-ciphertext attacks. Moreover, we require a CBBE scheme to be collusion resistant, which follows the original security requirement for broadcast encryption (BE) systems. More precisely, even if all the users in the system collude, only the users in the group chosen by the broadcaster and with valid certificates can recover the plaintext.

One may think that a CBBE scheme could be achieved easily by combining the two cryptographic primitives, namely certificate-based encryption (CBE) and BE schemes. Unfortunately, this is false. The primary difficulty of building such a scheme is due to the requirement of achieving short size ciphertexts, in comparison to merely combining the CBE ciphertexts and the BE ciphertexts into one with linear size in both the number of

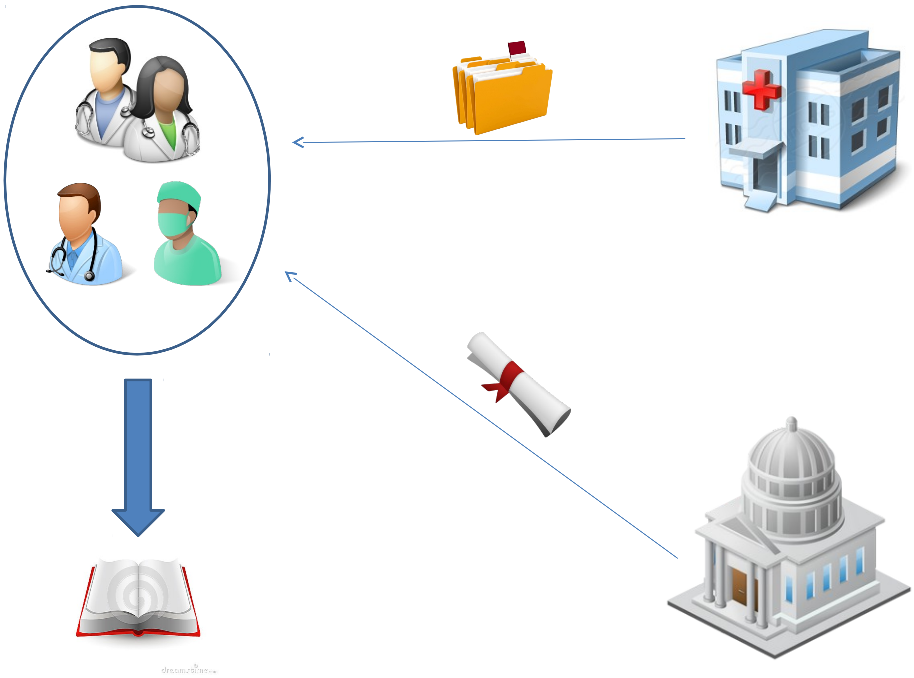


Figure 4.2: Certificate-Based Broadcast Encryption. A hospital, as a broadcaster, encrypts some medical information and forwards them to the medical staff members. A medical institute and other health legislators, as certifiers, deliver licenses to the medical staff members working in the aforementioned hospital. A medical staff member can recover the information in plain if and only if his/her licenses are up-to-date.

users and the number of certifiers.

Furthermore, the notion of CBE usually allows a sender to interact with only a single receiver, with the help of a certifier. Hence, a simple combination of the existing CBE and BE schemes will lead to a construction with a linear size of ciphertext (in the number of both users and certifiers), which is undesirable. Otherwise, it would be better for the broadcaster to just simply encrypt each ciphertext individually for each user. Therefore, using a subtle combination of CBE and BE systems, we achieve to obtain an efficient and practical CBBE construction with short ciphertexts, private keys and certificates, in the random oracle model.

In this chapter, we present two CBBE constructions as a combination of BE and CBE systems The first one is a trivial scheme CBBE_{basic} such that the ciphertext size is linear in the number of both users and certifiers involved in the system and the security is the adaptive IND-CCA one in the standard model assuming that both the BE and the CBE schemes are adaptively IND-CCA secure in the standard model. Conversely, the CBBE scheme CBBE_{effic} is efficient with short ciphertexts, private keys and certificates, and is selectively IND-CCA secure and selectively collusion resistant in the random oracle model. This scheme combines Boneh et al.'s BE [42] and Gentry's CBE [93] systems. In Table 4.2, we compare the components' size and the security level between existing schemes and ours.

	max	Size of PC	Size of SK	Size of $Cert$	Size of Hdr	ecurity level
[42] first BE scheme	0	$O(s)$	$O(1)$	$O(1)$	$O(1)$	selective IND-CPA in the standard model
[93] CBE scheme	1	$O(s)$	$O(1)$	$O(1)$	$O(1)$	adaptive IND-CCA in the random oracle model
[217] multi-receiver CBE scheme	1	$O(s)$	$O(1)$	$O(1)$	$O(s)$	n/a
[79] multi-receiver CBE scheme	1	$O(s)$	$O(1)$	$O(1)$	$O(s)$	selective IND-CPA in the random oracle model
our basic scheme CBBE_{basic}	k	$O(k \cdot s)$	$O(1)$	$O(1)$	$O(k \cdot s)$	adaptive IND-CCA in the standard model
our efficient scheme CBBE_{effic}	k	$O(k \cdot s)$	$O(1)$	$O(1)$	$O(1)$	selective IND-CCA in the random oracle model

Table 4.2: Comparison among various schemes. Let max denote the maximum number of invovled certifiers, PC denote the public components (including the public parameters and all the public keys), SK denote all the private keys, $Cert$ denote the certificates and Hdr denote the headers. Let s be the number of users and k be the number of certifiers invovled in the system. We note that $k = 0$ in [42] and $k = 1$ in [93, 217, 79]. We extend the Gentry's CBE scheme [93] to s users interacting with one certifier. We consider the size of one private key for one user or one certifier, the size of one certificate for one user given one certifier, and the size of one header given one message and two groups of users and certifiers respectively.

4.2.2 Protocol Definition

A certificate-based broadcast encryption (CBBE) scheme is composed of the following four algorithms:

- $\mathsf{Setup}(\lambda,s,k) \to (params,\{(pk_i,sk_i)\}_{i\in[1,s]},\{(pk_{c_j},sk_{c_j})\}_{j\in[1,k]})$. On inputs the security parameter λ, the total number s of users and the total number k of certifiers, output the public parameters *params* and the public and private key pairs (pk_i,sk_i) for users $i \in [1,s]$ (note that an algorithm KeyGen - to generate the keys of the involved entities - is implicitly embedded in the algorithm Setup).

  We suppose that the public and private key pairs (pk_{c_j},sk_{c_j}) for certifiers $j \in [1,k]$ are generated by the certifiers themselves.

- $\mathsf{Certif}(params,S_c,\{(pk_{c_j},sk_{c_j})\}_{j\in S_c},pk_i,l) \to \{Certif_{i,j,l}\}_{j\in S_c}$. On inputs the public parameters *params*, the group $S_c \subseteq [1,k]$ of certifiers selected by the broadcaster, the certifier c_j's public and private key pair (pk_{c_j},sk_{c_j}) for all $j \in S_c$, a user i's public key pk_i for $i \in [1,s]$ and a time period l, output the certificates $Certif_{i,j,l}$ for i, j and l.

- $\mathsf{Encrypt}(params,S_u,S_c,\{pk_i\}_{i\in S_u},\{pk_{c_j}\}_{j\in S_c},l) \to (Hdr,K)$. On inputs the public parameters *params*, a group $S_u \subseteq [1,s]$ of users selected by the broadcaster, a group $S_c \subseteq [1,k]$ of certifiers selected by the broadcaster, the user i's public key pk_i for all $i \in S_u$, the certifier c_j's public key pk_{c_j} for all $j \in S_c$ and the time period l, output the header Hdr and the session key K.

- $\mathsf{Decrypt}(params,S_u,S_c,l,(pk_i,sk_i),\{Certif_{i,j,l}\}_{j\in S_c},Hdr) \to K/\perp$. On inputs the public parameters *params*, the group $S_u \subseteq [1,s]$ of users selected by the broadcaster, the group $S_c \subseteq [1,k]$ of certifiers selected by the broadcaster, the time period l, the user i's public and private key pair (pk_i,sk_i) and the certificates $Certif_{i,j,l}$ for all $i \in [1,s]$, $j \in S_c$ and l, and the header Hdr, output the session key K if $i \in S_u$ and all the $Certif_{i,j,l}$ are valid for time period l; otherwise $\perp$.

Correctness. We require that a certificate-based broadcast encryption scheme is correct if for any $\lambda,s,k \in \mathbb{N}$, $(params,\{(pk_i,sk_i)\}_{i\in[1,s]},\{(pk_{c_j},sk_{c_j})\}_{j\in[1,k]}) \leftarrow \mathsf{Setup}(\lambda,s,k)$, $\{Certif_{i,j,l}\}_{j\in S_c} \leftarrow \mathsf{Certif}(params,S_c,\{(pk_{c_j},sk_{c_j})\}_{j\in S_c},pk_i,l)$ and $(Hdr,K) \leftarrow \mathsf{Encrypt}(params,S_u,S_c,\{pk_i\}_{i\in S_u},\{pk_{c_j}\}_{j\in S_c},l)$, and if $i \in S_u$ and all the $Certif_{i,j,l}$ are valid for a time period l and $j \in S_c$, we have that $\mathsf{Decrypt}(params,S_u,S_c,l,(pk_i,sk_i),\{Certif_{i,j,l}\}_{j\in S_c},Hdr) = K$.

N.B. Obersve that, in the algorithm Certif, a group of certifiers c_j such that $j \in S_c \subseteq [1,k]$ is required, and this group is the same one than the one used to generate the header and the session key. This means that a user needs exactly the certificates delivered by the certifiers belonging to a group determined by the broadcaster to be able to decrypt the message.

4.2.3 Security Models

Indistinguishability Chosen-Ciphertext Attack Model

We present the definitions for indistinguishability chosen-ciphertext attack (IND-CCA) security models. We adopt the security definitions from the CBE system given in [93].

There are two attacks which may be launched by either an uncertified user (i.e. a user with no valid certificate) or by an untrusted certifier that we capture in the following two games. In Game 1, the adversary plays the role of an uncertified user: it first proves that it knows the private key of the uncertified user and then, it can make decryption and certification queries. In Game 2, the adversary plays the role of a trusted certifier: it first proves that it knows the private key of the certifier and then, it can make decryption queries. Eventually, we say that the CBBE system is secure if no adversary can win either Game 1 or Game 2.

Game 1. In the following, we define the adaptive IND-CCA security notion against uncertified users. A CBBE scheme is adaptively IND-CCA secure if no PPT adversary $\mathscr{A}_1$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}_1$ as follows:

Setup. The challenger runs the algorithm Setup on input the tuple (λ, s, k) to obtain the public parameters *params*, the public keys pk_i for users $i \in [1,s]$ and the public keys pk_{c_j} for certifiers c_j such that $j \in [1,k]$, and gives them to $\mathscr{A}_1$.

Query Phase 1. The adversary can adaptively make the following queries:

- *Certification Query* $< l, i, S_c >$. Given a time period l, a user $i \in [1,s]$ and certifiers c_j such that $j \in S_c \subseteq [1,k]$, the challenger first checks that the element sk_i is the private key corresponding to the public key pk_i. If so, it runs the algorithm Certif and returns $Certif_{i,j,l}$ to the adversary $\mathscr{A}_1$; otherwise, it returns $\perp$.
- *Decryption Query* $< l, i, S_c, Hdr >$. Given a time period l, a user $i \in [1,s]$, certifiers c_j such that $j \in S_c \subseteq [1,k]$ and a header Hdr, the challenger first checks that the element sk_i is the private key corresponding to the public key pk_i. If so, it returns $K \leftarrow \mathsf{Decrypt}(params, S_u, S_c, l, (pk_i, sk_i), \{Certif_{i,j,l}\}_{j \in S_c}, Hdr)$ to the adversary $\mathscr{A}_1$; otherwise, it returns $\perp$.

Challenge. Given a time period l^*, $\mathscr{A}_1$ outputs a challenge group $S_u^* \subseteq [1,s]$. For a user $i \in S_u^*$, the challenger first checks that the element sk_i^* is the private key corresponding to the public key pk_i^*. If so, it chooses a group $S_c^* \subseteq [1,k]$ and computes $(Hdr^*, K^*) \leftarrow \mathsf{Encrypt}(params, S_u^*, S_c^*, \{pk_i^*\}_{i \in S_u^*}, \{pk_{c_j}^*\}_{j \in S_c^*}, l^*)$. It then chooses a random bit $\mu \in_R \{0,1\}$, sets $K_\mu = K^*$, picks at random $K_{1-\mu}$ in the key space, and gives (Hdr^*, K_0, K_1) to the adversary $\mathscr{A}_1$. Otherwise, it gives $\perp$.

Query Phase 2. The phase is similar to the query phase 1, except that (l^*, i, S_c^*), such that $i \in S_u^*$, is not subject to any valid certification query and (l^*, i, S_c^*, Hdr^*), such that $i \in S_u^*$, is not subject to a valid decryption query.

Guess. The adversary $\mathscr{A}_1$ outputs its guess $\mu' \in \{0,1\}$ for μ and wins the game if $\mu' = \mu$,

We define $\mathscr{A}_1$'s advantage in attacking the CBBE scheme following Game 1 as $Adv_{CBBE,\mathscr{A}_1}^{Game1}$ $(\lambda) = |Pr[\mu' = \mu] - \frac{1}{2}|$.

Game 2. In the following, we define the adaptive IND-CCA security notion against untrusted certifiers. A CBBE scheme is adaptively IND-CCA secure if no PPT adversary $\mathscr{A}_2$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}_2$ as follows:

Setup. The challenger runs the algorithm Setup on input the tuple (λ, s, k) to obtain the public parameters *params*, the public keys pk_i for users $i \in [1,s]$ and the public keys pk_{c_j} for certifiers c_j such that $j \in [1,k]$, and gives them to $\mathscr{A}_2$.

Query Phase 1. The adversary can adaptively make the following queries:

- *Decryption Query* $< l, i, S_u, S_c, Hdr >$. Given a time period l, a user $i \in [1,s]$,a group $S_u \subseteq [1,s]$ of user such that $i \in S_u$, certifiers c_j such that $j \in S_c \subseteq [1,k]$ and a header Hdr, the challenger first checks that the element sk_{c_j} is the private key corresponding to the public key pk_{c_j}. If so, it returns $K \leftarrow \mathsf{Decrypt}(params, S_u, S_c, l, (pk_i, sk_i), \{Certif_{i,j,l}\}_{j \in S_c}, Hdr)$ to the adversary $\mathscr{A}_2$; otherwise, it returns $\perp$.

Challenge. Given a time period l^*, $\mathscr{A}_2$ outputs a challenge group $S_c^* \subseteq [1,k]$. For a certifier c_j such that $j \in S_c^*$, the challenger first checks that the element $sk_{c_j}^*$ is the private key corresponding to the public key $pk_{c_j}^*$. If so, it chooses a group $S_u^* \subseteq [1,s]$ and computes $(Hdr^*, K^*) \leftarrow \mathsf{Encrypt}(params, S_u^*, S_c^*, \{pk_i^*\}_{i \in S_u^*}, \{pk_{c_j}^*\}_{j \in S_c^*}, l^*)$. It then chooses a random bit $\mu \in_R \{0,1\}$, sets $K_\mu = K^*$, picks at random $K_{1-\mu}$ in the key space, and gives (Hdr^*, K_0, K_1) to the adversary $\mathscr{A}_2$. Otherwise, it gives $\perp$.

Query Phase 2. The phase is similar to the query phase 1, except that $(l^*, i, S_u^*, S_c^*, Hdr^*)$, such that $i \in S_u^*$, is not subject to a valid deryption query.

Guess. The adversary $\mathscr{A}_2$ outputs its guess $\mu' \in \{0,1\}$ for μ and wins the game if $\mu = \mu'$

We define $\mathscr{A}_2$'s advantage in attacking the CBBE scheme following Game 2 as $Adv_{CBBE,\mathscr{A}_2}^{Game2}$ $(\lambda) = |Pr[\mu' = \mu] - \frac{1}{2}|$.

We say that a CBBE scheme is adaptively IND-CCA secure if no PPT algorithm $\mathscr{A}$ that plays the two games by making at most q_{cf} certification queries (in Game 1) and $q_d = q_{d_1} + q_{d_2}$ decryption queries (in the Game 1 and Game 2 respectively), has non-negligible advantage in either Game 1 or Game 2, i.e. $Adv_{CBBE,\mathscr{A}}^{Game1}(\lambda) + Adv_{CBBE,\mathscr{A}}^{Game2}(\lambda) = Adv_{CBBE,\mathscr{A}}^{A-IND-CCA}(\lambda)$ is negligible.

N.B. We prove that our efficient CBBE scheme is selectively secure. Note that the selective IND-CCA security model is weaker than the adaptive IND-CCA security model. In this case, an adversary $\mathscr{A}$ provides the group $S_u^* \subseteq [1,s]$ as the one that it wishes to be challenged on at the beginning of the security game 1, before the setup phase, as well as the group $S_c^* \subseteq [1,k]$ as the one that it wishes to be challenged on at the beginning of the security game 2, before the setup phase.

Selective Collusion Resistance Model

In the following, we define the security notion for selective collusion resistance. A CBBE scheme is selectively collusion resistant if no PPT adversary $\mathscr{A}$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}$ as follows:

Initialization. The adversary $\mathscr{A}$ outputs a group $S_{u,s'} \subseteq [1,s]$ of colluding users, such that $|S_{u,s'}| = s' \leq s$.

Setup. The challenger runs the algorithm Setup on input the tuple (λ, s, k) to obtain the public parameters *params*, the public and private keys pairs (pk_i, sk_i) for users $i \in S_{u,s'}$ and the public and private keys pair (pk_{c_j}, sk_{c_j}) for the certifier c_j such that $j \in [1,k]$. $\mathscr{B}$ gives *params*, $\{pk_i, sk_i\}_{i \in S_{u,s'}}$ and $\{pk_{c_j}\}_{j \in [1,k]}$ to the adversary $\mathscr{A}$ and keeps $\{sk_{c_j}\}_{j \in [1,k]}$.

$\mathscr{B}$ then runs the algorithm Certif to obtain the certificates $Certif_{i,j,l}$ for a user $i \in S_{u,s'}$, a certifier c_j such that $j \in S_c \subseteq [1,k]$ (for a group S_c that the challenger has chosen) and a time period l, and gives them to the adversary $\mathscr{A}$.

Challenge. The challenger returns $(Hdr, K) \leftarrow \mathsf{Encrypt}(params, S_u, S_c, \{pk_i\}_{i \in S_u}, \{pk_{c_j}\}_{j \in S_c}, l')$ for a group $S_u \subseteq [1,s]$ such that $S_u \cap S_{u,s'} = \emptyset$, for a group $S_c \subseteq [1,k]$ and a time period l' such that $l' \neq l$. $\mathscr{B}$ gives the header Hdr to the adversary $\mathscr{A}$ and keeps the session key K.

Win. The adversary $\mathscr{A}$ outputs

$$\begin{aligned} K^* \quad \leftarrow \quad & \mathsf{Decrypt}(params, S_u, S_c, l', (f(\{pk_i\}_{i \in S'}), f(\{sk_i\}_{i \in S'})), \\ & f(\{Certif_{i,j,l}\}_{i \in S', j \in S_c}), Hdr) \end{aligned}$$

where $S' \subseteq S_{u,s'}$ and f is a sujective function that takes as input either the public keys or private keys or the certificates, and outputs either a new public key or a new private key or a new certificate as a combination of public keys, private keys or certificates respectively. The adversary $\mathscr{A}$ wins the game if $K^* = K$.

We define $\mathscr{A}$'s advantage in attacking the CBBE scheme following the above game as $Adv_{CBBE,\mathscr{A}}^{S-CollRes}(\lambda) = Pr[\mathscr{A}$ succeeds].

A CBBE scheme is selectively collusion resistant if no PPT $\mathscr{A}$ has non-negligible advantage $Adv_{CBBE,\mathscr{A}}^{S-CollRes}(\lambda)$ in winning the above game.

4.2.4 Basic Construction: CBBE_{basic}

We consider a simple and straightforward combination between a BE scheme and a CBE scheme to obtain a CBBE scheme. Suppose that we are given an adaptive IND-CCA secure BE system with algorithms $(\mathsf{Setup}_{BE}, \mathsf{KeyGen}_{BE}, \mathsf{Encrypt}_{BE}, \mathsf{Decrypt}_{BE})$ and an adaptive IND-CCA secure CBE system with algorithms $(\mathsf{Setup}_{CBE}, \mathsf{KeyGen}_{CBE}, \mathsf{Upd1}_{CBE}, \mathsf{Upd2}_{CBE}, \mathsf{Encrypt}_{CBE}, \mathsf{Decrypt}_{CBE})$. Then, we build an adaptive IND-CCA secure CBBE scheme CBBE_{basic} with algorithms $(\mathsf{Setup}, \mathsf{Certif}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows.

- $\mathsf{Setup}(\lambda, s, k) \rightarrow (params, \{(pk_i, sk_i)\}_{i \in [1,s]}, \{(pk_{c_j}, sk_{c_j})\}_{j \in [1,k]})$. First run $(params, msk) \leftarrow \mathsf{Setup}_{BE}(\lambda, s)$ and then $\{sk_i^{BE}\}_{i \in [1,s]} \leftarrow \mathsf{KeyGen}_{BE}(params, msk)$.

  Second run $\mathsf{Setup}_{CBE}(\lambda, t) \rightarrow (params_j, msk_j)$ for $j \in [1,k]$, and then $(pk_i, sk_i^{CBE}) \leftarrow \mathsf{KeyGen}_{CBE}(\lambda, t)$ for $i \in [1,s]$.

  Finally, return the public parameters *params*, the user i's public and private key pair $(pk_i, sk_i) = (pk_i, (sk_i^{BE}, sk_i^{CBE}))$ for $i \in [1,s]$, and the certifier c_j's public and private key pair $(pk_{c_j}, sk_{c_j}) = (params_j, msk_j)$ for $j \in [1,k]$.

- $\mathsf{Certif}(params, S_c, \{(pk_{c_j}, sk_{c_j})\}_{j\in S_c}, pk_i, l) \to \{Certif_{i,j,l}\}_{j\in S_c}$. We recall that $(pk_{c_j}, sk_{c_j}) = (params_j, msk_j)$. Run $Cert'_{i,j,l} \leftarrow \mathsf{Upd1}_{CBE}(params_j, msk_j, pk_i, l, n)$ for a string n from the string space, a user $i \in [1,s]$, a certifier c_j such that $j \in S_c$ and a time period l. Run $Cert_{i,j,l} \leftarrow \mathsf{Upd2}_{CBE}(params_j, l, Cert'_{i,j,l}, Cert_{i,j,l-1})$. Finally, return the certificate $Certif_{i,j,l}$ for all $j \in S_c$.

- $\mathsf{Encrypt}(params, S_u, S_c, \{pk_i\}_{i\in S_u}, \{pk_{c_j}\}_{j\in S_c}, l) \to (Hdr, K)$. Let a group $S_u \subseteq [1,s]$ of users and a group $S_c \subseteq [1,k]$ of indices j such that c_j is a certifier.

  First run $(Hdr^{BE}, K) \leftarrow \mathsf{Encrypt}_{BE}(params, S_u)$. Split Hdr^{BE} in $|S_c|$ parts as $Hdr^{BE,1}$, $Hdr^{BE,2}, \cdots, Hdr^{BE,|S_c|}$. Second run $C_{i,j} \leftarrow \mathsf{Encrypt}_{CBE}(params_j, pk_i, Hdr^{BE,j}, l, n)$ for a string n from the string space, a user $i \in S_u$, a certifier c_j for $j \in S_c$, and a time period l. Set $C_i = \{C_{i,j}\}_{j\in S_c}$. Finally, return $Hdr = \{C_i\}_{i\in S_u}$.

  A label $\mathscr{L}$ can be included in the header Hdr and contains information about how C_i is associated with user i and how $C_{i,j}$ is associated with certifier c_j.

- $\mathsf{Decrypt}(params, S_u, S_c, l, (pk_i, sk_i), \{Certif_{i,j,l}\}_{j\in S_c}, Hdr) \to K/\perp$.

  For $i \in S_u$ and $j \in S_c$, run $Hdr^{BE,j} \leftarrow \mathsf{Decrypt}_{CBE}(params_j, sk_i^{CBE}, Certif_{i,j,l}, C_{i,j}, l)$, concatenate the $|S_c|$ values $Hdr^{BE,j}$ to obtain Hdr^{BE}, and then run $K \leftarrow \mathsf{Decrypt}_{BE}(params, Hdr^{BE}, S_u, i, sk_i^{BE})$.

Correctness. The correctness of the CBBE system simply follows from the correctness of both the BE system and the CBE system.

4.2.5 Security Proofs for CBBE_{basic}

Theorem. A basic CBBE scheme CBBE_{basic} achieves adaptive IND-CCA security in the standard model if the BE scheme and the CBE scheme are both adaptively IND-CCA secure in the standard model.

Let $\mathscr{A}$ be an adversary that attacks the basic CBBE scheme. Then, there exist a challenger $\mathscr{B}_1$ attacking the BE system and a challenger $\mathscr{B}_2$ attacking the CBE system, such that $Adv^{A-IND-CCA}_{\mathsf{CBBE}_{basic},\mathscr{A}}(\lambda) \leq Adv^{A-IND-CCA}_{BE,\mathscr{B}_1}(\lambda) + Adv^{A-IND-CCA}_{CBE,\mathscr{B}_2}(\lambda)$.

We present the security proof as a sequence of games. Let E_x be the event that $\mathscr{A}$ wins the game $Game_x$, for $x \in [0,2]$.

$Game_0$. $Game_0$ is identical to the real adaptive security game defined previously. Thus, $|Pr[E_0] - \frac{1}{2}| = Adv^{A-IND-CCA}_{\mathsf{CBBE}_{basic},\mathscr{A}}(\lambda)$.

$Game_1$. $Game_1$ is identical to $Game_0$, except that the challenge session key K is chosen at random from the key space by a challenger $\mathscr{C}_1$ that plays the adaptive IND-CCA game with $\mathscr{B}_1$ against the BE scheme. Therefore, $|Pr[E_1] - Pr[E_0]| = Adv^{A-IND-CCA}_{BE,\mathscr{B}_1}(\lambda)$.

The challenger $\mathscr{C}_1$ sets the public parameters $params$ and the users' public keys pk_i as in the real BE scheme and forwards them to $\mathscr{B}_1$ who then sends them to $\mathscr{A}$. The challenger $\mathscr{C}_2$ sets the certifiers' public keys $params_j$ as in the real CBE scheme and forwards them to $\mathscr{B}_2$ who then sends them to $\mathscr{A}$.

When $\mathscr{A}$ asks for a user CBBE private key for $i \in [1,s]$, $\mathscr{B}_1$ queries the challenger $\mathscr{C}_1$ to obtain a BE private key for i. The challenger $\mathscr{C}_1$ sends back sk_i^{BE} to $\mathscr{B}_1$ that gives it to the adversary. Moreover, $\mathscr{B}_2$ queries the challenger $\mathscr{C}_2$ to obtain a CBE private key for i. The challenger $\mathscr{C}_2$ sends back sk_i^{CBE} to $\mathscr{B}_2$ that gives it to the adversary. Finally, $\mathscr{A}$ sets $sk_i = (sk_i^{BE}, sk_i^{CBE})$.

When $\mathscr{A}$ asks for a CBBE certificate for $i \in [1,s]$, $j \in S_c \subseteq [1,k]$ and a time period l, $\mathscr{B}_2$ queries the challenger $\mathscr{C}_2$ to obtain the CBE certificate. $\mathscr{C}_2$ sends $Certif_{i,j,l}$ to $\mathscr{B}_2$ that forwards it to $\mathscr{A}$.

$\mathscr{A}$ also calls a decryption oracle by sending requests of the form (l,i,S_c,Hdr) for a time period l, a user i, $S_c \subseteq [1,k]$ and some header Hdr, which are answered using the relevant private key sk_i and the certificates $Certif_{i,j,l}$ for $j \in S_c$.

Once the query phase is over, $\mathscr{A}$ submits a challenge group $S_u^* \subseteq [1,s]$. $\mathscr{B}_1$ queries the challenger $\mathscr{C}_1$ for a challenge header and session key pair on S_u^*. The challenger responds by computing $(Hdr^{BE}, K_0) \leftarrow \mathsf{Encrypt}_{BE}(params, S_u^*)$ and by choosing K_1 at random from the key space. It then generates a bit $\mu \in_R \{0,1\}$ and gives (Hdr^{BE}, K_μ) to $\mathscr{B}_1$ who sends it to $\mathscr{A}$.

Thereafter, $\mathscr{A}$ splits Hdr^{BE} in $|S_c|$ parts as $Hdr^{BE,1}, Hdr^{BE,2}, \cdots, Hdr^{BE,|S_c|}$, and submits to $\mathscr{B}_2$. The latter queries the challenger $\mathscr{C}_2$ for a challenge ciphertext on $Hdr^{BE,1}, Hdr^{BE,2}, \cdots, Hdr^{BE,|S_c|}$. $\mathscr{C}_2$ replies by computing $C_{i,j} \leftarrow \mathsf{Encrypt}_{CBE}($ $params_j, pk_i, Hdr^{BE,j}, l, n)$ and setting $C_i = \{C_{i,j}\}_{j \in S_c}$, and sends back $Hdr = \{C_i\}_{i \in S_u^*}$ to $\mathscr{B}_2$ that gives it to $\mathscr{A}$.

Eventually, $\mathscr{A}$ outputs a bit μ' as a guess for μ, and wins if $\mu' = \mu$.

If $\mu = 0$, then $\mathscr{A}$'s view is as in $Game_0$. We notice that the private keys and the certificates from $\mathscr{B}_1$ and $\mathscr{B}_2$ are appropriately distributed, and the session key K_0 is correctly generated. If $\mu = 1$, then $\mathscr{A}$'s view is as in $Game_1$ since K_1 is chosen at random from the key space.

$Game_2$**.** $Game_2$ is identical to $Game_1$, except that the BE header Hdr_{BE} is chosen at random from the header space and splitted into $|S_c^*|$ parts $Hdr^{BE,j}$ for $j \in S_c^*$. Moreover, the CBBE header is set as follows: $Hdr = \{C_i\}_{i \in S_u^*}$ for a challenge group S_u^* chosen by the adversary $\mathscr{A}$ such that $C_{i,j} \leftarrow \mathsf{Encrypt}_{CBE}(params_j, pk_i, Hdr^{BE,j}, l, n)$. Therefore, $|Pr[E_2] - Pr[E_1]| = Adv_{CBE,\mathscr{B}_2}^{A-IND-CCA}(\lambda)$.

The challenger $\mathscr{C}_1$ sets the public parameters $params$ and the users' public keys pk_i as in the real BE scheme and forwards them to $\mathscr{B}_1$ who then sends them to $\mathscr{A}$. The challenger $\mathscr{C}_2$ sets the certifiers' public keys $params_j$ as in the real CBE scheme and forwards them to $\mathscr{B}_2$ who then sends them to $\mathscr{A}$.

When $\mathscr{A}$ asks for a certifier CBBE private key for $j \in [1,k]$, $\mathscr{B}_2$ queries the challenger $\mathscr{C}_2$ to obtain a CBE master secret key for j. The challenger $\mathscr{C}_2$ sends back msk_j to $\mathscr{B}_2$ that gives it to the adversary.

When $\mathscr{A}$ asks for a CBBE certificate for $i \in [1,s]$, $j \in S_c \subseteq [1,k]$ and a time period l, $\mathscr{B}_2$ queries the challenger $\mathscr{C}_2$ to obtain the CBE certificate. $\mathscr{C}_2$ sends $Certif_{i,j,l}$ to $\mathscr{B}_2$ that forwards it to $\mathscr{A}$.

$\mathscr{A}$ also calls a decryption oracle by sending requests of the form(l,i,S_c,Hdr) for a time period l, a user i, $S_c \subseteq [1,k]$ and some header Hdr, which are answered using the relevant private key sk_i and the certificates $Certif_{i,j,l}$ for $j \in S_c$.

Once the query phase is over, $\mathscr{A}$ submits two challenge sets $S_u^* \subseteq [1,s]$ and $S_c^* \subseteq [1,k]$. $\mathscr{B}_1$ asks the challenger $\mathscr{C}_1$ for the challenge pair of header and session key by submitting S_u^*. $\mathscr{C}_1$ answers by computing $(Hdr^{BE}, K) \leftarrow \mathsf{Encrypt}_{BE}(params, S_u^*)$, by setting $Hdr_0^{BE} = Hdr^{BE}$ and by choosing at random Hdr_1^{BE} from the header space. $\mathscr{C}_1$ sends back $(Hdr_0^{BE}, Hdr_1^{BE}, K)$ to $\mathscr{B}_1$.

Then, $\mathscr{B}_2$ is challenged on Hdr_0^{BE} and Hdr_1^{BE} that are forwarded to $\mathscr{C}_2$. The latter responds by flipping a bit $\mu \in_R \{0,1\}$, by dividing Hdr_μ^{BE} into $|S_c^*|$ parts and by sending back to $\mathscr{B}_2$ the components $(C_{i,j})_\mu \leftarrow \mathsf{Encrypt}_{CBE}(params_j, pk_i, Hdr_\mu^{BE,j}, l, n)$ to $\mathscr{B}_2$ for $j \in S_c^*$. Then, $\mathscr{B}_2$ sets $(C_i)_\mu = \{(C_{i,j})_\mu\}_{j \in S_c}$ and gives $(Hdr_\mu = \{(C_i)_\mu\}_{i \in S_u^*}, K)$ to $\mathscr{A}$.

Eventually, $\mathscr{A}$ outputs a bit μ' as a guess for μ.

If $\mu = 0$, $\mathscr{A}$'s view is as in $Game_1$. We notice that the private keys and the certificates are appropriately distributed, and the header Hdr_0^{BE} is correctly generated. If $\mu = 1$, $\mathscr{A}$'s view is as in $Game_2$ since the header Hdr_1^{BE} is a random element from the header space.

The theorem follows since in $Game_2$, both the header and the session key are independent elements from the bit μ, meaning that $|Pr[E_2] - \frac{1}{2}| = 0$.

4.2.6 Efficient Construction: CBBE_{effic}

The following CBBE construction CBBE_{effic} is an effective combination of the Boneh et al.'s (BGW) BE scheme [42] and the Gentry's CBE scheme [93]. Notice that the Gentry's CBE scheme is designed for a communication between one sender and one receiver. Therefore, applying this scheme directly to the Boneh et al.'s scheme will lead to headers with linear size in both the number of users and the number of certifiers, which is impractical. However, we manage to overcome this issue and achieve constant size for headers, private keys and certificates. Moreover, the efficient CBBE scheme CBBE_{effic} is proved selective IND-CCA secure in the random oracle model using the standard transformation from the REACT scheme proposed by Okamoto and Pointcheval [174], as well as selectively collusion resistant.

- $\mathsf{Setup}(\lambda, n, k) \rightarrow (params, \{(pk_i, sk_i)\}_{i \in [1,s]}, \{(pk_{c_j}, sk_{c_j})\}_{j \in [1,k]})$. First, run $(p, \mathbb{G}_1, \mathbb{G}_T, e, g) \leftarrow \mathscr{G}(\lambda)$. Pick at random $a \in_R \mathbb{Z}_p$ and compute g^{a^i} for $i = [1,s] \cup [s+2, 2s]$. Pick at random $\gamma \in_R \mathbb{Z}_p$ and compute $v = g^\gamma$. Choose three hash functions $H_1 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_1^2 \rightarrow \mathbb{G}_1$ and $H_3 : \mathbb{G}_T \times \mathbb{G}_1^4 \rightarrow \{0,1\}^\lambda$.

  For a user $i \in [1,s]$, compute the user's private key sk_i as $d_i = (g^{a^i})^\gamma = v^{a^i}$. Let the user i's public key pk_i be g^{a^i} for $i = [1,s]$.

  Independently, for $j \in [1,k]$, a certifier c_j computes its own public and private key pair as follows. First, choose at random an exponent $\sigma_j \in_R \mathbb{Z}_p$ and then compute the public key pk_{c_j} as $w_j = g^{\sigma_j}$. Set the private key sk_{c_j} as $d_{c_j} = \sigma_j$.

  Set the public parameters $params$ as $(p, \mathbb{G}_1, \mathbb{G}_T, e, g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, v, w_1, \cdots, w_k, H_1, H_2, H_3)$.

- $\mathsf{Certif}(params, S_c, \{(pk_{c_j}, sk_{c_j})\}_{j \in S_c}, pk_i, l) \rightarrow \{Certif_{i,j,l}\}_{j \in S_c}$. Let the time period l be represented as a string in $\{0,1\}^*$. Pick at random $r_{i,j,l} \in_R \mathbb{Z}_p$, for $j \in S_c \subseteq [1,k]$, and compute the user i's certificate $Cert_{i,j,l} = e_{i,j,l} = (e_{i,j,l,1}, e_{i,j,l,2})$ as follows:

$$\begin{aligned} e_{i,j,l,1} &= (g^{a^i})^{\sigma_j} \cdot H_1(w_j, l)^{\sum_{j \in S_c} r_{i,j,l}} = w_j^{a^i} \cdot H_1(w_j, l)^{\sum_{j \in S_c} r_{i,j,l}} \\ e_{i,j,l,2} &= g^{r_{i,j,l}} \end{aligned}$$

- $\mathsf{Encrypt}(params, S_u, S_c, \{pk_i\}_{i \in S_u}, \{pk_{c_j}\}_{j \in S_c}, l) \rightarrow (Hdr, K)$. First, pick at random an exponent $t \in_R \mathbb{Z}_p$, compute the session key $K = e(g^{a^s}, g^a)^t$ and set the header $Hdr = (C_1, C_2, C_3, C_4, C_5)$ as follows:

$$\begin{aligned} C_1 &= g^t \\ C_2 &= \prod_{j \in S_c} H_1(w_j, l)^t \\ C_3 &= (v \cdot \prod_{j \in S_c} w_j \cdot \prod_{i' \in S_u} g^{a^{s+1-i'}})^t \\ C_4 &= H_2(C_1, C_3)^t \\ C_5 &= H_3(K, C_1, C_2, C_3, C_4) \end{aligned}$$

- $\mathsf{Decrypt}(params, S_u, S_c, l, (pk_i, sk_i), \{Certif_{i,j,l}\}_{j \in S_c}, Hdr) \rightarrow K/\perp$. Let a user $i \in S_u$ have a private key $sk_i = d_i$ and the certificates $Cert_{i,j,l} = e_{i,j,l} = (e_{i,j,l,1}, e_{i,j,l,2})$ for $j \in S_c$. Let a header Hdr be parsed as $(C_1, C_2, C_3, C_4, C_5)$. First, check whether $e(C_1, H_2(C_1, C_3)) \stackrel{?}{=} e(g, C_4)$, and output

$$K = \frac{e(g^{a^i}, C_3) \cdot e(\prod_{j \in S_c} e_{i,j,l,2}, C_2)}{e(d_i \cdot \prod_{j \in S_c} e_{i,j,l,1} \cdot \prod_{i' \in S_u \setminus \{i\}} g^{a^{s+1-i'+i}}, C_1)} = e(g^{a^s}, g^a)^t.$$

Then, compute $C_5' = H_3(K, C_1, C_2, C_3, C_4)$. If $C_5' = C_5$, then return K; otherwise, return $\perp$.

Correctness. Notice that $(g^{a^i})^{a^{i'}} = g^{a^{i+i'}}$ for any i, i'. Given a time period l, a user $i \in S_u \subseteq [1, s]$ with a private key $sk_i = d_i$ and certificates $Cert_{i,j,l} = e_{i,l,j}$ for $j \in S_c \subseteq [1, k]$, retrieve K as follows:

$$\begin{aligned} K &= \frac{e(g^{a^i}, C_3) \cdot e(\prod_{j \in S_c} e_{i,j,l,2}, C_2)}{e(d_i \cdot \prod_{j \in S_c} e_{i,j,l,1} \cdot \prod_{i' \in S_u \setminus \{i\}} g^{a^{s+1-i'+i}}, C_1)} \\ &= \frac{e(g^{a^i}, (v \cdot \prod_{j \in S_c} w_j \cdot \prod_{i' \in S_u} g^{a^{s+1-i'}})^t) \cdot e(\prod_{j \in S_c} g^{r_{i,j,l}}, \prod_{j \in S_c} H_1(w_j, l)^t)}{e(v^{a^i} \cdot \prod_{j \in S_c} w_j^{a^i} \cdot H_1(w_j, l)^{\sum_{j \in S_c} r_{i,j,l}} \cdot \prod_{i' \in S_u \setminus \{i\}} g^{a^{s+1-i'+i}}, g^t)} \\ &= \frac{e(g^{a^i}, (v \cdot \prod_{j \in S_c} w_j \cdot \prod_{i' \in S_u \setminus \{i\}} g^{a^{s+1-i'}})^t) \cdot e(\prod_{j \in S_c} g^{r_{i,j,l}}, \prod_{j \in S_c} H_1(w_j, l)^t)}{e(v^{a^i} \cdot \prod_{j \in S_c} w_j^{a^i} \cdot H_1(w_j, l)^{\sum_{j \in S_c} r_{i,j,l}} \cdot \prod_{i' \in S_u \setminus \{i\}} g^{a^{s+1-i'+i}}, g^t)} \\ &\quad \cdot e(g^{a^i}, (g^{a^{s+1-i}})^t) \\ &= \frac{e(g, (v^{a^i} \cdot \prod_{j \in S_c} w_j^{a^i} \cdot \prod_{i' \in S_u \setminus \{i\}} g^{a^{s+1-i'+i}})^t)}{e(v^{a^i} \cdot \prod_{j \in S_c} w_j^{a^i} \cdot \prod_{i' \in S_u \setminus \{i\}} g^{a^{s+1-i'+i}}, g^t)} \\ &\quad \cdot \frac{e(g^{\sum_{j \in S_c} r_{i,j,l}}, \prod_{j \in S_c} H_1(w_j, l)^t)}{e(\prod_{j \in S_c} H_1(w_j, l)^{\sum_{j \in S_c} r_{i,j,l}}, g^t)} \cdot e(g^{a^{s+1}}, g)^t \\ &= e(g^{a^{s+1}}, g)^t = e(g^{a^s}, g^a)^t \end{aligned}$$

N.B. In the above construction, when the algorithm Certif is run, the certifiers c_j for indices $j \in S_c \subseteq [1, k]$ have to generate the certificates together for a given user $i \in [1, s]$.

More precisely, for $j \in S_c$, each certifer c_j picks at random an exponent $r_{i,j,l} \in_R \mathbb{Z}_p$ and shares it with the other selected certifiers. We assume that this sharing is done securely and efficiently.

4.2.7 Security Proofs for CBBE_{effic}

Selective IND-CCA Security Proof

Theorem. Suppose that the s-DBDHE problem and the k-DBDHE problem hold in $(\mathbb{G}_1, \mathbb{G}_T)$, then the efficient CBBE scheme CBBE_{effic} is selectivey IND-CCA secure in the random oracle model.

Game 1. We assume there exists an adversary $\mathscr{A}_1$ that breaks the selective IND-CCA security against uncertified users of the efficient CBBE scheme with advantage $Adv^{S-IND-CCA}_{\mathsf{CBBE}_{effic},\mathscr{A}_1}(\lambda)$ greater than ε_1 by making q_{H_1}, q_{H_2} and q_{H_3} random oracle queries, q_{cf} certification queries and q_{d_1} decryption queries. We construct a challenger $\mathscr{B}$ that can break the s-DBDHE problem in $(\mathbb{G}_1, \mathbb{G}_T)$ by interacting with the adversary $\mathscr{A}_1$ as follows.

$\mathscr{B}$ first receives the output $(p, \mathbb{G}_1, \mathbb{G}_T, e) \leftarrow \mathscr{G}(\lambda)$, a generator g of $\mathbb{G}_1$ and a problem intance $(g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, h = g^b)$ and Z that is either equal to $e(g^{a^{s+1}}, h)$ or to a random element in $\mathbb{G}_T$. $\mathscr{B}$ makes use of the three random oracles H_1, H_2 and H_3 and holds three corresponding hash lists L_1, L_2 and L_3, initially set as empty, to store all the query-answer pairs.

Initialization. $\mathscr{A}_1$ outputs a group $S_u^* \subseteq [1, s]$ of users that it wishes to be challenged on.

Setup. $\mathscr{B}$ generates the public parameters *params*, the private keys $sk_i = d_i$ for users $i \notin S_u^*$ and the private keys $sk_{c_j} = d_{c_j}$ for certifiers c_j such that $j \in [1, k]$ as follows. It chooses random elements $x, y_1 \cdots, y_k \in_R \mathbb{Z}_p$ and sets $v = g^x / \prod_{i' \in S_u^*} g^{a^{s+1-i'}}$, $w_j = g^{y_j}$ and $d_{c_j} = y_j$ for $j \in [1, k]$. It then computes

$$d_i = (g^{a^i})^x / \prod_{i' \in S_u^*} g^{a^{s+1-i'+i}} = g^{x \cdot a^i} \cdot (\prod_{i' \in S_u^*} g^{a^{s+1-i'}})^{-a^i} = v^{a^i}.$$

Eventually, $\mathscr{B}$ gives $\mathscr{A}_1$ the public parameters $params = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, v, w_1, \cdots, w_k, H_1, H_2, H_3)$, where H_1, H_2 and H_3 are controlled by $\mathscr{B}$ as follows:

- Upon receiving a query (w_j, l_j) to the random oracle H_1 for some $j \in [1, q_{H_1}]$:
  - If $((w_j, l_j), u_j, U_j)$ exists in L_1, then return U_j.
  - Otherwise, choose $u_j \in_R \mathbb{Z}_p$ at random and compute $U_j = g^{u_j}$. Put $((w_j, l_j), u_j, U_j)$ in L_1 and return U_j as answer.
- Upon receiving a query (W_{1j}, W_{2j}) to the random oracle H_2 for some $j \in [1, q_{H_2}]$:
  - If $((W_{1j}, W_{2j}), X_j)$ exists in L_2, then return X_j.
  - Otherwise, choose $X_j \in_R \mathbb{G}_1$ at random. Put $((W_{1j}, W_{2j}), X_j)$ in L_2 and return X_j as answer.

- Upon receiving a query $(K_j, C_{1j}, C_{2j}, C_{3j}, C_{4j})$ to the random oracle H_3 for some $j \in [1, q_{H_3}]$:
  - If $((K_j, C_{1j}, C_{2j}, C_{3j}, C_{4j}), C_{5j})$ exists in L_3, then return C_{5j}.
  - Otherwise, choose $C_{5j} \in_R \{0,1\}^\lambda$ at random. Put $(K_j, C_{1j}, C_{2j}, C_{3j}, C_{4j})$, $C_{5j})$ in L_3 and return C_{5j} as answer.

Query Phase 1. $\mathscr{B}$ answers $\mathscr{A}_1$'s queries as follows.

- *Certification Query* $< l_{i'}, i, S_c >$. Given $l_{i'}$ for $i' \in [1, q_{cf}]$, if $((w_{i',j}, l_{i'}), u_{i',j}, U_{i',j})$ exists in L_1, then return the pair $(u_{i',j}, U_{i',j})$ for $j \in [1,k]$. Otherwise, choose $u_{i',j} \in_R \mathbb{Z}_p$ at random, compute $U_{i',j} = g^{u_{i',j}}$ and put $((w_{i',j}, l_{i'})$, $u_{i',j}, U_{i',j})$ in L_1.

  Then, given $i \in [1,s]$ and c_j such that $j \in S_c \subseteq [1,k]$, pick at random $r_{i,j,l_{i'}} \in_R \mathbb{Z}_p$ and return the certificate $Cert_{i,j,l_{i'}} = e_{i,j,l_{i'}} = (e_{i,j,l_{i'},1}, e_{i,j,l_{i'},2})$ where

$$\begin{aligned} e_{i,j,l_{i'},1} &= (g^{a^i})^{y_j} \cdot U_{i',j}^{\sum_{j \in S_c} r_{i,j,l_{i'}}} = g^{y_j \cdot a^i} \cdot (g^{u_{i',j}})^{\sum_{j \in S_c} r_{i,j,l_{i'}}} \\ &= g^{y_j \cdot a^i} \cdot H_1(w_{i',j}, l_{i'})^{\sum_{j \in S_c} r_{i,j,l_{i'}}} = w_{i',j}^{a^i} \cdot g^{\sum_{j \in S_c} r_{i,j,l_{i'}} \cdot u_{i',j}} \\ e_{i,j,l_{i'},2} &= g^{r_{i,j,l_{i'}}} \end{aligned}$$

- *Decryption Query* $< l_{i'}, i, S_c, Hdr_{i'}, >$. Given $l_{i'}$ for $i' \in [1, q_{d_1}]$, $i \in [1,s]$, $S_c \subseteq [1,k]$ and a header $Hdr_{i'}$ parsed as $(C_{1i'}, C_{2i'}, C_{3i'}, C_{4i'}, C_{5i'})$,
  - if $((K_{i'}, C_{1i'}, C_{2i'}, C_{3i'}, C_{4i'}), C_{5i'})$ exists in L_3, then first compute $H_1(w_{i',j}, l_{i'})$ using the simulation of H_1 as above and check whether $e(C_{2i'}, g) \overset{?}{=} e(C_{1i'}, \prod_{j \in S_c} H_1(w_{i',j}, l_{i'}))$ for $j \in S_c \subseteq [1,k]$. If not, return $\perp$. Otherwise, compute $H_2(C_{1i'}, C_{3i'})$ using the simulation of H_2 as above and check whether $e(C_{1i'}, H_2(C_{1i'}, C_{3i'})) \overset{?}{=} e(g, C_{4i'})$. If not, return $\perp$. Otherwise, check whether
$$K_{i'} \overset{?}{=} \frac{e(g^{a^i}, C_{3i'}) \cdot e(g, C_{2i'})}{e((g^{a^i})^{x+\sum_{j \in S_c} y_j} \cdot \prod_{j \in S_c} H_1(w_{i',j}, l_{i'}), C_{1i'})}$$
    If the above equation holds, then return $K_{i'}$. Otherwise, return $\perp$;
  - else, return $\perp$.

Challenge. $\mathscr{B}$ generates the challenge header and session key pair according to the challenge group S_u^* as follows. First, $\mathscr{B}$ sets $C_1^* = h$ and searchs in L_1 to get u that corresponds to (w_j, l) such that $j \in S_c \subseteq [1,k]$. Then, it computes $C_2^* = h^u$. It also computes $C_3^* = h^{x+\sum_{j \in S_c} y_j}$. Informally, we write $h = g^t$ for an unknown $t \in \mathbb{Z}_p$. Then, the challenger randomly chooses an exponent $z \in_R \mathbb{Z}_p$ and sets $C_4^* = h^z = H_2(C_1^*, C_3^*)^t$. Second, it randomly chooses a bit $\mu \in_R \{0,1\}$, sets $K_\mu = Z$ and picks at random $K_{1-\mu}$ in $\mathbb{G}_T$. It also picks $C_5^* \in_R \{0,1\}^\lambda$ at random and sets $C_5^* = H_3(K_\mu, C_1^*, C_2^*, C_3^*, C_4^*)$. Finally, it returns $(Hdr^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*), K_0, K_1)$ as the challenge to $\mathscr{A}_1$.

If $Z = e(g^{a^{s+1}}, h)$, then (Hdr^*, K_0, K_1) is a valid challenge header and session key pair for $\mathscr{A}_1$'s view as in the real attack. Indeed, if we write $h = g^t$ for an unknown

$t \in \mathbb{Z}_p$, then

$$\begin{aligned}
h^{x+\sum_{j\in S_c} y_j} &= h^x \cdot \prod_{j\in S_c} h^{y_j} \cdot \left(\frac{\prod_{i\in S_u^*} g^{a^{s+1-i}}}{\prod_{i\in S_u^*} g^{a^{s+1-i}}}\right)^t \\
&= g^{t\cdot x} \cdot \prod_{j\in S_c} g^{t\cdot y_j} \cdot \left(\frac{\prod_{i\in S_u^*} g^{a^{s+1-i}}}{\prod_{j\in S_u^*} g^{a^{s+1-i}}}\right)^t \\
&= \left(\frac{g^x}{\prod_{i\in S_u^*} g^{a^{s+1-i}}}\right)^t \cdot \prod_{j\in S_c} g^{t\cdot y_j} \cdot (\prod_{i\in S_u^*} g^{a^{s+1-i}})^t \\
&= \left(\frac{g^x}{\prod_{i\in S_u^*} g^{a^{s+1-i}}} \cdot \prod_{j\in S_c} g^{y_j} \cdot \prod_{i\in S_u^*} g^{a^{s+1-i}}\right)^t \\
&= (v \cdot \prod_{j\in S_c} w_j \cdot \prod_{i\in S_u^*} g^{a^{s+1-i}})^t = C_3^*
\end{aligned}$$

Therefore, by definition, Hdr^* is a valid encryption of the session key $e(g^{a^{s+1}}, g)^t = e(g^{a^s}, g^a)^t$. Moreover, $e(g^{a^{s+1}}, g)^t = e(g^{a^{s+1}}, h) = Z = K_\mu$, and thus (Hdr^*, K_0, K_1) is a valid challenge to $\mathscr{A}_1$. If $Z \in_R \mathbb{G}_T$, then K_0 and K_1 are random independent elements in $\mathbb{G}_T$.

Query Phase 2. $\mathscr{B}$ responds to the certification and decryption queries as in the query phase 1. We note that if $(K_\mu, C_1^*, C_2^*, C_3^*, C_4^*)$ is asked to the random oracle H_3, then the value C_5^* created in the simulation of the challenge phase is returned.

Guess. The adversary $\mathscr{A}_1$ outputs its guess $\mu' \in \{0,1\}$ for μ. The adversary wins if $\mu' = \mu$.

Analysis. The simulations to generate the private keys and the certificates are perfect since the responses are correct regarding $\mathscr{A}_1$'s view.

The simulation of the random oracle H_1 is not entirely perfect. Let *QueryH*$_1$ be the event that $\mathscr{A}_1$ has queried (w_j^*, l^*) to H_1 for $j \in [1,k]$ before the challenge phase. This event happens with probability k/p. Except for the case above, the simulation of H_1 is perfect.

The simulations of the random oracles H_2 and H_3 are not entirely perfect. Let *QueryH*$_2$ and *QueryH*$_3$ be the events that $\mathscr{A}_1$ has queried (C_1^*, C_3^*) to H_2 and $(K_\mu, C_1^*, C_2^*, C_3^*, C_4^*)$ to H_3 before the challenge phase. These two events happen with probability $1/p$ and $1/2^\lambda$ respectively. Except for the cases above, the simulations of H_2 and H_3 are perfect.

The simulation of the decryption oracle is nearly perfect, except that a valid header can be rejected sometimes. Indeed, in the simulation of the decryption oracle, if (K, C_1, C_2, C_3, C_4) has not been queried to H_3, then the header is rejected. This leads to two cases:

1. $\mathscr{A}_1$ uses the value C_5^*, which is part of the challenge, as a part of its decryption query.

2. $\mathscr{A}_1$ has guessed a right value for the output of H_3 without querying it.

However, in the first case, since $(C_1^*, C_2^*, C_3^*, C_4^*)$ and K_μ are provided as input to H_3, the decryption query that $\mathscr{A}_1$ would ask is the same as the challenge, which is not allowed to query. The second case may happen but with negligible probability $1/2^{\lambda}$. Let *DecO* denote the event that $\mathscr{A}_1$ correctly guesses the output of H_3. Therefore, if $\mathscr{B}$ does not correctly guess the output of H_3, $\mathscr{A}_1$'s view in the simulation is identical to the one in the real attack.

Thus, we have

$$Adv_{\mathscr{B}}^{DBDHE} = |Pr[\mu' = \mu | \neg QueryH_1 \wedge \neg QueryH_2 \wedge \neg QueryH_3 \wedge \neg DecO] - \frac{1}{2}|.$$

By definition of $\mathscr{A}_1$'s advantage, we have $|Pr[\mu' = \mu] - \frac{1}{2}| > \varepsilon_1 - Pr[QueryH_1] - Pr[QueryH_2] - Pr[QueryH_3] - Pr[DecO]$, that gives us

$$\begin{aligned} & |Pr[\mu' = \mu | \neg QueryH_1 \wedge \neg QueryH_2 \wedge \neg QueryH_3 \wedge \neg DecO] - \frac{1}{2}| \\ > \; & |Pr[\mu' = \mu] - Pr[QueryH_1] - Pr[QueryH_2] - Pr[QueryH_3] - Pr[DecO] - \frac{1}{2}| \\ > \; & \varepsilon_1 - Pr[QueryH_1] - Pr[QueryH_2] - Pr[QueryH_3] - Pr[DecO] \end{aligned}$$

Since $\mathscr{A}_1$ makes q_{H_1}, q_{H_2} and q_{H_3} random oracle queries during the attack, therefore $Pr[QueryH_1] \leq k \cdot q_{H_1}/p$, $Pr[QueryH_2] \leq q_{H_2}/p$ and $Pr[QueryH_3] \leq q_{H_3}/2^{\lambda}$. In the same way, since $\mathscr{A}_1$ makes q_{d_1} decryption queries during the attack, we get that $Pr[DecO]$ $\leq q_{d_1}/2^{\lambda}$. Therefore, we obtain $\mathscr{B}$'s advantage $Adv_{\mathscr{B}}^{DBDHE} > \varepsilon_1 - \frac{k \cdot q_{H_1} + q_{H_2}}{p} - \frac{q_{d_1} + q_{H_3}}{2^{\lambda}}$.

Game 2. The security proof for Game 2 is rouglhy the same than the one above for Game 1, except that the adversary, playing the role of an untrusted certifier, cannot make queries to the certification oracle.

We assume there exists an adversary $\mathscr{A}_2$ that breaks the selective IND-CCA security against uncertified users of the efficient CBBE scheme with advantage $Adv_{\mathsf{CBBE}_{effic},\mathscr{A}_2}^{S-IND-CCA}(\lambda)$ greater than ε_2 by making q_{H_1}, q_{H_2} and q_{H_3} random oracle queries and q_{d_2} decryption queries. We construct a challenger $\mathscr{B}$ that can break the k-DBDHE problem in $(\mathbb{G}_1, \mathbb{G}_T)$ by interacting with the adversary $\mathscr{A}_2$.

Therefore, goind through the same analysis, we obtain $\mathscr{B}$'s advantage $Adv_{\mathscr{B}}^{DBDHE} > \varepsilon_2 - \frac{k \cdot q_{H_1} + q_{H_2}}{p} - \frac{q_{d_2} + q_{H_3}}{2^{\lambda}}$.

Selective Collusion Resistance Proof

Theorem. Suppose that the s-BDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$, then the efficient CBBE scheme CBBE_{effic} is selectively collusion resistant in the random oracle model.

We assume there exists an adversary $\mathscr{A}$ that breaks the selective collusion resistance of the CBBE scheme with advantage $Adv_{\mathsf{CBBE}_{effic},\mathscr{A}}^{S-CollRes}(\lambda)$. We construct a challenger $\mathscr{B}$ that interacts with $\mathscr{A}$ as follows.

Initialization. $\mathscr{A}$ chooses a subgroup $S_{u,s'} \subseteq [1,s]$ of colluding users such that $|S_{u,s'}| = s' \leq s$.

Setup and Query Phase. $\mathscr{B}$ runs Setup on input the tuple (λ, s, k) to obtain the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_T, e, g, g^{a}, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}}, v, w_1, \cdots, w_k, H_1, H_2,$

H_3), the private keys $sk_i = d_i = v^{a^i}$ for users $i \in S_{u,s'}$ and the public keys $pk_{c_j} = w_j = g^{\sigma_j}$ for certifiers c_j such that $j \in [1,k]$. It gives these elements to $\mathscr{A}$ and keeps the certifier c_j's private key $sk_{c_j} = d_{c_j} = \sigma_j$ for $j \in [1,k]$. It also runs Certif on input the tuple $(params, S_c, \{(pk_{c_j}, sk_{c_j})\}_{j \in S_c}, pk_i, l)$ to obtain the certificates $Cert_{i,j,l} = e_{i,j,l} = (e_{i,j,l,1} = w_j^{a^i} \cdot H_1(w_j, l)^{\sum_{j \in S_c} r_{i,j,l}},\ e_{i,j,l,2} = g^{r_{i,j,l}})$ for user $i \in S_{u,s'}$, index $j \in S_c \subseteq [1,k]$ and time period l. It gives these elements to $\mathscr{A}$.

Challenge. $\mathscr{B}$ chooses a group $S \subseteq [1,s]$ of users such that $S \cap S_{u,s'} = \emptyset$, a time period $l' \neq l$ and a group $S_c \subseteq [1,k]$. It runs $(Hdr, K) \leftarrow \mathsf{Encrypt}(params, S, S_c, l')$ and forwards $Hdr = (C_1 = g^t, C_2 = \prod_{j \in S_c} H_1(w_j, l')^t, C_3 = (v \cdot \prod_{j \in S_c} w_j \cdot \prod_{i' \in S} g^{a^{s+1-i'}})^t, C_4 = H_2(C_1, C_3)^t, C_5 = H_3(K, C_1, C_2, C_3, C_4))$ to $\mathscr{A}$ while keeping the session key $K = e(g^{a^s}, g^a)^t$.

Output. $\mathscr{A}$ computes a new private key $sk_\chi = d_\chi$ and a new certificate $Cert_{\chi,j,l} = e_{\chi,j,l}$ from the private keys and the certificates of users in $S_{u,s'}$ that it has previously obtained. More precisely, $\mathscr{A}$ defines a group $S_{u,\bar{s}} \subseteq S_{u,s'}$ and sets

$$\begin{aligned}
d_\chi &= \prod_{i' \in S_{u,\bar{s}}} d_{i'} = v^{\sum_{i' \in S_{u,\bar{s}}} a^{i'}} = v^{f_\chi(a)} \\
e_{\chi,j,l,1} &= \prod_{i' \in S_{u,\bar{s}}} e_{i',j,l,1} = w_j^{\sum_{i' \in S_{u,\bar{s}}} a^{i'}} \cdot H_1(w_j, l)^{\sum_{j \in S_c} \sum_{i' \in S_{u,\bar{s}}} r_{i',j,l}} \\
&= w_j^{f_\chi(a)} \cdot H_1(w_j, l)^{\sum_{j \in S_c} r_{\chi,j,l}} \\
e_{\chi,j,l,2} &= g^{\sum_{i' \in S_{u,\bar{s}}} r_{i',j,l}} = g^{r_{\chi,j,l}}
\end{aligned}$$

where $f_\chi(a) = \sum_{i' \in S_{u,\bar{s}}} a^{i'}$.

Analysis. For a user i belonging to S, some terms cancel out as follows:

$$\begin{aligned}
&\frac{e(g^{a^i}, (v \cdot \prod_{j \in S_c} w_j \cdot \prod_{i' \in S \setminus \{i\}} g^{a^{s+1-i'}})^t)}{e(v^{f_\chi(a)} \cdot \prod_{j \in S_c} w_j^{f_\chi(a)} \cdot \prod_{i' \in S \setminus \{i\}} g^{a^{s+1-i'+i}}, g^t)} \cdot \frac{e(\prod_{j \in S_c} g^{r_{\chi,j,l}}, \prod_{j \in S_c} H_1(w_j, l')^t)}{e(\prod_{j \in S_c} H_1(w_j, l)^{\sum_{j \in S_c} r_{\chi,j,l}}, g^t)} \\
= \quad & \frac{e(g, v^{a^i} \cdot \prod_{j \in S_c} w_j^{a^i})}{e(v^{f_\chi(a)} \cdot \prod_{j \in S_c} w_j^{f_\chi(a)}, g)} \cdot \frac{e(g, \prod_{j \in S_c} H_1(w_j, l'))}{e(\prod_{j \in S_c} H_1(w_j, l), g)}
\end{aligned}$$

The ratio $\frac{e(g, v^{a^i} \cdot \prod_{j \in S_c} w_j^{a^i})}{e(v^{f_\chi(a)} \cdot \prod_{j \in S_c} w_j^{f_\chi(a)}, g)}$ should be equal to 1, meaning that the two terms $v^{a^i - f_\chi(a)}$ and $w_j^{a^i - f_\chi(a)}$ should be wiped out. We should get that $f_\chi(a) = \sum_{i' \in S_{u,\bar{s}}} a^{i'} = a^i \bmod p$. In other words, a is a root of the polynomial $P(x) = \sum_{i' \in S_{u,\bar{s}}} x^{i'} - x^i =$ of degree $\bar{s} + 1$. This happens with probability $Pr[f_\chi(a) = a^i \bmod p] \leq (\bar{s} + 1)/p$.

We now consider the ratio $\frac{e(g, \prod_{j \in S_c} H_1(w_j, l'))}{e(\prod_{j \in S_c} H_1(w_j, l), g)}$. Let us suppose that $l \neq l'$ and that the hash function H_1 is a random oracle. Therefore, the probability that the two outputs are equal is equal to $Pr[H_1(w_{j'}, l') = H_1(w_j, l)] \leq 1/p$, given two indices $j, j' \in [1,k]$. Finally, $\mathscr{A}$ has negligible advantage $Adv^{S-CollRes}_{\mathsf{CBBE}_{effic}, \mathscr{A}}(\lambda) \leq (\bar{s} + 2)/p$ to retrieve the session key K.

4.2.8 Discussion

We constructed our efficient CBBE scheme based on Boneh et al.'s BE [42] and Gentry's CBE [93] schemes since:

- the Boneh et al.'s BE scheme is fully collusion resistant and has short ciphertexts and private keys (their sizes are constant in the number of users),
- the Gentry's CBE scheme achieves adaptive IND-CCA security in the random oracle model.

In Table 4.3, we compare the sizes of the public parameters, the private keys and the headers in the basic and efficient schemes respectively. Let n be the number of users in the system and S_u be the group of users selected by the broadcaster. Let k be the number of certifers in the system and S_c be the group of certifers selected by the broadcaster. The public components PC comprise the public parameters $params$, the public keys pk_i of the users and the public keys pk_{c_j} of the certifiers. The private keys SK refer to sk_i, sk_{c_j}, the certificates $Cert$ refer to $Cert_{i,j,l}$ and the header Hdr simply refers to Hdr in the construction.

	CBBE_{basic}	CBBE_{effic}
Size of PC	$O(k \cdot s)$	$O(k \cdot s)$
Size of SK	$O(1)$	$O(1)$
Size of $Cert$	$O(1)$	$O(1)$
Size of Hdr	$O(\|S_c\| \times \|S_u\|) = O(k \cdot s)$	$O(1)$
Security Model	adaptive IND-CCA in the standard model	selective IND-CCA in the random oracle model

Table 4.3: Comparative table between the basic scheme CBBE_{basic} and the efficient scheme CBBE_{effic}. Let PC denote the public components (including the public parameters and the public keys), SK denote the private keys of both the users and the certifiers, $Cert$ denote the certificates and Hdr denote the headers. We consider the size of one private key for one user or one certifier, the size of one certificate for one user given one certifier, and the size of one header given one message and two groups of users and certifiers respectively.

The basic construction CBBE_{basic} is a simple combination of BE and CBE systems such that the header Hdr has size linear in the number of users in group S_u and the number of certifers in group S_c, which is impractical. We notice that the efficient construction CBBE_{effic} yields constant size for the private key, the certificate and the header, which is really practical.

If the BE and CBE schemes are assumed to be adaptively IND-CCA secure in the standard model, then the basic scheme CBBE_{basic} is adaptively CCA secure in the standard model. However, these hypotheses are quite strong compared to the existing schemes for BE and CBE systems. Gentry's CBE [93] achieves adaptive IND-CCA security in the random oracle model, and a practical BE with short ciphertexts and private keys, which is due to Boneh et al. [42], is selectively IND-CPA secure in the standard model. Based on these two schemes, our efficient CBBE scheme CBBE_{effic} reaches selective IND-CCA security in the random oracle. Although the adaptive IND-CCA security is the strongest existing model, we feel that selective security achieves a good level of security and can be more conceivable for better efficiency and performance in some cases.

4.2.9 Performance

In the following table, we evaluate the efficiency of our efficient CBBE symmetric pairing-based scheme. We use results of cryptographic operation implementations (group exponentiations of a random group element with a random exponent and pairing operations on random group elements) using the MIRACL framework [203, 196] for a 128-bit security level. All the following experiments are based on a dual core IntelR XeonR CPU W3503@2.40GHz with 2.0GB RAM running Ubuntu R10.04. The elliptic curve utilised for all the benchmarks was the super-singular symmetric curve $y^2 = x^3 + 1 \mod p$ with embedding degree 2 for suitable primes p.

We do not take into account multiplication/division in $\mathbb{G}_1$ and $\mathbb{G}_T$ as well as exponentiation in $\mathbb{G}_T$ as these operations have timings negligible compared to the ones for exponentiation in $\mathbb{G}_1$ and pairing operation.

	Group Exponentiation (in $\mathbb{G}_1$)	Pairing
Time/operation	34.4	176.6
Setup	11,008	
Certif	2,064	
Encrypt	137.6	176.6
Decrypt		706.4

Table 4.4: Timings for our efficient CBBE symmetric pairing-based system CBBE_{effic}. Times are in milliseconds. We assume that there are $s = 100$ users and $k = 20$ certifiers.

We note that the total time in the algorithm Setup is substantial, but we recall that this algorithm should be run only once to generate the public parameters and the static private keys for both users and certifiers. In the algorithm Certif, it requires 2,064 milliseconds since up to $k = 20$ certificates are created. Finally, in the algorithms Encrypt and Decrypt, it takes 324.2 milliseconds and 706.4 milliseconds respectively, mainly due to the cost of pairing computations: these results remain the same for every execution of the protocol, since the number of group exponentiation and pairing operation is constant.

4.2.10 Conclusion

We presented a CBBE primitive to answer the problematic arisen by the above scenario. We demonstrated that this primitive is very practical in enabling a secure communication among medical staff members in a hospital. We properly defined the scheme definition as well as the security requirements. We constructed a basic CBBE scheme that reaches adaptive IND-CCA security in the standard model but with components' size linear in the number of involved entities, and an efficient CBBE scheme with short ciphertexts, private keys and certificates, and proved it selectively IND-CCA secure and selectively collusion resistant in the random oracle model.

Chapter 5

Accessing and Retrieving Electronic Health Records Stored on Cloud Servers

Since EHRs are numerous and computationally heavy, the hospital and the patients cannot afford to keep them on their private personal storage devices. Thus, our solution is to store the EHRs on external storages such that hospital staff members and patients should be able to deal with them easily and efficiently. Nevertheless, we have to ensure that EHRs are securely and privately stored, as well as their access is restricted to authorised hospital staff members regarding rights and privileges carefully attributed beforehand. Moreover, we should not allow the hospital staff members to retrieve more documents than the ones they request. This means that another token should be brought by the members to complete their authorisation.

We propose to store the EHRs under their encrypted form, so that the cloud server almost knows nothing on the contents of the stored EHRs. By "almost", we mean that the cloud server might learn some pieces of information on the requested EHRs since we should guarantee that the hospital staff members precise their document search through a token. We also suggest to let the cloud server be the party who verifies the access rights and privileges, in order to avoid other computational and communication costs due to an external verifier.

5.1 Certificate-Based Encryption with Keyword Search

Authorising hospital staff members to access and retrieve medical documents brings security and privacy issues. In Figure 5.1, we highlight the general context as follows: each hospital staff member receives his/her access right from a medical institute (or any other health legislator, such as the government). The hospital staff member's access right can evoluate over the time with the help of the medical institute.

An access right allows a hospital staff member to access and retrieve medical information, such as EHRs, that are stored on a local cloud server maintained by the IT team of the hospital (that belongs to the administration staff). When a hospital staff member desires to collect some patients' medical documents, s/he sends a request containing a keyword, as a description of the medical documents that s/he is looking for, along with his/her access right component. The local cloud server then checks the elements sent by the hospital staff member, and if both the keyword and the acces right are accepted regarding some requirements, then the cloud server releases the requested medical documents to the hospital staff member.

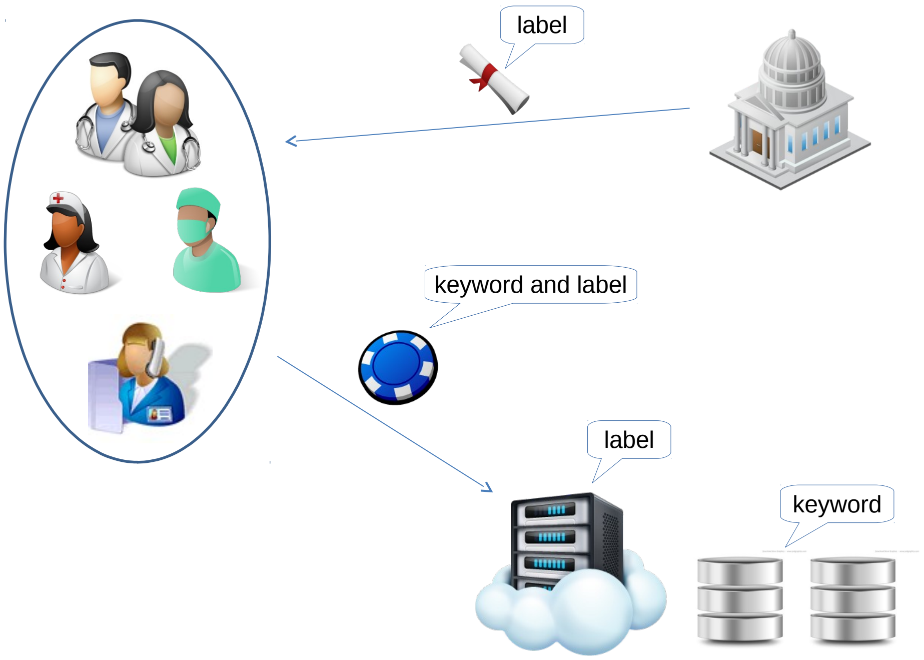


Figure 5.1: Certificate-Based Encryption with Keyword Search. A medical institute, acting as a certifier, first generates certificates for all the hospital staff members. It also computes update keys according to some groups of hospital staff members, such that only a hospital staff member in such groups can successfully refresh its certificate. These certificates contain labels as a description of the access rights given to the hospital staff members. Then, a hospital staff member wants to retrieve a document and sends a request to the cloud server under the form of a trapdoor that was generated regarding a keyword and his/her most refreshed certificate. The cloud server checks that the elements embedded in the trapdoor match the ones found in the requested document using the database.

We present three scenarios that describe the authorisation process in a hospital to enable its staff members to acquire access to patients' EHRs. Let us consider our first scenario as follows. A patient, Alice, visits the hospital for a gyneacological checkup. She requests that the checkup will be conducted by a female clinician. On her arrival, a nurse, Daisy, takes Alice's blood sample and records this information "securely" in Alice's EHR, which is stored on the local server of the hospital. The security of the EHR is done by encrypting this information prior to uploading it to the local server. Once this is done, Alice will need to wait until she is called by the available female gyneacologist as requested.

Let Fanny be a female gyneacologist in the hospital. She has to access Alice's EHR, and in particular the blood sample report, in order to discuss with the patient on some potential issues. To do so, Fanny sends a request to the server that contains a descriptive keyword for the record that she needs to access (e.g. "Alice") and some information about her access right (e.g. she is a gyneacologist in the hospital). The information will only be delivered (or decryptable) by Fanny if her access right is valid and the documents that she is looking for (i.e. Alice's EHR) is available on the local server. This scenario is depicted in Figure 5.2.

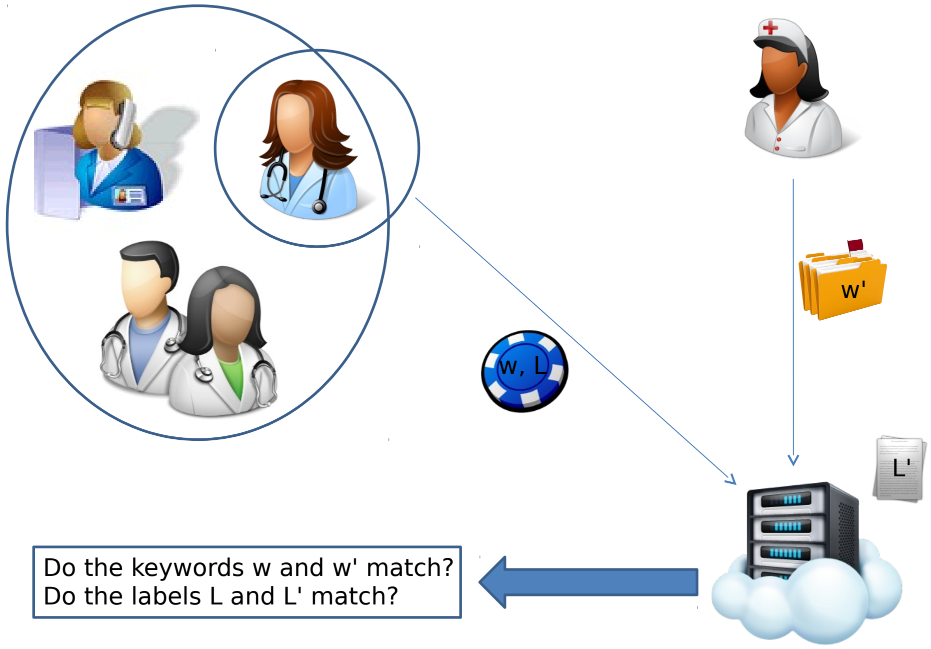

w'
w, L
L'
Do the keywords w and w' match?
Do the labels L and L' match?

Figure 5.2: Daisy uploads Alice's EHR on the local cloud server of the hospital. The records contain an encryption of the keyword w' that she has chosen to describe them. Fanny sends a file encrypting her keyword w and her label L (corresponding to her access right) as a request to retrieve Alice's records. The server holds a label L' according to the hospital regulation. If the keywords and the labels respectively match, then the server released the requested records to Fanny.

Our second scenario takes place in pediatrics and kids hospital. A child, Charles, has contracted chickenpox and his parents decide to bring him to the hospital to check his condition. Since this disease is also contagious to adults and other children, Charles has to been examined by a pediatrician who is immune against chickenpox. Thus, this requirement has to be added to the clinician's access right to let this clinician be able to retrieve Charles's EHR. At the end of the diagnosis, the clinician will also need to store the result of the examination in Charles's EHR. This scenario is depicted in Figure 5.3.

Our last scenario is due to the communciation difficulty. Eva, who is a Russian tourist and currently visiting her 80-year-old uncle in Australia, visits the emergency department due to her broken ankle. Unfortunately, Eva does not speak English but Russian, and she requires a clinician that speaks her language. Fortunately, an orthopedist, Frank, satisfies this requirement. By providing his access right that includes this linguistic feature and a descriptive keyword, Frank will be able to retrieve Eva's EHR. This scenario is depicted in Figure 5.4.

These scenarios illustrate a required framework to authorise hospital staff members to access and retrieve the necessary medical documents to conduct their job. As illustrated earlier, there are two essential information pieces required, namely the *access right* of a hospital staff member that has to be in accordance with the hospital requirements (regarding the patients' specific requirements for instance) and a *keyword* selected by the hospital staff member as the description of the requested medical documents.

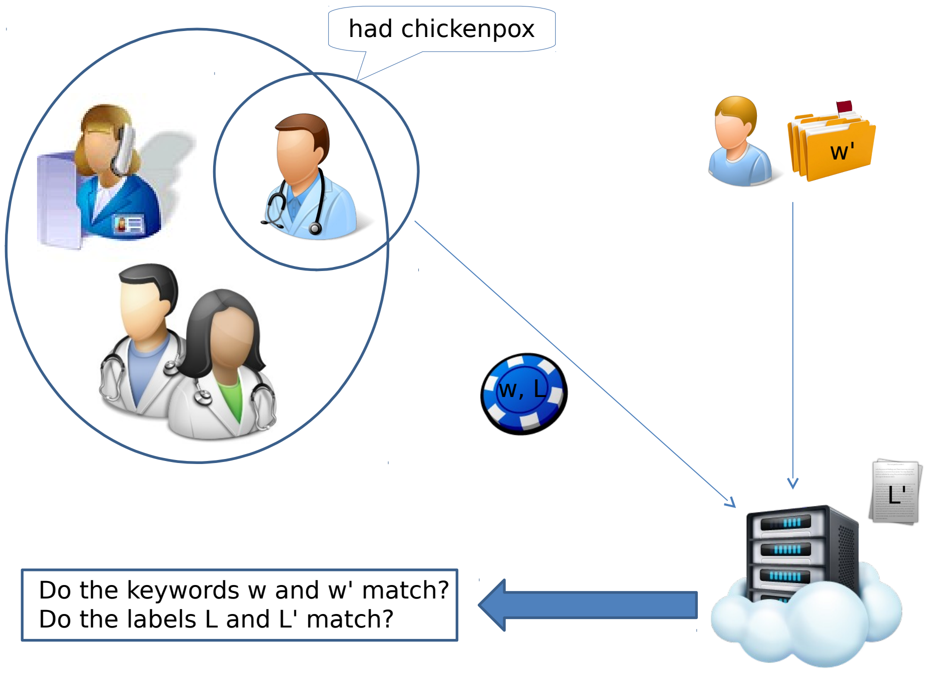

had chickenpox
w'
w, L
L'
Do the keywords w and w' match?
Do the labels L and L' match?

Figure 5.3: A sick child, Charles, is associated with an EHR that can be found in the hospital's database. A pediatrician is chosen among other clinicians because of his position (i.e. he has a medical degree related to kids) as well as the fact that he has already contracted the chickenpox.

5.1.1 Contributions

In order to capture the above scenarios, we consider four entities:

1. An *uploader* that transfers the contents of medical information (e.g. EHRs) to a server;

2. A *server* that stores the uploader's encrypted contents of medical information and delivers these contents to the receivers if and only if certain conditions are met;

3. Several *receivers*, that ask the server to retrieve some contents of the medical information;

4. A *certifier* that delivers certificates to the receivers according to their access right status.

In practice, the receivers are the hospital staff members, while the uploader can be anyone, the certifier is the IT team of the hospital (that belongs to the administration staff) and the server is the hospital database linked to a private cloud server. The assumption is that we do not want the administrator of the server to be able to read the medical documents without proper consents, and therefore the medical information that will be encrypted and stored. Hence, we adopt the honest-but-curious model for the server.

We suppose that Daisy, a nurse working at the hospital, needs to access some medical documents. To do so, Daisy chooses a keyword w^R describing the documents that she

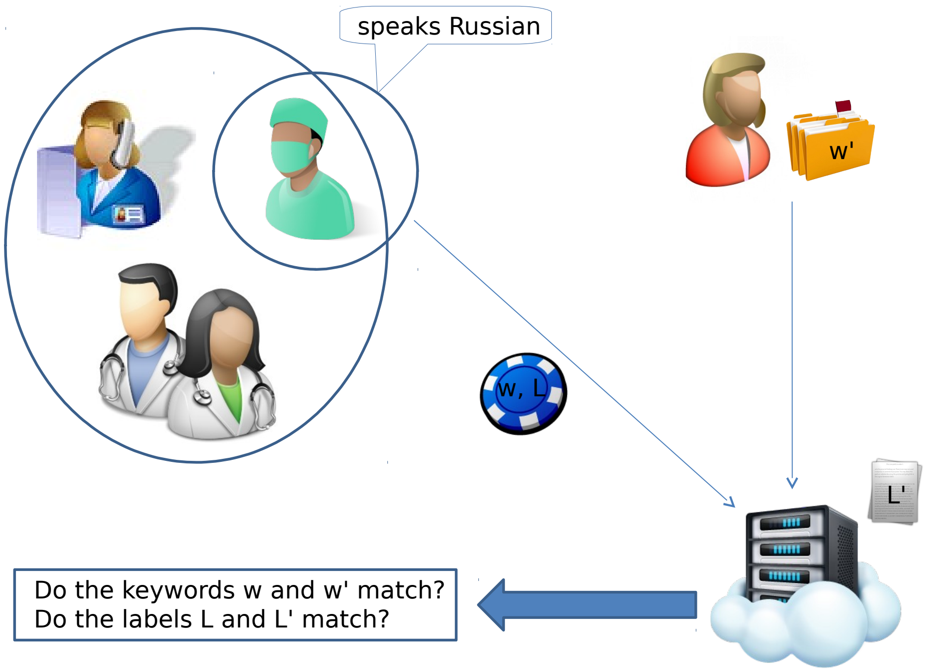

speaks Russian
w'
w, L
L'
Do the keywords w and w' match?
Do the labels L and L' match?

Figure 5.4: A Russian tourist, Eva, is associated with an EHR that can be found in the hospital's database. A clinician, Frank, is able to speak Russian, and thus this feature permits him to be selected among other medical staff members to auscultate Eva.

wants to retrieve, creates a trapdoor that is a function of the keyword w^R and a certificate, and sends this trapdoor to the server as an access request.

Daisy is equipped with a valid certificate in order to conduct this operation. Daisy's certificate is updated regularly from the broadcasted information provided by the certifier. The update key is created given a group S of hospital staff members and a label L^R. Note that the update key is broadcasted to all the people working in the hospital but only people belonging to the group S can successfully update their certificates. In addition, the label L^R refers to information such as access rights and other privileges and features, and enables the hospital staff members in S to access and retrieve some documents. Note that the certificate needs to be refreshed since the label L^R and/or the group S might change over the time.

When Daisy sends the trapdoor to the server, the latter checks it with a ciphertext encrypting a keyword w^S that describes the requested documents (the keyword w^S was chosen by the person who encrypted the documents) and the current label L^S (the label L^S is determined by the hospital). The server is actually verifying whether the keywords and the labels match respectively (i.e. whether $w^R = w^S$ and $L^R = L^S$). If so, then the server releases the encrypted documents to Daisy; otherwise, the server keeps them on its local storage.

We illustrate our certificate-based encryption with keyword search (CBEKS) protocol in Figure 5.5 such that the four users are depicted. First, the certifier generates the first certificate given to each receiver as an encryption of a label L_1^R. It also computes update keys that are sent to all the receivers. Receivers can successfully refresh their certificates using the update keys if and only if they belong to a group S_j specified by the certifier. Then, the uploader computes a ciphertext C_{w^S} for a keyword w^S and transmits it to the server. Each receiver computes a trapdoor $Trap_{w_i^R,L_j^R}$ according to a keyword w_i^R and the most recent certificate encrypting the label L_j^R; the receiver then forwards it to the server. At last, the server checks whether the two keywords match and whether the label embedded into the trapdoor is equal to the current label L_l^S, i.e. $w_i^R = w^S$ and $L_j^R = L_l^S$.

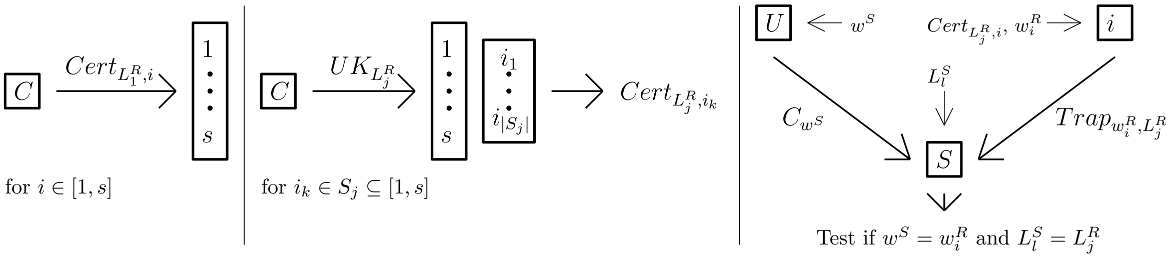


Figure 5.5: A certifier C generates a first certificate $Cert_{L_1^R,i}$ for a label L_1^R, and gives it to a receiver i, where $i \in [1,s]$. Thereafter, C computes an update key $UK_{L_j^R}$ for L_j^R and a group $S_j \subseteq [1,s]$ of receivers, and gives it to i. Then, i uses $UK_{L_j^R}$ to obtain a refreshed certificate $Cert_{L_j^R,i}$ if and only if $i \in S_j$. Afterwards, the uploader U generates a ciphertext C_{w^S} for a keyword w^S, and transfers it to a server S. With inputs w_i^R and $Cert_{L_j^R,i}$ for L_j^R, i computes a trapdoor $Trap_{w_i^R,L_j^R}$ and forwards it to S. Finally, given L_l^S, C_{w^S} and $Trap_{w_i^R,L_j^R}$, S tests whether $w^S = w_i^R$ and $L_l^S = L_j^R$.

Attribute-based encryption (ABE) [104] is a cryptographic primitive that involves attributes to generate private keys and ciphertexts. Attributes can be seen as the components describing the access right of a receiver. Observe that no certificate is delivered and no key update is possible in an ABE system. Instead, private keys are created with respect to each attribute taken individually and have to be re-generated from scratch each time that an attribute is added, deleted or modified. We note that treating the attributes individually each time is cumbersome and inefficient. Moreover, an ABE system is not equipped with searching capability: no keyword or other requirement is necessary to complete the authorisation step. Thus, such primitive does not seem suitable to satisfy the medical criteria cited above.

In Section 4.2, we proposed a certificate-based broadcast encryption (CBBE) protocol. This protocol enables medical staff members (receivers) to communicate among themselves, as well as to retrieve medical documents such that authorisation is given through licences (i.e. certificates) that allow the medical staff to practice in the given hospital. We observe that a CBBE system does not satisfy the medical scenario described in this section. First, certificates cannot be collectively updated, but have to be individually re-generated in case of access right changes. Moreover, a receiver does not search for medical documents, but acquires access to all of them after successful authorisation (even the ones that this receiver is not interested to acquire).

In this section, we address the problem of authorising receivers in a sensitive environment to let them access and retrieve medical documents securely. In order to access medical information encrypted by an uploader and stored on a server, a receiver has to provide a trapdoor that embeds two elements: a keyword that describes the targeted information and a certificate that includes a label as the access right of the receiver. The receiver will retrieve the encrypted information if and only if the keyword that it has chosen is the same than the one defined by the uploader and if the label specifying the access right of the receiver is the same than the current one held by the server. To do so, we define the CBEKS primitive. We also specify the corresponding security models, namely computational consistency, indistinguishability against chosen keyword and ciphertext attacks, indistinguishability against keyword-guessing attacks and collusion resistance. We provide a CBEKS construction that is proven secure in the standard model with respect the aforementioned security models.

At a glance, by just simply observing the name of the primitive *certificate-based encryption with keyword search*, one may think that this is a trivial combination between a certificate-based encryption (CBE) scheme and a public-key encryption with keyword search (PEKS). Unfortunately, this is not the case. This is due to the fact that a CBE scheme [93] requires an interaction between one single certifier and one single receiver. This is not suitable for CBEKS as we involve many receivers. Furthermore, the certificates cannot be refreshed using update keys, but rather they need to be re-generated every time. Hence, the direct use of CBE scheme will not be satisfactory. More importantly, the CBE scheme in [93] only works in the random oracle model, and to make our CBEKS scheme useful in practice, the scheme must be proven secure in the standard model (i.e. to guarantee the security since CBEKS deals with sensitive data). Another attempt is to combine Fang et al.'s secure-channel free PEKS (SCF-PEKS) scheme [82] and the CBBE scheme from Section 4.2, which provides the broadcast mechanism. Again, the resulting scheme will not suffice to satisfy the requirements in CBEKS with a simple modification, and more importantly, if the modification is successul then the scheme will again be only secure in the random oracle model.

We summarize the advantages and disadvantages of the aforementioned primitives and combinations of primitives in Table 5.1. Observe that our CBEKS system satisfies all the requirements and features highlighted previously.

Cryptographic Primitive	Certificate	Broadcast	Key Update	Keyword
ABE [104]	×	×	×	×
CBBE [Sec. 4.2]	✓	✓	×	×
PEKS [82] + CBE [93]	✓	×	×	✓
PEKS [82] + CBBE [Sec. 4.2]	✓	✓	×	✓
CBEKS	✓	✓	✓	✓

Table 5.1: Comparisons of the new primitive CBEKS, the existing primitives ABE and CBBE and the combination of the primitive PEKS with the primitives CBE and CBBE respectively.

5.1.2 Protocol Definition

In the definition of the protocol, a label is a reference to some information about access rights (e.g. privileges and features). This label is supposed to be a unique element in $\mathbb{Z}_p$, for a prime number p, meaning that it refers to only one collection of rights and two labels with different collections cannot be equal.

We assume that the certifier and the server know the labels; however, none of the receivers should get any information about these elements. We presume that the certifier and the server communicate securely to agree on all the possible labels. We do not consider how they communicate in this protocol by making the hypothesis that they can proceed easily and naturally. For instance, in a hospital, let the certifier be the IT team and the server be the database center, meaning that they both belong to the administration staff of the hospital. Thus, there exists a way for the certifier and the server to share the information about the labels, since they are allowed to know the access rights and the privileges given to the hospital staff members.

A certificate-based encryption with keyword search (CBEKS) scheme is composed of the following seven algorithms:

- $\mathsf{Setup}(\lambda, s) \to (params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,s]})$. On inputs a security parameter λ and a total number of receivers s, output the public parameters *params*, the server's private key sk_S, the certifier's private key sk_C and the receivers' private keys $\{sk_{R,i}\}_{i\in[1,s]}$.

  As suggested above, the server and the certifier might receive information about the label framework. The public parameters *params* include the public keys of all the involved users.

- $\mathsf{Encrypt}(params, w^S) \to C_{w^S}$. On inputs the public parameters *params* and a uploader's keyword w^S, output the ciphertext C_{w^S} for w^S.

  Note that the keyword w^S is chosen by the uploader that encrypts the message m (e.g. a patient's EHR) and uploads the resulting ciphertext C_m along with C_{w^S} on the server. In this section, we do not focus on the encryption-decryption process for C_m but rather on the encryption-decryption process for C_{w^S}.

- $\mathsf{CertGen}(params, sk_C, L_1^R, i) \to Cert_{L_1^R,i}$. On inputs the public parameters *params*, the certifier's private key sk_C, a label L_1^R and a receiver i, output the receiver i's first certificate $Cert_{L_1^R,i}$ for L_1^R.

  The certifier might keep some elements used to generate $Cert_{L_1^R,i}$, such that the label L_1^R and additional information, on its local storage in order to create the update key $UK_{L_2^R}$. Let $AI_{L_1^R}$ be the auxiliary information that the certifier stores.

- $\mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j) \to UK_{L_j^R}$. On inputs the public parameters *params*, the certifier's private key sk_C, the auxiliary information $AI_{L_{j-1}^R}$, a label L_j^R where $1 < j$ and a group S_j of receivers, output the update key $UK_{L_j^R}$ for L_j^R and S_j.

  The certifier might keep some elements used to generate $UK_{L_j^R}$, such that the label L_j^R and additional information, on its local storage in order to create the update key $UK_{L_{j+1}^R}$. Let $AI_{L_j^R}$ be the auxiliary information that the certifier stores.

  The algorithm $\mathsf{UpdtKeyGen}$ is run in three cases:

  1. There is a new label L_j^R replacing L_{j-1}^R while $S_j = S_{j-1}$;
  2. There is a new group S_j replacing S_{j-1} while $L_j^R = L_{j-1}^R$;
  3. There are a new label L_j^R replacing L_{j-1}^R and a new group S_j replacing S_{j-1}.

In all cases, we write $AI_{L^R_{j-1}}$, L^R_j and S_j as the inputs for the algorithm UpdtKeyGen, such that $AI_{L^R_{j-1}}$ was computed given L^R_{j-1} and S_{j-1}. For $j = 2$, $AI_{L^R_1}$ corresponds to the auxiliary information from $Cert_{L^R_1,i}$, for $i \in [1,s]$. We assume that a description of the group S_j cam be found in $UK_{L^R_j}$.

- $\mathsf{UpdtCert}(params, Cert_{L^R_{j-1},i}, UK_{L^R_j}) \rightarrow Cert_{L^R_j,i}$. On inputs the public parameters $params$, the receiver i's previous certificate $Cert_{L^R_{j-1},i}$ for L^R_{j-1} and the key update $UK_{L^R_j}$ for L^R_j where $1 < j$, output the receiver i's refreshed certificate $Cert_{L^R_j,i}$ for L^R_j if $i \in S_j$; output $\perp$ otherwise.

- $\mathsf{TrapGen}(params, sk_{R,i}, w^R_i, Cert_{L^R_j,i}) \rightarrow Trap_{w^R_i,L^R_j}$. On inputs the public parameters $params$, the receiver i's private key $sk_{R,i}$, a receiver i's keyword w^R_i and a receiver i's certificate $Cert_{L^R_j,i}$ for L^R_j, output the receiver i's trapdoor $Trap_{w^R_i,L^R_j}$ for w^R_i and L^R_j.

- $\mathsf{Test}(params, sk_S, C_{w^S}, L^S_l, Trap_{w^R_i,L^R_j}) \rightarrow true/false$. On inputs the public parameters $params$, the server's private key sk_S, the ciphertext C_{w^S} for w^S, a label L^S_l where $1 \leq l$ and the receiver i's trapdoor $Trap_{w^R_i,L^R_j}$ for w^R_i and L^R_j where $1 \leq j$, ouptut $true$ if $[w^S = w^R_i] \wedge [L^S_l = L^R_j]$; output $false$ otherwise.

Correctness. For any $\lambda \in \mathbb{N}$, $(params, sk_S, sk_C, \{sk_{R,i}\}_{i \in [1,s]}) \leftarrow \mathsf{Setup}(\lambda, s)$, let a ciphertext be $C_{w^S} \leftarrow \mathsf{Encrypt}(params, w^S)$ for a keyword w^S. Let a receiver $i \in [1,s]$ have a first certificate $Cert_{L^R_1,i} \leftarrow \mathsf{CertGen}(params, sk_C, L^R_1, i)$ for a label L^R_1. Given an update key $UK_{L^R_j} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L^R_{j-1}}, L^R_j, S_j)$ such that $i \in S_j \subseteq [1,s]$, and a previous certificate $Cert_{L^R_{j-1},i}$ for a label L^R_{j-1}, the certificate is refreshed as follows: $Cert_{L^R_j,i} \leftarrow \mathsf{UpdtCert}(params, Cert_{L^R_{j-1},i}, UK_{L^R_j})$ for a label L^R_j where $1 < j$. Then, let the receiver i create a trapdoor $Trap_{w^R_i,L^R_j} \leftarrow \mathsf{TrapGen}(params, sk_{R,i}, w^R_i, Cert_{L^R_j,i})$ for a keyword w^R_i and the label L^R_j. If $j = 1$, then we simply use $Cert_{L^R_1,i} \leftarrow \mathsf{CertGen}(params, sk_C, L^R_1, i)$ during the trapdoor generation.

We require that a public-key encryption with keyword search scheme is correct if for a label L^S_l, $[w^S = w^R_i] \wedge [L^S_l = L^R_j]$ where $1 \leq j, l$, then $\mathsf{Test}(params, sk_S, C_{w^S}, L^S_l, Trap_{w^R_i,L^R_j})$ outputs $true$; otherwise, $\mathsf{Test}(params, sk_S, C_{w^S}, L^S_l, Trap_{w^R_i,L^R_j})$ outputs $false$.

N.B. We combine the algorithms Setup and KeyGen into an unique algorithm Setup.

5.1.3 Security Models

Before describing the security levels that we expect for our scheme, we recall the existing security models in the literature. In [82], the SCF-PEKS scheme is proven secure against chosen keyword and ciphertext attacks (IND-CKCA) and against keyword-guessing attacks (IND-KGA) in the standard model, as well as proven computationally consistent. More precisely, the scheme proposed in [82] is proven secure in terms of indistinguishability under chosen keyword and ciphertext attacks, meaning that

1. the server that has not obtained the trapdoors for given keywords cannot tell which ciphertext encrypts which keyword;
2. the receiver that has not obtained the server's private key cannot make any decisions about the ciphertexts, even though it gets all the trapdoors for the keywords that it holds.

In addition, a security proof is given in [82] to cover the notion of indistinguishability under keyword-guessing attack, that guarantees that an outsider (neither the server nor the receiver) that has obtained the trapdoor for a challenge keyword cannot observe the link between the trapdoor and any keywords. We will demonstrate that our CBEKS scheme reaches similar security notions regarding keywords and ciphertexts that we adapt to deal with labels and certificates.

In [42], a Broadcast Encryption (BE) scheme achieves fully collusion resistance (CR). More precisely, this system can broadcast a session key to any group of receivers and remains secure even if malicious non-authorised receivers collude. Such a property should be achieved to deal with the fact that the certifier forwards an update key to all the receivers, such that only a group of authorised receivers will successfully refresh their certificates.

We now give an overview of the threats and attacks that our CBEKS system should elude:

Security against the Server: The server handles the labels for the test process, meaning that it can obtain information from the certificates about the access rights given to the receivers. However, the server should learn nothing about the uploader's keyword (through the ciphertext) and the receivers' keyword (through the trapdoor), except whether they match or not. This is formalized in the indistinguishability against chosen keyword and ciphertext attack (IND-CKCA) game played by the server.

Security against the Certifier: As the server, the certifier knows the labels since it has the task to create the first certificates and update keys. Nevertheless, it should not be able to update itself the receivers' certificates. Moreover, even if intercepting ciphertexts and trapdoors, the certifier should not get any information about the keywords chosen by the uploader and the receivers respectively. This is formalized in the IND-CKCA game played by the certifier.

Security against the Receiver: The receiver gets the first certificate from the certifier, along with update keys, such that the latters can be efficiently used if and only if the receiver has been authorised by the certifier. The receiver should not be able to learn anything about the labels and so, the access rights embedded into its first certificate and the subsequent update keys. Note that the receiver can guess whether his/her refreshed certificate is a correct one or a fake one, since we suppose that the group of authorised receivers is contained in clear into each update key. Even waylaying a ciphertext, a receiver should not have the capability to know the embedded uploader's keyword, and even check that whether the keywords match. This is formalized in the IND-CKCA game played by the receiver.

Moreover, observe that the trapdoor is generated given a keyword and a certificate to avoid the following collusion attack. We suppose that the trapdoor only encrypts a keyword and that a receiver has to provide both his/her trapdoor and his/her certificate to the server in order to verifiy the matches of the keywords and the labels. In this scenario, we can let a first receiver compute the trapdoor encrypting a keyword and have only obsolete certificates, and a second receiver get a recent certificate that is still valid. Thus, these two receivers can manage to pass the test by sending to the server the trapdoor and the fresh certificate respectively.

Security against an Outsider: An outsider is neither the server nor the certifier nor a receiver. This outsider will guess keywords (for instance, keywords with low entropy) and check its choices in an off-line way. If the outsider has successfully initiated a keyword-guessing attack, then it can learn which keywords were chosen by the uploader and by the receivers, and so the security of the protocol might be broken. This is formalized in the indistinguishability against keyword-guessing attack (IND-KGA) game.

Collusion Resistance: The update keys are delivered by the certifier in order to let a group of authorised receivers to resfresh their certificates, regarding either a new label or a new group or both. This group is supposed to be included into the update key in clear, and the latter is sent to all the receivers. One important feature that the update key should satisfy is its collusion resistance: even if all the non-authorised receivers collude, they cannot generate a well-formed refreshed certificate from the update key. This is formalized in the collusion resistance game.

We provide several security games where the adversary plays the role of either the server or the certifier or a receiver. We also give a security game when the adversary acts as an outsider (neither the server nor the certifier nor a receiver) or as a group of colluding receivers.

The security models that we define below are computational consistency, indistinguishability against chosen keyword and ciphertext attack (IND-CKCA), indistinguishability against keyword-guessing attack (IND-KGA) and collusion resistance (CR). Compared to the IND-CKCA and IND-KGA models given in [82], the adversary has access to more oracles in our case, in order to satisfy the label-based situation of our protocol. Informally, in addition to the trapdoor queries and the test queries, the adversary can make first certificate queries, update key queries and refreshed certificate queries. If the adversary makes an update key query or a refreshed certificate query for a label L_j^R, then the challenger computes the requested element using the previous queried label L_{j-1}^R or a random label L_{j-1}^R for a first query.

In the collusion resistance game, we let the adversary select a group $S^* \subseteq [1,s]$ of receivers beforehand, and the challenger will reply to the adversary's queries according to this group S^*.

In summary, depending on the role that it is playing, the adversary is given access to different oracles:

- *First certificate query:* the adversary can ask the challenger for a first certificate query by giving a label L. The challenger responds by sending back a first certificate $Cert$ for L to the adversary.

- *Update key query:* the adversary can ask the challenger for an update key query by giving a label L. The challenger chooses a group of receivers S and responds by sending back an update key UK for L and S to the adversary.

- *Refreshed certificate query:* the adversary can ask the challenger for a refreshed certificate query by giving a label L. The challenger responds by sending back a refreshed certificate $Cert$ for L to the adversary.

- *Trapdoor query:* the adversary can ask the challenger for a trapdoor query by giving a keyword w and a label L. The challenger responds by sending back a trapdoor $Trap$ for w and L to the adversary.

- *Test query:* the adversary can ask the challenger for a test query by giving a ciphertext C, a keyword w and two labels L, L'. The challenger responds by sending back the result (either $true$ or $false$) to the adversary.

Consistency Model

The definition of consistency follows the ones given in [82, 1] except that the adversary has to choose more elements along with the two keywords: along with the uploader's keyword w^S, it selects a corresponding label L_l^S where $1 \leq l$, and along with the receiver's keyword w_i^R, it first selects a receiver $i \in [1,s]$ and then a label L_1^R as well as an index $1 < j$ indicating the number of times that the certificate should be refreshed.

Let λ be the security parameter and s the total number of receivers. Suppose there exists an adversary $\mathscr{A}$ that wants to make consistency fail. The consistency is formally defined through the experiment $Exp_{\mathscr{A}}^{Cons}(\lambda)$ as follows:

$(params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,s]}) \leftarrow \mathsf{Setup}(\lambda, s)$;

$((w^S, L_l^S), (w_i^R, L_1^R, j, i)) \leftarrow \mathscr{A}(params, m)$ for $1 \leq l,\ 1 < j$ and $i \in [1,s]$;

$C_{w^S} \leftarrow \mathsf{Encrypt}(params, w^S)$;

$Cert_{L_1^R,i} \leftarrow \mathsf{CertGen}(params, sk_C, L_1^R, i)$;

$UK_{L_j^R} \leftarrow \mathsf{UpdtKeyGen}^{(j-1)}(params, sk_C, AI_{L_1^R}, L_j^R, S_j)$;

$Cert_{L_j^R,i} \leftarrow \mathsf{UpdtCert}^{(j-1)}(params, sk_{R,i}, Cert_{L_1^R,i}, UK_{L_j^R})$;

$Trap_{w_i^R,L_j^R} \leftarrow \mathsf{TrapGen}(params, sk_R, w_i^R, Cert_{L_j^R,i})$;

If $i \in S_k$ for all $1 \leq k \leq j$, $(L_l^S = L_j^R)$, $(w^S \neq w_i^R)$ and $\mathsf{Test}(params, sk_S, C_{w^S}, L_l^S,$
$Trap_{w_i^R,L_j^R}) \rightarrow true$ then return 1, else return 0.

The advantage of $\mathscr{A}$ is defined as follows $Adv_{CBEKS,\mathscr{A}}^{Cons}(\lambda) = Pr[Exp_{CBEKS,\mathscr{A}}^{Cons}(\lambda) = 1]$. The scheme is said to be *computationally consistent* if any probabilistic polynomial time (PPT) adversary $\mathscr{A}$ wins the above experiment with negligible advantage.

N.B. The notation $\mathsf{UpdtKeyGen}^{(j-1)}$ denotes that the algorithm $\mathsf{UpdtKeyGen}$ is run $j-1$ times on inputs $L_2^R, L_3^R, \cdots, L_j^R$ respectively, and the notation $\mathsf{UpdtCert}^{(j-1)}$ denotes that the algorithm $\mathsf{UpdtCert}$ is run $j-1$ times on inputs $UK_{L_2^R}, UK_{L_3^R}, \cdots, UK_{L_j^R}$ respectively.

Note that $Exp_{CBEKS,\mathscr{A}}^{Cons}(\lambda) = 1$ can be formulated as "$\mathscr{A}$ succeeds".

Remark. Jeong et al. [124] noticed that the consistency of a SCF-PEKS scheme turns this scheme to not be secure against keyword-guessing attacks, in case the adversary is the server. Nevertheless, as suggested in [82], this attack should not be considered. Instead, we regard the keyword-guessing attacks launched by an outsider (neither the server nor the certifier nor a receiver).

Indistinguishability of CBEKS against Chosen Keyword and Ciphertext Attack Model

Let λ be the security parameter, **KeywS** be the keyword space, **LabS** be the label space and **CiphS** be the ciphertext space. Let an adversary $\mathscr{A}$ and a challenger $\mathscr{B}$ play the following three games $Game_S$, $Game_C$ and $Game_R$.

Game played by the Server: $Game_S$.

The adversary $\mathscr{A}$ is assumed to be the server (inside attacker).

Setup. $\mathscr{B}$ runs the algorithm Setup on inputs λ and s to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,s]}$. The challenger sends $params$ and sk_S to $\mathscr{A}$.

Query Phase 1. $\mathscr{A}$ makes the queries as follows:

- *First Certificate Query* $< L_1^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the first certificate query for any label $L_1^R \in$ **LabS** of its choice. The challenger answers by giving the first certificate $Cert_{L_1^R,i} \leftarrow \mathsf{CertGen}(params, sk_C, L_1^R, i)$ to $\mathscr{A}$ for which $i \in [1,s]$.
- *Update Key Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the update key query for any label $L_j^R \in$ **LabS**. The challenger first makes a first certificate query on L_{j-1}^R to get $AI_{L_{j-1}^R}$, and answers by giving the update key $UK_{L_j^R} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j)$ to $\mathscr{A}$ for which $S_j \subseteq [1,s]$.
- *Refreshed Certificate Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the refreshed certificate query for any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query on L_{j-1}^R to get $Cert_{L_{j-1}^R,i}$ for which $i \in [1,s]$ and an update key query on L_j^R to get $UK_{L_j^R}$, and answers by giving the refreshed certificate $Cert_{L_j^R} \leftarrow \mathsf{UpdtCert}(params, sk_{R,i}, UK_{L_j^R}, Cert_{L_{j-1}^R,i})$ to $\mathscr{A}$.
- *Trapdoor Query* $< w_i^R, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the trapdoor query for any keyword $w_i^R \in$ **KeywS** and any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query or a refreshed certificate query on L_j^R to get $Cert_{L_j^R,i}$ for which $i \in [1,s]$, and answers by giving the trapdoor $Trap_{w_i^R,L_j^R} \leftarrow \mathsf{TrapGen}(params, sk_{R,i}, w_i^R, Cert_{L_j^R,i})$ to $\mathscr{A}$.
- *Test Query* $< C_{w^S}, w_i^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any ciphertext $C_{w^S} \in$ **CiphS**, any keyword $w_i^R \in$ **KeywS** and any labels $L_l^S, L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query or a refreshed query on L_j^R to get $Cert_{L_j^R,i}$ and then a trapdoor query on w_i^R to get $Trap_{w_i^R,L_j^R}$ for which $i \in [1,s]$, and answers by giving the result $\mathsf{Test}(params, sk_S, C_{w^S}, L_l^S, Trap_{w_i^R,L_j^R})$ to $\mathscr{A}$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a challenge keyword pair (w_0, w_1) such that neither w_0 nor w_1 has been queried to obtain a corresponding trapdoor in the query phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and by computing a challenge ciphertext $C_{w_\mu} \leftarrow \mathsf{Encrypt}(params, w_\mu)$. The challenger sends C_{w_μ} to the adversary. Note that the challenger randomly selects the bit μ: we assume that the challenger cannot submit the same bit over and over.

Query Phase 2. $\mathscr{A}$ issues a number of queries as in the query phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R >=< w_0, L_j^R >$ or $< w_i^R, L_j^R >=< w_1, L_j^R >$, and $< C_{w^S}, w_i^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< C_{w^S}, w_i^R, L_l^S, L_j^R >=< C_{w_0}, w_0, L_l^S, L_j^R >$ or $< C_{w^S}, w_i^R, L_l^S, L_j^R >=< C_{w_1}, w_1, L_l^S, L_j^R >$.

Guess. The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.

We define the adversary's advantage in $Game_S$ by $Adv_{CBEKS,\mathscr{A}}^{Game_S}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Game played by the Certifier: ***Game$_C$*.**

The adversary $\mathscr{A}$ is assumed to be the certifier (outside attacker).

Setup. $\mathscr{B}$ runs the algorithm Setup on input λ and s to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,s]}$. The challenger sends $params$ and sk_C to $\mathscr{A}$.

Query Phase 1. $\mathscr{A}$ makes the queries as follows:

- *Refreshed Certificate Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the refreshed certificate query for any label $L_j^R \in$ **LabS** of its choice. The challenger first gets a first certificate $Cert_{L_{j-1}^R,i}$ for which $i \in [1,s]$ and an update key $UK_{L_j^R}$ for L_j^R, and answers by giving the refreshed certificate $Cert_{L_j^R} \leftarrow \mathsf{UpdtCert}(params, sk_{R,i}, UK_{L_j^R}, Cert_{L_{j-1}^R,i})$ to $\mathscr{A}$.
- *Trapdoor Query* $< w_i^R, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the trapdoor query for any keyword $w_i^R \in$ **KeywS** and any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a refreshed certificate query on L_j^R to get $Cert_{L_j^R,i}$ for which $i \in [1,s]$, and answers by giving the trapdoor $Trap_{w_i^R,L_j^R} \leftarrow \mathsf{TrapGen}(params, sk_{R,i}, w_i^R, Cert_{L_j^R,i})$ to $\mathscr{A}$.
- *Test Query* $< C_{w^S}, w_i^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any ciphertext $C_{w^S} \in$ **CiphS**, any keyword $w_i^R \in$ **KeywS** and any labels $L_l^S, L_j^R \in$ **LabS** of its choice. The challenger first makes a refreshed certificate query on L_j^R to get $Cert_{L_j^R,i}$ and then a trapdoor query on w_i^R to get $Trap_{w_i^R,L_j^R}$ for which $i \in [1,s]$, and answers by giving the result $\mathsf{Test}(params, sk_S, C_{w^S}, L_l^S, Trap_{w_i^R,L_j^R})$ to $\mathscr{A}$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a challenge keyword pair (w_0, w_1) such that neither w_0 nor w_1 has been queried to obtain a corresponding trapdoor in the query phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and by computing a challenge ciphertext $C_{w_\mu} \leftarrow \mathsf{Encrypt}(params, w_\mu)$. The challenger sends C_{w_μ} to the adversary. Note that the challenger randomly selects the bit μ: we assume that the challenger cannot submit the same bit over and over.

Query Phase 2. $\mathscr{A}$ issues a number of queries as in the query phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R >=< w_0, L_j^R >$ or $< w_i^R, L_j^R >=< w_1, L_j^R >$, and $< C_{w^S}, w_i^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< C_{w^S}, w_i^R, L_l^S, L_j^R >=< C_{w_0}, w_0, L_l^S, L_j^R >$ or $< C_{w^S}, w_i^R, L_l^S, L_j^R >=< C_{w_1}, w_1, L_l^S, L_j^R >$.

Guess. The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.

We define the adversary's advantage in $Game_C$ by $Adv_{CBEKS,\mathscr{A}}^{Game_C}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Game played by the Receiver: ***Game$_R$*.**

The adversary $\mathscr{A}$ is assumed to be the receiver (outside attacker).

Initialization. $\mathscr{A}$ begins by selecting an index $i^* \in [1,s]$ of the receiver that it wants to play.

Setup. $\mathscr{B}$ runs the algorithm Setup on input λ and s to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,s]}$. The challenger sends $params$ and sk_{R,i^*} to $\mathscr{A}$.

Query Phase 1. $\mathscr{A}$ makes the queries as follows:

- *First Certificate Query* $< L_1^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the first certificate query for any label $L_1^R \in$ **LabS** of its choice. The challenger answers by giving the first certificate $Cert_{L_1^R, i^*} \leftarrow \mathsf{CertGen}(params, sk_C, L_1^R, i^*)$ to $\mathscr{A}$.
- *Update Key Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the update key query for any label $L_j^R \in$ **LabS**. The challenger first makes a first certificate query on L_{j-1}^R to get $AI_{L_{j-1}^R}$, and answer by giving the update key $UK_{L_j^R} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j)$ to $\mathscr{A}$ for a group S_j of receivers that includes i^*.
- *Test Query* $< C_{w^S}, w_{i^*}^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any ciphertext $C_{w^S} \in$ **CiphS**, any keyword $w_{i^*}^R \in$ **KeywS** and any labels $L_l^S, L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query on L_j^R to get $Cert_{L_j^R, i^*}$ and generates $Trap_{w_{i^*}^R, L_j^R}$. and answers by giving the result $\mathsf{Test}(params, sk_S, C_{w^S}, L_l^S, Trap_{w_{i^*}^R, L_j^R}, Cert_{L_j^R, i^*})$ to $\mathscr{A}$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a challenge keyword pair (w_0, w_1) such that neither w_0 nor w_1 has been queried to obtain a corresponding trapdoor in the query phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and by computing a challenge ciphertext $C_{w_\mu} \leftarrow \mathsf{Encrypt}(params, w_\mu)$. The challenger sends C_{w_μ} to the adversary. Note that the challenger randomly selects the bit μ: we assume that the challenger cannot submit the same bit over and over.

Query Phase 2. $\mathscr{A}$ issues a number of queries as in the query phase 1. The restriction is that $< C_{w^S}, w_{i^*}^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< C_{w^S}, w_{i^*}^R, L_l^S, L_j^R > = < C_{w_0}, w_0, L_l^S, L_j^R >$ or $< C_{w^S}, w_{i^*}^R, L_l^S, L_j^R > = < C_{w_1}, w_1, L_l^S, L_j^R >$.

Guess. The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.

We define the adversary's advantage in $Game_R$ by $Adv_{CBEKS,\mathscr{A}}^{Game_R}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Definition. The CBEKS scheme is said to be IND-CKCA secure if $Adv_{CBEKS,\mathscr{A}}^{Game_S}$, $Adv_{CBEKS,\mathscr{A}}^{Game_C}$ and $Adv_{CBEKS,\mathscr{A}}^{Game_R}$ are all negligible.

Indistinguishability of CBEKS against Keyword-Guessing Attack Model

Let λ be the security parameter, **KeywS** be the keyword space, **LabS** be the label space and **CiphS** be the ciphertext space. Let $\mathscr{A}$ be an outside adversary that is neither the server nor the certifier nor the receiver and that makes the keyword-guessing attack by interacting with a challenger $\mathscr{B}$. We consider the following game.

Setup. $\mathscr{B}$ runs the algorithm Setup on input λ and s to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,s]}$. The challenger sends $params$ to $\mathscr{A}$.

Query Phase 1. $\mathscr{A}$ makes the queries as follows:

- *First Certificate Query* $< L_1^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the first certificate query for any label $L_1^R \in$ **LabS** of its choice. The challenger answers by giving the certificate $Cert_{L_1^R,i} \leftarrow \mathsf{CertGen}(params, sk_C, L_1^R, i)$ to $\mathscr{A}$ for which $i \in [1,s]$.
- *Update Key Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the update key query for any label $L_j^R \in$ **LabS**. The challenger first makes a first certificate query on L_{j-1}^R to get $AI_{L_{j-1}^R}$, and answer by giving the update key $UK_{L_j^R} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j)$ to $\mathscr{A}$ for which $S_j \in [1,s]$.
- *Refreshed Certificate Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the refreshed certificate query for any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query on L_{j-1}^R to get $Cert_{L_{j-1}^R,i}$ for which $i \in [1,s]$ and an update key query on L_j^R to get $UK_{L_j^R}$, and answers by giving the refreshed certificate $Cert_{L_j^R} \leftarrow \mathsf{UpdtCert}(params, sk_{R,i}, UK_{L_j^R}, Cert_{L_{j-1}^R,i})$ to $\mathscr{A}$.
- *Trapdoor Query* $< w_i^R, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the trapdoor query for any keyword $w_i^R \in$ **KeywS** and any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query or a refreshed certificate query on L_j^R to get $Cert_{L_j^R,i}$ for which $i \in [1,s]$, and answers by giving the trapdoor $Trap_{w_i^R,L_j^R} \leftarrow \mathsf{TrapGen}(params, sk_{R,i}, w_i^R, Cert_{L_j^R,i})$ to $\mathscr{A}$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a challenge keyword pair (w_0, w_1) such that neither w_0 nor w_1 has been queried to obtain a corresponding trapdoor in the query phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and a label L_j^R, and by computing a challenge trapdoor $Trap_{w_\mu, L_j^R} \leftarrow \mathsf{TrapGen}(params, sk_{R,i}, w_\mu, Cert_{L_j^R,i})$ where $Cert_{L_j^R,i}$ is the certificate issued for L_j^R. The challenger sends $Trap_{w_\mu, L_j^R}$ to the adversary. Note that the challenger randomly selects the bit μ: we assume that the challenger cannot submit the same bit over and over.

Query Phase 2. $\mathscr{A}$ issues a number of queries as in the query phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R > = < w_0, L_j^R >$ or $< w_i^R, L_j^R > = < w_1, L_j^R >$.

Guess. The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.

We define the adversary's advantage in the above game by $Adv_{CBEKS,\mathscr{A}}^{IND-KGA}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Definition. The CBEKS scheme is said to be IND-KGA secure if $Adv_{CBEKS,\mathscr{A}}^{IND-KGA}(\lambda)$ is negligible.

Collusion Resistance Model

Let λ be the security parameter, **KeywS** be the keyword space, **LabS** be the label space and **CiphS** be the ciphertext space. Let $\mathscr{A}$ be a group of colluding receivers that attacks the collusion resistance of the update keys by interacting with a challenger $\mathscr{B}$. We consider the following game.

Initialization. $\mathscr{A}$ begins by selecting a group $S^* \subseteq [1,s]$ of receivers that it wants to be challenged on.

Setup. $\mathscr{B}$ runs the algorithm Setup on input λ and s to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i \in [1,s]}$. The challenger sends $params$ and $\{sk_{R,i}\}_{i \in [1,s] \setminus S^*}$ to $\mathscr{A}$.

Query Phase 1. $\mathscr{A}$ makes the queries as follows:

- *First Certificate Query* $< L_1^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the first certificate query for any label $L_1^R \in$ **LabS** of its choice. The challenger answers by giving the certificate $Cert_{L_1^R,i} \leftarrow \mathsf{CertGen}(params, sk_C, L_1^R, i)$ to $\mathscr{A}$ for which $i \in S \subseteq S^*$.
- *Refreshed Certificate Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the refreshed certificate query for any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query on L_{j-1}^R to get $Cert_{L_{j-1}^R,i}$ for which $i \in S \subseteq S^*$, then computes an update key $UK_{L_j^R}$ for L_j^R, and answers by giving the refreshed certificate $Cert_{L_j^R} \leftarrow \mathsf{UpdtCert}(params, sk_{R,i}, UK_{L_j^R}, Cert_{L_{j-1}^R,i})$ to $\mathscr{A}$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a challenge label pair (L_0, L_1) such that neither L_0 nor L_1 has been queried to obtain a corresponding certificate or trapdoor in the query phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and by computing a challenge update key $UK_{L_\mu} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L_\mu - 1}, L_\mu, S^*)$. The challenger sends

UK_{L_μ} to the adversary. We denote by $L_\mu - 1$ the label preceding the label L_μ. Note that the challenger randomly selects the bit μ: we assume that the challenger cannot submit the same bit over and over.

Query Phase 2. $\mathscr{A}$ issues a number of queries as in the query phase 1. The restriction is that $< L_j^R >$ are not allowed to be queried as first certificate or refreshed certificate queries if $< L_j^R >=< L_0 >$ or $< L_j^R >=< L_1 >$.

Guess. The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.

We define the adversary's advantage in the above game by $Adv_{CBEKS,\mathscr{A}}^{CollRes}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Definition. The CBEKS scheme is said to be collusion resistant if $Adv_{CBEKS,\mathscr{A}}^{CollRes}(\lambda)$ is negligible.

5.1.4 Construction

The following CBEKS construction is inspired from the Boneh et al.'s broadcast encryption (BE) scheme [42]. A BE scheme allows a sender to forward encrypted information to all the recipients such that only a group of recipients selected by the sender can recover the original information in plain. Such property is useful to let the certifier to send encrypted update keys to all the receivers, while only some of them are able to successfully retrieve these update keys regarding the selection of the certifier. Thus, we let the certifier choose a group of receivers when it is generating an update key, such that only the receivers in this group will be able to correctly update their certificate. Therefore, it seems natural that the construction will lead to BE schemes. The main advantage of the Boneh et al.'s scheme is the constant size of both the receiver's private keys and the ciphertexts. The scheme is proved collusion resistant and selectively secure against chosen-ciphertext attacks in the standard model.

The construction below is also established on Waters' IBE scheme [235], where the algorithm KeyGen in the IBE scheme corresponds to the algorithm TrapGen in our CBEKS scheme. We let each receiver choose a keyword and generate a corresponding trapdoor that is given to the server in order to check that the receiver's keyword matches the uploader's one. Waters' scheme is efficient in that the private key and the ciphertext have constant size, and the decryption only involves two pairing computations. The scheme is proved semantically secure in the standard model. Observe that Abdalla et al. [1] showed that Waters' IBE scheme is not anonymous (meaning that the ciphertext might reveal the identity of the recipient). Moreover, Boneh et al. [40] presented a transformation of an IBE scheme into a PEKS scheme. Nevertheless, the authors noticed that the IBE scheme is required to be anonymous in order to provide a PEKS scheme against chosen message attacks. Therefore, such issue directly applied to our CBEKS scheme; however we manage to overcome it as follows.

First, note that in addition to the keywords that have to match, a receiver should provide a label L_j^R that corresponds to the label L_l^S held by the server for a successful test outcome. This means that a receiver meets two verification steps through the label and the keyword that enhance the security of the CBEKS scheme.

We now assume that the label of the receiver L_j^R matches the label of the server L_l^S. As noticed by Fang et al. [82], we have to ensure that an adversarial receiver cannot modify a ciphertext C_{w^S} into a new valid ciphertext C'_{w^S} without knowing the keyword w^S. However, this adversarial receiver would be able to generate a trapdoor $Trap_{w_i^R, L_j^R}$ for a guessed keyword w_i^R using its private key, and so could obtain the relation between the modified ciphertext C'_{w^S} and the trapdoor $Trap_{w_i^R, L_j^R}$ through interacting with the server as in a real environment. For this reason, Fang et al. [82] suggested to introduce a test query in the security model, as well as a strongly unforgeable one-time signature $\sigma = \mathsf{Sign}(ssk, (C_1, C_2, C_3, C_4, C_5))$ on the tuple $(C_1, C_2, C_3, C_4, C_5)$ such that $C_4 = g_1^{\kappa}$ and $C_5 = (A^{svk}B)^{\kappa}$ for a random exponent $\kappa \in_R \mathbb{Z}_p$ and a verification key svk.

Moreover, Waters' IBE scheme takes place in the standard model. Indeed, Waters provided a hash function $H : \{0,1\}^n \to \mathbb{G}_1$ that is collision resistant, while is not seen as a random oracle.

Our CBEKS construction is as follows:

- $\mathsf{Setup}(\lambda, s) \to (params, sk_S, sk_C, \{sk_{R,i}\}_{i \in [1,s]})$. Let λ be the security parameter and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be the bilinear map parameters. Pick at random $A, B \in_R \mathbb{G}_2$ and let $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a strongly unforgeable one-time signature scheme. Pick at random $\alpha \in_R \mathbb{Z}_p$ and compute $g_1^{\alpha^i}$ for $i \in [1,s] \cup [s+2, 2s]$ and $g_2^{\alpha^i}$ for $i \in [1,s]$. Pick at random $\beta \in_R \mathbb{Z}_p$ and compute g_1^{β} and g_2^{β}. Pick at random $\gamma \in_R \mathbb{Z}_p$ and compute $g_1^{\gamma\alpha^i}$ for $i \in [0,s]$. Pick at random $\delta_1, \cdots, \delta_s, \xi, \omega \in_R \mathbb{Z}_p$ and compute $g_1^{\delta_1}, \cdots, g_1^{\delta_s}$, g_2^{ξ} and g_1^{ω}.

  Finally, set the public parameters as follows: $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i \in [1,s] \cup [s+2,2s]}, \{g_2^{\alpha^i}\}_{i \in [1,s]}, g_2^{\beta}, \{g_1^{\delta_i}\}_{i \in [1,s]}, g_2^{\xi}, g_1^{\omega}, \mathsf{OTS})$. Set the receiver i's private key as follows: $sk_{R,i} = (\delta_i, g_1^{\beta}, g_1^{\gamma\alpha^i})$ for $i \in [1,s]$. Set the server's private key as follows: $sk_S = \xi$. Set the certifier's private key as follows: $sk_C = (\omega, g_1^{\gamma})$.

- $\mathsf{Encrypt}(params, w^S) \to C_{w^S}$. Let the keyword space be **KeywS** $= \{0,1\}^n$ where $2^n << p$. Choose $n+1$ random elements $e_0, e_1, \cdots, e_n \in_R \mathbb{Z}_p$ and compute $h_k = g_1^{e_k}$ for $k \in [0,n]$. Set $h = (h_0, h_1, \cdots, h_n)$ as the public description of the hash function $H : \{0,1\}^n \to \mathbb{G}_1$, and $e = (e_0, e_1, \cdots, e_n)$ is kept secret by the uploader.

  Second, select a one-time signature key pair $(ssk, svk) \leftarrow \mathsf{KeyGen}(\lambda)$. Pick at random $y, \kappa \in_R \mathbb{Z}_p$ and compute $g_1^{h(w^S)y}$, $(g_2^{\beta} \cdot g_2^{-w^S})^y = g_2^{(\beta - w^S)y}$, $(g_2^{\xi})^y$, g_1^{κ} and $(A^{svk}B)^{\kappa}$. Then, compute the one-time signature $\sigma = \mathsf{Sign}(ssk, (g_1^{h(w^S)y}, g_2^{(\beta - w^S)y}, g_2^{\xi y}, g_1^{\kappa}, (A^{svk}B)^{\kappa}))$. Set the ciphertext as follows: $C_{w^S} = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma) = (svk, g_1^{h(w^S)y}, g_2^{(\beta - w^S)y}, g_2^{\xi y}, g_1^{\kappa}, (A^{svk}B)^{\kappa}, \sigma)$.

- $\mathsf{CertGen}(params, sk_C, L_1^R, i) \to Cert_{L_1^R, i}$. Pick at random $r_1 \in_R \mathbb{Z}_p$ and compute $e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_1 L_1^R}$ and $g_2^{r_1}$. The certifier sends $(e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_1 L_1^R}, g_2^{r_1})$ to the receiver and the latter calculates $(g_2^{r_1})^{\delta_i}$ (where δ_i is one of the components of the receiver i's private key $sk_{R,i}$). Moreover, the certifier keeps $AI_{L_1^R} = g_1^{r_1 L_1^R}$ on its local storage. Finally, set the certificate as follows: $Cert_{L_1^{R,i}} = (cert_{1,1}, cert_{1,2}) = (e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_1 L_1^R}, g_2^{\delta_i r_1})$.

- $\mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j) \to UK_{L_j^R}$. Let $AI_{L_{j-1}^R} = g_1^{r_{j-1} L_{j-1}^R}$ and $S_j \subseteq$

$[1,s]$. Pick at random $s_j, r_j \in_R \mathbb{Z}_p$ and compute

$$\begin{aligned} & g_2^{s_j},\ g_2^{r_j}, (g_1^{\gamma}\cdot \prod_{k\in S_j} g_1^{\alpha^{s+1-k}})^{s_j}, e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j}\cdot \frac{e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R}}{e(g_1^{r_{j-1}L_{j-1}^R}, g_2^{\xi})^{\omega}} \\ = \ & e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j}\cdot e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R - r_{j-1}L_{j-1}^R} \end{aligned}$$

The certifier keeps $AI_{L_j^R} = g_1^{r_j L_j^R}$ on its local storage. Finally, set the update key as follows: $UK_{L_j^R} = (uk_{j,1}, uk_{j,2}, uk_{j,3}, uk_{j,4}) = (g_2^{s_j}, g_2^{r_j}, (g_1^{\gamma}\cdot\prod_{k\in S_j} g_1^{\alpha^{s+1-k}})^{s_j}, e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j}\cdot e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R - r_{j-1}L_{j-1}^R})$.

- $\mathsf{UpdtCert}(params, Cert_{L_{j-1}^R,i}, UK_{L_j^R}) \rightarrow Cert_{L_j^R,i}$. Suppose that $i \in S_j$. First, parse the certificate $Cert_{L_{j-1}^R,i}$ as $(cert_{j-1,1}, cert_{j-1,2}) = (e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_{j-1}L_{j-1}^R}, g_2^{\delta_i r_{j-1}})$ and the update key $UK_{L_j^R}$ as $(uk_{j,1}, uk_{j,2}, uk_{j,3}, uk_{j,4}) = (g_2^{s_j}, g_2^{r_j}, (g_1^{\gamma}\cdot\prod_{k\in S_j} g_1^{\alpha^{s+1-k}})^{s_j}, e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j}\cdot e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R - r_{j-1}L_{j-1}^R})$. Second, compute

$$\begin{aligned} \frac{e(uk_{j,3}, g_2^{\alpha^i})}{e(g_1^{\gamma\alpha^i}\cdot\prod_{k\in S_j, k\neq i} g_1^{\alpha^{s+1-k+i}}, uk_{j,1})} &= \frac{e((g_1^{\gamma}\cdot\prod_{k\in S_j} g_1^{\alpha^{s+1-k}})^{s_j}, g_2^{\alpha_i})}{e(g_1^{\gamma\alpha^i}\cdot\prod_{k\in S_j, k\neq i} g_1^{\alpha^{s+1-k+i}}, g_2^{s_j})} \\ &= e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j}, \\ \frac{uk_{j,4}}{e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j}} &= \frac{e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j}\cdot e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R - r_{j-1}L_{j-1}^R}}{e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j}} \\ &= e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R - r_{j-1}L_{j-1}^R}. \end{aligned}$$

  Then, compute $(uk_{j,2})^{\delta_i} = (g_2^{r_j})^{\delta_i}$ (where δ_i is one of the components of the receiver i's private key $sk_{R,i}$) and

$$\begin{aligned} cert_{j-1,1}\cdot(e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R - r_{j-1}L_{j-1}^R})^{\delta_i} &= e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_{j-1}L_{j-1}^R}\cdot(e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R - r_{j-1}L_{j-1}^R})^{\delta_i} \\ &= e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R} \end{aligned}$$

  Finally, set the refreshed certificate as follows: $Cert_{L_j^R,i} = (cert_{j,1}, cert_{j,2}) = (e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R}, g_2^{\delta_i r_j})$.

- $\mathsf{TrapGen}(params, sk_{R,i}, w_i^R, Cert_{L_j^R,i}) \rightarrow Trap_{w_i^R, L_j^R}$. First, parse the certificate $Cert_{L_j^R,i}$ as $(cert_{j,1}, cert_{j,2}) = (e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R}, g_2^{\delta_i r_j})$. Pick at random $v, x, z \in_R \mathbb{Z}_p$ and compute

$$\begin{aligned} & cert_{j,1}\cdot e(H(w_i^R), g_2^{\xi})^{\delta_i x} = e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R}\cdot e(H(w_i^R), g_2^{\xi})^{\delta_i x} \\ & e(H(w_i^R), g_2)^{\delta_i x} \\ & (g_1^{\beta}\cdot g_1^{-w_i^R})^{v}\cdot(H(w_i^R))^{\delta_i z} = g_1^{(\beta - w_i^R)v}\cdot H(w_i^R)^{\delta_i z} \\ & (g_2^{\xi})^{\delta_i z} \\ & cert_{j,2} = g_2^{\delta_i r_j} \\ & g_1^{v} \end{aligned}$$

Set the trapdoor as follows: $Trap_{w_i^R, L_j^R} = (T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6}) = (e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R} \cdot e(H(w_i^R), g_2^{\xi})^{\delta_i x}, e(H(w_i^R), g_2)^{\delta_i x}, g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{\delta_i z}, g_2^{\xi \delta_i z}, g_2^{\delta_i r_j}, g_1^{v})$.

- $\mathsf{Test}(params, sk_S, C_{w^S}, L_l^S, Trap_{w_i^R, L_j^R}) \rightarrow true/false$. First, parse the ciphertext C_{w^S} as $(C_0, C_1, C_2, C_3, C_4, C_5, \sigma) = (svk, g_1^{h(w^S)y}, g_2^{(\beta - w^S)y}, g_2^{\xi y}, g_1^{\kappa}, (A^{svk}B)^{\kappa}, \sigma)$ and the trapdoor $Trap_{w_i^R, L_j^R}$ as $(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6}) = (e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R} \cdot e(H(w_i^R), g_2^{\xi})^{\delta_i x}, e(H(w_i^R), g_2)^{\delta_i x}, g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{\delta_i z}, g_2^{\xi \delta_i z}, g_2^{\delta_i r_j}, g_1^{v})$.
  Second, test if $\mathsf{Verify}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = true$ and $e(C_4, A^{C_0}B) = e(g_1, C_5)$. Then, check that $T_{j,1} = e(g_1^{\omega}, T_{j,5})^{\xi L_l^S} \cdot (T_{j,2})^{\xi}$ and $\frac{e(T_{j,3}, C_3)}{e(C_1, T_{j,4})} = e(T_{j,6}, C_2)^{\xi}$. If the above equations hold, then output $true$; otherwise, output $false$.

Correctness. First, we get that

$$e(C_4, A^{C_0}B) = e(g_1^{\kappa}, A^{svk}B) = e(g_1, (A^{svk}B)^{\kappa}) = e(g_1, C_5).$$

We suppose that $w^S = w_i^R$ and $L_l^S = L_j^R$.

$$\begin{aligned}
T_{j,1} &= e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R} \cdot e(H(w_i^R), g_2^{\xi})^{\delta_i x} \\
&= e(g_1^{\omega}, g_2^{\delta_i r_j})^{\xi L_l^S} \cdot (e(H(w_i^R), g_2)^{\delta_i x})^{\xi} \\
&= e(g_1^{\omega}, T_{j,5})^{\xi L_l^S} \cdot (T_{j,2})^{\xi} \\
\frac{e(T_{j,3}, C_3)}{e(C_1, T_{j,4})} &= \frac{e(g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{\delta_i z}, g_2^{\xi y})}{e(g_1^{h(w^S)y}, g_2^{\xi \delta_i z})} \\
&= e(g_1^{(\beta - w_i^R)v}, g_2^{\xi y}) \cdot \frac{e(H(w_i^R)^{\delta_i z}, g_2^{\xi y})}{e(g_1^{h(w^S)y}, g_2^{\xi \delta_i z})} \\
&= e(g_1^{(\beta - w_i^R)v}, g_2^{\xi y}) \cdot \frac{e((g_1^{h(w_i^R)})^{\delta_i z}, g_2^{\xi y})}{e(g_1^{h(w^S)y}, g_2^{\xi \delta_i z})} \\
&= e(g_1^{(\beta - w^S)v}, g_2^{\xi y}) \cdot \frac{e((g_1^{h(w^S)})^{\delta_i z}, g_2^{\xi y})}{e(g_1^{h(w^S)y}, g_2^{\xi \delta_i z})} \\
&= e(g_1^{v}, g_2^{(\beta - w^S)y})^{\xi} = e(T_{j,6}, C_2)^{\xi}
\end{aligned}$$

5.1.5 Security Proofs

Proof of the Consistency Security

Theorem. Suppose that the DL assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$, then the CBEKS scheme is computationally consistent in the standard model.

Suppose there exists a PPT adversary $\mathscr{A}$ that attacks the computational consistency of the CBEKS scheme. A challenger $\mathscr{B}$ tries to solve the DL problem by playing the consistency game with the adversary $\mathscr{A}$ as follows.

$\mathscr{B}$ receives a DL problem instance $(g_1, g_1^{\delta_i}, g_2, g_2^{\delta_i})$, such that g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$, and has to output δ_i in $\mathbb{Z}_p$. Let $Adv^{DL}_{\mathscr{B},\mathbb{G}_1}(\lambda)$ be the advantage function that $\mathscr{B}$ solves the DL problem in $\mathbb{G}_1$ and $Adv^{DL}_{\mathscr{B},\mathbb{G}_2}(\lambda)$ be the advantage function that $\mathscr{B}$ solves the DL problem in $\mathbb{G}_2$.

Let (w^S, L_l^S) and (w_i^R, L_1^R, j, i) be the tuples that $\mathscr{A}$ returns in the consistency experiment. Let a ciphertext C_{w^S} be $(C_0 = svk, C_1 = g_1^{h(w^s)y}, C_2 = g_2^{(\beta - w^S)y}, C_3 = g_2^{\xi y}, C_4 = g_1^{\kappa}, C_5 = A^{svk}B)^{\kappa})$, for a one-time signature key pair $(ssk, svk) \leftarrow \mathsf{KeyGen}(\lambda)$, two elements A, B in $\mathbb{G}_2$ and two random exponents y, κ in $\mathbb{Z}_p$. Let a trapdoor $Trap_{w_i^R, L_j^R}$ be $(T_{j,1} = e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R} \cdot e(H(w_i^R)^{\delta_i}, g_2^{\xi})^x, T_{j,2} = e(H(w_i^R)^{\delta_i}, g_2)^x, T_{j,3} = g_1^{(\beta - w_i^R)v} \cdot (H(w_i^R)^{\delta_i})^z$, $T_{j,4} = (g_2^{\delta_i})^{\xi z}, T_{j,5} = (g_2^{\delta_i})^{r_j}, T_{j,6} = g_1^{v})$, for random exponents $\xi, \omega, r_j, v, x, z$ in $\mathbb{Z}_p$. We let $H(w_i^R)^{\delta_i}$ be computed as $(g_1^{\delta_i})^{h(w_i^R)}$.

We assume that the algorithm CertGen was run with input L_1^R and that the algorithms UpdtKeyGen and UpdtCert were called $j-1$ times to obtain $Cert_{L_j^R,i}$ and then $Trap_{w_i^R,L_j^R}$ since the receiver i is supposed to belong to S_k for all $1 \leq k \leq j$. In addition, we suppose that $L_l^R = L_j^R$; however, we do not need this hypothesis in our proof.

We suppose that the uploader's keyword and the receiver's keyword differ, i.e. $w^S \neq w_i^R$. The adversary wins exactly when $\frac{e(T_{j,3},C_3)}{e(C_1,T_{j,4})} = e(T_{j,6},C_2)^{\xi}$. Therefore, we get that:

$$\Leftrightarrow \quad \frac{e(g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{\delta_i z}, g_2^{\xi y})}{e(g_1^{h(w^S)y}, g_2^{\xi \delta_i z})} = e(g_1^{v}, g_2^{(\beta - w^S)y})^{\xi}$$

$$\Leftrightarrow \quad e(g_1^{(\beta - w_i^R)v}, g_2^{\xi y}) \cdot \frac{e(H(w_i^R)^{\delta_i z}, g_2^{\xi y})}{e(g_1^{h(w^S)y}, g_2^{\xi \delta_i z})} = e(g_1^{v}, g_2^{(\beta - w^S)y})^{\xi}$$

$$\Leftrightarrow \quad (\beta - w_i^R)v\xi y + (h(w_i^R) - h(w^S))\delta_i z \xi y = (\beta - w^S)v\xi y \mod p$$

We suppose that $\xi, y \neq 0 \mod p$ (the event that both ξ and y are equal to $0 \mod p$ happens with probability $1/p^2$), so that $(\beta - w_i^R)v + (h(w_i^R) - h(w^S))\delta_i z = (\beta - w^S)v \mod p$.

- *Case 1:* $h(w_i^R) = h(w^S) \neq 0 \mod p$. In this situation, we get that $(\beta - w_i^R)v = (\beta - w^S)v$, and so $w_i^R = w^S$ such that $v \neq 0 \mod p$ (with probability $1 - 1/p$). We thus obtain a contradiction as we have assumed that $w^S \neq w_i^R$.

- *Case 2:* $h(w_i^R) \neq h(w^S)$. In this situation, we get that $(h(w_i^R) - h(w^S))\delta_i z = v(\beta - w^S - \beta + w_i^R) \mod p$, and so $\delta_i = \frac{w_i^R - w^S}{h(w_i^R) - h(w^S)} \cdot \frac{v}{z} \mod p$, meaning that we have a solution to the DL problem.

- *Case 3:* $h(w) = 0 \mod p$ *for* $w = w_i^R \vee w^S$. Let us fix a keyword w. The exponents $e_0, e_1, \cdots, e_n$ are randomly selected in $\mathbb{Z}_p$. Note that the total number of possibilities for w is 2^n. Then, we have that the probability that $h(w) = 0 \mod p$ is equal to $2^n/p$.

Finally, if $i \in S_k$ for all $1 \leq k \leq j$, $L_l^S = L_j^R$, $w^S \neq w_i^R$ and $\mathsf{Test}(params, sk_S, C_{w^S}, L_l^S, Trap_{w_i^R, L_j^R}) \rightarrow true$, the advantage of $\mathscr{A}$ is upper bounded as follows:

$$Adv^{Cons}_{CBEKS,\mathscr{A}}(\lambda) = Pr[Exp^{Cons}_{CBEKS,\mathscr{A}}(\lambda) = 1] \leq \frac{1}{p^2} + \frac{1}{p} + \frac{2^n}{p} + Adv^{DL}_{\mathscr{B},\mathbb{G}_1}(\lambda) + Adv^{DL}_{\mathscr{B},\mathbb{G}_2}(\lambda).$$

To complete the above proof, authors in [82] mentioned that the private key of an identity *id* in Waters' IBE scheme [235] is computed as $(g^{s\cdot\alpha}H(id)^r, g)$, for random exponents $r,s \in \mathbb{Z}_p$, where $g^{s\cdot\alpha}$ corresponds to the master secret key of the protocol. Wherefore, if $h(id) = 0$, then one can retrieve the master secret key $g^{s\cdot\alpha}$, which threatens the robustness of this system.

Proof of the IND-CKCA Security

Theorem. Suppose that the SXDH assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$ and that OTS is a strongly unforgeable one-time signature scheme, then the CBEKS scheme is IND-CKCA secure in the standard model.

The proof of this theorem will result from the proofs of the three lemmas. These lemmas represent $Game_S$ (server), $Game_C$ (certifier) and $Game_R$ (receiver), respectively.

Game played by the Server: $Game_S$

Lemma. Suppose that the SXDH assumption holds, then the CBEKS scheme is semantically secure against a chosen keyword and ciphertext attack in $Game_S$ in the standard model.

Suppose that there exists a PPT adversary $\mathscr{A}$ in $Game_S$ that can attack the CBEKS scheme in the standard model with advantage $Adv^{Game_S}_{CBEKS,\mathscr{A}}(\lambda) \geq \varepsilon$. We build a challenger $\mathscr{B}$ that has advantage at least ε in solving the SXDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. $\mathscr{B}$ receives a random SXDH problem instance $(g_1, g_1^a, g_1^b, g_2, g_2^a, g_2^b, Z)$ where Z is either g_2^{ab} or a random element in $\mathbb{G}_2$, such that g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$. We let the challenger proceed as follows.

Setup. $\mathscr{B}$ computes the public parameters *params* and the server's private key sk_S as follows.

First, $\mathscr{B}$ chooses at random $\alpha \in_R \mathbb{Z}_p$ and generates $g_1^{\alpha^i}$ for $i \in [1,s] \cup [s+2, 2s]$ and $g_2^{\alpha^i}$ for $i \in [1,s]$. It also randomly chooses $\delta_1, \cdots, \delta_s, \xi, \omega \in_R \mathbb{Z}_p$ and computes $g_1^{\delta_1}, \cdots, g_1^{\delta_s}, g_2^{\xi}, g_1^{\omega}$. In addition, the challenger picks at random $A, B \in_R \mathbb{G}_2$ and chooses a strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i\in[1,s]\cup[s+2,2s]}, \{g_2^{\alpha^i}\}_{i\in[1,s]}, g_2^a, \{g_1^{\delta_i}\}_{i\in[1,s]}, g_2^{\xi}, g_1^{\omega}, \mathsf{OTS})$ and the server's private key $sk_S = \xi$. Note that since the exponents in $\mathbb{Z}_p$ are uniformly chosen at random, these public parameters have an identical distribution to that in the actual construction and that $\mathscr{B}$ has all the necessary values to compute the private key sk_S.

Let the keyword space **KeywS** be $\{0,1\}^n$. $\mathscr{B}$ chooses $n+1$ random elements $e_0, e_1, \cdots, e_n$ in $\mathbb{Z}_p$ and computes $h_i = g_1^{e_i}$ for $i \in [0,n]$. Let $h = (h_0, h_1, \cdots, h_n)$ be the public description of the hash function H. The algebraic hash function $H : \{0,1\}^n \rightarrow \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as $h(w) = e_0 + \sum_{i=1}^n (e_i \cdot w_i)$ and $H(w) = h_0 \cdot \prod_{i=1}^n (h_i^{w_i}) = g_1^{h(w)}$.

Query Phase 1. $\mathscr{A}$ makes the following queries:

- *First Certificate Query* $< L_1^R >$. If $\mathscr{A}$ queries L_1^R to the first certificate query generation oracle, then $\mathscr{B}$ picks at random $r_1 \in_R \mathbb{Z}_p$ and computes $e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_1 L_1^R}$ and $g_2^{r_1 \delta_i}$. It sends these two elements to $\mathscr{A}$ as the first certificate $Cert_{L_1^R,i}$.
- *Update Key Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the update key generation oracle, then $\mathscr{B}$ randomly selects γ, s_j, r_j in $\mathbb{Z}_p$ and generates $g_2^{s_j}$, $g_2^{r_j}$, $(g_1^{\gamma} \cdot \prod_{k \in S} g_1^{\alpha^{s+1-k}})^{s_j}$ for $S \subseteq [1,s]$, and $e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j} \cdot e(g_1^{\omega}, g_2^{\xi})^{r_j L_j^R - r_1 L_1^R}$. The challenger forwards these elements to $\mathscr{A}$ as the update key $UK_{L_j^R}$.
- *Refreshed Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the refreshed certificate generation oracle, then $\mathscr{B}$ picks at random $r_j \in_R \mathbb{Z}_p$ and sends the elements $e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R}$ and $g_2^{\delta_i r_j}$ to $\mathscr{A}$ as the refreshed certificate $Cert_{L_j^R,i}$.
- *Trapdoor Query* $< w_i^R, L_j^R >$. If $\mathscr{A}$ queries (w_i^R, L_j^R) to the trapdoor generation oracle, then $\mathscr{B}$ selects v, x, z at random in $\mathbb{Z}_p$, generates $e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R} \cdot e(H(w_i^R), g_2^{\xi})^{\delta_i x}$, $e(H(w_i^R), g_2)^{\delta_i x}$, $(g_1^a)^v \cdot g_1^{-w_i^R v} \cdot H(w_i^R)^{\delta_i z}$, $(g_2^{\xi})^{\delta_i z}$, $g_2^{\delta_i r_j}$ and g_1^v as the trapdoor $Trap_{w_i^R, L_j^R}$, and gives these elements to $\mathscr{A}$
- *Test Query* $< C_{w^S}, w_i^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any C_{w^S}, w_i^R, L_l^S and L_j^R. The challenger first makes a refreshed certificate query on L_j^R, then makes a trapdoor query on w_i^R and L_j^R, and responds to $\mathscr{A}$ by sending the result $\mathsf{Test}(params, sk_S, C_{w^S}, L_l^S, Trap_{w_i^R, L_j^R})$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a keyword pair (w_0, w_1). The challenger answers by choosing a random bit $\mu \in_R \{0,1\}$ and by letting the challenge keyword be $w^* = w_\mu$. Then, it selects a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathsf{KeyGen}(\lambda)$ and an exponent $\kappa \in_R \mathbb{Z}_p$, and sets $C_0 = svk^*$, $C_4 = g_1^{\kappa}$ and $C_5 = (A^{svk^*}B)^{\kappa}$. It also sets $C_1 = (g_1^b)^{h(w^*)}$, $C_2 = Z \cdot (g_2^b)^{(-w^*)}$ and $C_3 = (g_2^b)^{\xi}$, and generates a one-time signature $\sigma = \mathsf{Sign}(ssk^*, (C_1, C_2, C_3, C_4, C_5))$. The challenger sends the challenge ciphertext $C^* = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$ to the adversary.

When $Z = g_2^{ab}$, then C^* is a valid challenge ciphertext to $\mathscr{A}$ as in the real attack. When Z is random in $\mathbb{G}_2$, then $C_2 = Z \cdot (g_2^b)^{(-w^*)}$ is a uniform element in $\mathbb{G}_2$, and thus the ciphertext gives no information about the challenger's bit μ.

Query Phase 2. $\mathscr{A}$ continues to make queries as in the query phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R > = < w_0, L_j^R >$ or $< w_i^R, L_j^R > = < w_1, L_j^R >$ and $< C_{w^S}, w_i^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< C_{w^S}, w_i^R, L_l^S, L_j^R > = < C^*, w_0, L_l^S, L_j^R >$ or $< C_{w^S}, w_i^R, L_l^S, L_j^R > = < C^*, w_1, L_l^S, L_j^R >$.

Guess. The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu = \mu'$, then $\mathscr{B}$ outputs 1 meaning that $Z = g_2^{ab}$; otherwise, $\mathscr{B}$ outputs 0 meaning that Z is a random element in $\mathbb{G}_2$.

Analysis. When $Z = g_2^{ab}$, then the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$. When $Z \in_R \mathbb{G}_2$, then $C_2 = Z \cdot (g_2^b)^{(-w^*)}$ is uniformly random in $\mathbb{G}_2$, and thus $Pr[\mu' = \mu] = \frac{1}{2}$. It follows that we have $Adv^{SXDH}_{\mathscr{B},\mathbb{G}_1,\mathbb{G}_2}(\lambda) \geq \varepsilon$.

Game played by the Certifier: *Game*$_C$

Lemma. Suppose that the SXDH assumption holds, then the CBEKS scheme is semantically secure against a chosen keyword and ciphertext attack in *Game*$_C$ in the standard model.

Suppose that there exists a PPT adversary $\mathscr{A}$ in *Game*$_C$ that can attack the CBEKS scheme in the standard model with advantage $Adv^{Game_C}_{CBEKS,\mathscr{A}}(\lambda) \geq \varepsilon$. We build a challenger $\mathscr{B}$ that has advantage at least ε in solving the SXDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. $\mathscr{B}$ receives a random SXDH problem instance $(g_1, g_1^a, g_1^b, g_2, g_2^a, g_2^b, Z)$ where Z is either g_2^{ab} or a random element in $\mathbb{G}_2$, such that g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$. We let the challenger proceed as follows.

Setup. $\mathscr{B}$ computes the public parameters *params* and the certifier's private key sk_C as follows. First, $\mathscr{B}$ chooses at random $\beta \in_R \mathbb{Z}_p$ and computes g_2^{β}. It then selects at random $\alpha \in_R \mathbb{Z}_p$ and generates $g_1^{\alpha^i}$ for $i \in [1,s] \cup [s+2, 2s]$ and $g_2^{\alpha^i}$ for $i \in [1,s]$. It also randomly chooses $\gamma \in_R \mathbb{Z}_p$ and sets g_1^{γ}. Futhermore, it picks at random $\delta_1, \cdots, \delta_s, \omega \in_R \mathbb{Z}_p$ and computes $g_1^{\delta_1}, \cdots, g_1^{\delta_s}, g_1^{\omega}$. In addition, the challenger picks at random $A, B \in_R \mathbb{G}_2$ and chooses a strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i\in[1,s]\cup[s+2,2s]}, \{g_2^{\alpha^i}\}_{i\in[1,s]}, g_2^{\beta}, \{g_1^{\delta_i}\}_{i\in[1,s]}, g_2^a, g_1^{\omega}, \mathsf{OTS})$ and the certifier's private key $sk_C = (\omega, g_1^{\gamma})$. Note that since the exponents in $\mathbb{Z}_p$ are uniformly chosen at random, these public parameters have an identical distribution to that in the actual construction and that $\mathscr{B}$ has all the necessary values to compute the private key sk_C.

Let the keyword space **KeywS** be $\{0,1\}^n$. $\mathscr{B}$ chooses $n+1$ random elements $e_0, e_1, \cdots, e_n$ in $\mathbb{Z}_p$ and computes $h_i = g_1^{e_i}$ for $i \in [0,n]$. Let $h = (h_0, h_1, \cdots, h_n)$ be the public description of the hash function H. The algebraic hash function $H : \{0,1\}^n \to \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as

$$h(w) = e_0 + \sum_{i=1}^{n}(e_i \cdot w_i) \text{ and } H(w) = h_0 \cdot \prod_{i=1}^{n}(h_i^{w_i}) = g_1^{h(w)}.$$

Query Phase 1. $\mathscr{A}$ makes the following queries:

- *Refreshed Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the refreshed certificate generation oracle, then $\mathscr{B}$ picks at random $r_j \in_R \mathbb{Z}_p$ and sends the elements $e(g_1^{\delta_i}, g_2^a)^{\omega r_j L_j^R}$ and $g_2^{\delta_i r_j}$ to $\mathscr{A}$ as the refreshed certificate $Cert_{L_j^R,i}$.
- *Trapdoor Query* $< w_i^R, L_j^R >$. If $\mathscr{A}$ queries (w_i^R, L_j^R) to the trapdoor generation oracle, then $\mathscr{B}$ selects v, x, z at random in $\mathbb{Z}_p$, generates $e(g_1^{\delta_i}, g_2^a)^{\omega r_j L_j^R}$.

$e(H(w_i^R),g_2^a)^{\delta_i x}$, $e(H(w_i^R),g_2)^{\delta_i x}$, $g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{\delta_i z}$, $(g_2^a)^{\delta_i z}$, $g_2^{\delta_i r_j}$ and g_1^v as the trapdoor $Trap_{w_i^R,L_j^R}$, and gives these elements to $\mathscr{A}$.

- *Test Query* $< C_{w^S}, w_i^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any C_{w^S}, w_i^R, L_l^S and L_j^R. The challenger first makes a refreshed certificate query on L_j^R, then makes a trapdoor query on w_i^R and L_j^R, and responds to $\mathscr{A}$ by sending the result $\mathsf{Test}(params, sk_S, C_{w^S}, L_l^S, Trap_{w_i^R,L_j^R})$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a keyword pair (w_0, w_1). The challenger answers by choosing a random bit $\mu \in_R \{0,1\}$ and by letting the challenge keyword be $w^* = w_\mu$. Then, it selects a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathsf{KeyGen}(\lambda)$ and an exponent $\kappa \in_R \mathbb{Z}_p$, and sets $C_0 = svk^*$, $C_4 = g_1^\kappa$ and $C_5 = (A^{svk^*}B)^\kappa$. It also sets $C_1 = (g_1^b)^{h(w^*)}$, $C_2 = (g_2^b)^{(\beta - w^*)}$ and $C_3 = Z$, and generates a one-time signature $\sigma = \mathsf{Sign}(ssk^*, (C_1, C_2, C_3, C_4, C_5))$. The challenger sends the challenge ciphertext $C^* = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$ to the adversary.

When $Z = g_2^{ab}$, then C^* is a valid challenge ciphertext to $\mathscr{A}$ as in the real attack. When Z is random in $\mathbb{G}_2$, then $C_3 = Z$ is a uniform element in $\mathbb{G}_2$, and so C^* gives no information about the challenger's bit μ.

Query Phase 2. $\mathscr{A}$ continues to make queries as in the query phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R > = < w_0, L_j^R >$ or $< w_i^R, L_j^R > = < w_1, L_j^R >$ and $< C_{w^S}, w_i^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< C_{w^S}, w_i^R, L_l^S, L_j^R > = < C^*, w_0, L_l^S, L_j^R >$ or $< C_{w^S}, w_i^R, L_l^S, L_j^R > = < C^*, w_1, L_l^S, L_j^R >$.

Guess. The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $\mathscr{B}$ outputs 1 meaning that $Z = g_2^{ab}$; otherwise, $\mathscr{B}$ outputs 0 meaning that Z is a random element in $\mathbb{G}_2$.

Analysis. When $Z = g_2^{ab}$, then the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$. When $Z \in_R \mathbb{G}_2$, then $C_3 = Z$ is uniformly random in $\mathbb{G}_2$, and thus $Pr[\mu' = \mu] = \frac{1}{2}$. It follows that we have $Adv_{\mathscr{B},\mathbb{G}_1,\mathbb{G}_2}^{SXDH}(\lambda) \geq \varepsilon$.

Game played by the Receiver: $Game_R$

Lemma. Suppose that the SXDH assumption holds and that OTS is a strongly unforgeable one-time signature scheme, then the CBEKS scheme is semantically secure against a chosen keyword and ciphertext attack in $Game_R$ in the standard model.

Suppose that there exists a PPT adversary $\mathscr{A}$ in $Game_R$ that can attack the CBEKS scheme in the standard model with advantage $Adv_{CBEKS,\mathscr{A}}^{Game_R}(\lambda) \geq \varepsilon$. We build a challenger $\mathscr{B}$ that has advantage at least ε in solving the SXDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. $\mathscr{B}$ receives a random SXDH problem instance $(g_1^a, g_1^b, g_2^a, g_2^b, Z)$ where Z is either g_2^{ab} or a random element in $\mathbb{G}_2$, such that g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$.

Let $C^* = (svk^*, C_1, C_2, C_3, C_4, C_5, \sigma)$ be the challenge ciphertext given to the adversary in $Game_R$. Let $EvOTS$ be the event that $\mathscr{A}$ makes a test query for $C = (svk^*, C_1', C_2', C_3', C_4', C_5', \sigma')$ such that $\mathsf{Verify}(svk^*, \sigma', (C_1', C_2', C_3', C_4', C_5')) = true$. In the query phase 1, the adversary does not have any information on svk^*. Thus, the probability of a pre-challenge occurrence of $EvOTS$ does not exceed $q_{to} \cdot Bound$ where q_{to} denotes the total number of queries made to the test oracle and $Bound$ is the maximum probability that any one-time verification key svk^* is output by KeyGen (which does not exceed $1/p$ by assumption). In the query phase 2, $EvOTS$ produces an algorithm that breaks the strong unforgeability of the one-time signature. Thus, the probability $Pr[EvOTS] \leq q_{to}/p + Adv^{OTS}$, where Adv^{OTS} denotes the probability defined for a one-time signature (that should be negligible by assumption).

We let the challenger proceed as follows.

Initialization. $\mathscr{A}$ selects a receiver $i^* \in [1,s]$ as the one it wants to be challenged on.

Setup. $\mathscr{B}$ computes the public parameters $params$ and the private keys sk_{R,i^*} of the receiver i^* as follows. First, $\mathscr{B}$ chooses at random $\beta \in_R \mathbb{Z}_p$ and computes g_1^{β}, g_2^{β}. It then selects at random $\alpha \in_R \mathbb{Z}_p$ and generates $g_1^{\alpha^i}$ for $i \in [1,s] \cup [s+2,2s]$ and $g_2^{\alpha^i}$ for $i \in [1,s]$. It also randomly chooses $\gamma \in_R \mathbb{Z}_p$ and sets $g_1^{\gamma\alpha^{i^*}}$. Futhermore, it picks at random $\delta_1, \cdots, \delta_s, \omega \in_R \mathbb{Z}_p$ and computes $g_1^{\delta_1}, \cdots, g_1^{\delta_s}, g_1^{\omega}$. In addition, the challenger picks at random $A, B \in_R \mathbb{G}_2$ and chooses a strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i\in[1,s]\cup[s+2,2s]}, \{g_2^{\alpha^i}\}_{i\in[1,s]}, g_2^{\beta}, \{g_1^{\delta_i}\}_{i\in[1,s]}, g_2^{a}, g_1^{\omega}, \mathsf{OTS})$ and the receiver i^*'s private key $sk_{R,i^*} = (\delta_{i^*}, g_1^{\beta}, g_1^{\gamma\alpha^{i^*}})$. Note that since the exponents in $\mathbb{Z}_p$ are uniformly chosen at random, these public parameters have an identical distribution to that in the actual construction and that $\mathscr{B}$ has all the necessary values to compute the private keys sk_{R,i^*}.

Let the keyword space **KeywS** be $\{0,1\}^n$. $\mathscr{B}$ chooses $n+1$ random elements $e_0, e_1, \cdots, e_n$ in $\mathbb{Z}_p$ and computes $h_i = g_1^{e_i}$ for $i \in [0,n]$. Let $h = (h_0, h_1, \cdots, h_n)$ be the public description of the hash function H. The algebraic hash function $H : \{0,1\}^n \rightarrow \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as

$$h(w) = e_0 + \sum_{i=1}^{n}(e_i \cdot w_i) \text{ and } H(w) = h_0 \cdot \prod_{i=1}^{n}(h_i^{w_i}) = g_1^{h(w)}.$$

Query Phase 1. $\mathscr{A}$ makes the following queries:

- *First Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the first certificate query generation oracle, then $\mathscr{B}$ picks at random $r_1 \in_R \mathbb{Z}_p$ and computes $e(g_1^{\delta_{i^*}}, g_2^{a})^{\omega r_1 L_1^R}$ and $g_2^{r_1\delta_{i^*}}$. It sends these two elements to $\mathscr{A}$ as the first certificate $Cert_{L_1^R,i^*}$.
- *Update Key Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the update key generation oracle, then $\mathscr{B}$ randomly selects s_j, r_j in $\mathbb{Z}_p$ and generates $g_2^{s_j}$, $g_2^{r_j}$, $(g_1^{\gamma} \cdot g_1^{\alpha^{s+1-i^*}})^{s_j}$, and $e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j} \cdot e(g_1^{\omega}, g_2^{a})^{r_j L_j^R - r_1 L_1^R}$. The challenger forwards these elements to $\mathscr{A}$ as the update key $UK_{L_j^R}$.
- *Test Query* $< C_{w^S}, w_{i^*}^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any C_{w^S}, $w_{i^*}^R$, L_l^S and L_j^R. The challenger first makes a first certificate query on L_j^R, then computes the trapdoor $Trap_{w_{i^*}^R, L_j^R}$, and tests if $\mathsf{Verify}(C_0', \sigma', (C_1', C_2', C_3',$

$C'_4, C'_5)) = true$ and $e(C'_4, A^{C'_0}B) = e(g_1, C'_5)$. If the above equation hold and given $C_{w^S} = (C'_0, C'_1, C'_2, C'_3, C'_4, C'_5, \sigma')$ and $C^* = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$, then the challenger meets two cases:

1. If $C'_0 = svk' \equiv svk^* = C_0$, then we get that the tuples $(C'_1, C'_2, C'_3, C'_4, C'_5, \sigma')$ and $(C_1, C_2, C_3, C_4, C_5, \sigma)$ are not equal. Thus, the challenger sees an occurrence of the event $EvOTS$ and aborts.
2. If $C'_0 = svk' \neq svk^* = C_0$, then we get that $e(C'_4, A^{C'_0}B) = e(g_1, C'_5)$ such that $C'_5 = (A^{svk'}B)^{\kappa}$ since the ciphertext is supposed to be valid.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a keyword pair (w_0, w_1). The challenger answers by choosing a random bit $\mu \in_R \{0, 1\}$ and by letting the challenge keyword be $w^* = w_\mu$. Then, it selects a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathsf{KeyGen}(\lambda)$ and an exponent $\kappa \in_R \mathbb{Z}_p$, and sets $C_0 = svk^*$, $C_4 = g_1^{\kappa}$ and $C_5 = (A^{svk^*}B)^{\kappa}$. It also sets $C_1 = (g_1^b)^{h(w^*)}$, $C_2 = (g_2^b)^{(\beta - w^*)}$ and $C_3 = Z$, and generates a one-time signature $\sigma = \mathsf{Sign}(ssk^*, (C_1, C_2, C_3, C_4, C_5))$. The challenger sends the challenge ciphertext $C^* = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$ to the adversary.

When $Z = g_2^{ab}$, then C^* is a valid challenge ciphertext to $\mathscr{A}$ as in the real attack. When Z is random in $\mathbb{G}_2$, then $C_3 = Z$ is a uniform element in $\mathbb{G}_2$, and thus the ciphertext gives no information about the challenger's bit μ.

Query Phase 2. $\mathscr{A}$ continues to make queries as in the query phase 1. The restriction is that $< C_{w^S}, w^R_{i^*}, L^S_l, L^R_j >$ are not allowed to be queried as test queries if $< C_{w^S}, w^R_{i^*}, L^S_l, L^R_j > = < C^*, w_0, L^S_l, L^R_j >$ or $< C_{w^S}, w^R_{i^*}, L^S_l, L^R_j > = < C^*, w_1, L^S_l, L^R_j >$.

Guess. The adversary outputs a bit $\mu' \in \{0, 1\}$. If $\mu' = \mu$, then $\mathscr{B}$ outputs 1 meaning that $Z = g_2^{ab}$; otherwise, $\mathscr{B}$ outputs 0 meaning that Z is a random element in $\mathbb{G}_2$.

Analysis. If the event $EvOTS$ does not occur and when $Z = g_2^{ab}$, then the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$. When $Z \in_R \mathbb{G}_2$, then $C_3 = Z$ is uniformly random in $\mathbb{G}_2$, and thus $Pr[\mu' = \mu] = \frac{1}{2}$. It follows that we have $Adv^{SXDH}_{\mathscr{B}, \mathbb{G}_1, \mathbb{G}_2}(\lambda) \geq \varepsilon + \frac{q_{to}}{p} + Adv^{OTS}$.

Proof of the IND-KGA Security

Theorem. Suppose that the DBDH assumption holds $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, then the CBEKS scheme is IND-KGA secure in the standard model.

Suppose that there exists a PPT adversary $\mathscr{A}$ that can trigger a keyword-guessing attack against the CBEKS scheme in the standard model with advantage $Adv^{IND-KGA}_{CBEKS, \mathscr{A}}(\lambda)$. We build a challenger $\mathscr{B}$ that plays the DBDH game in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ by interacting with the adversary. $\mathscr{B}$ receives a random DBDH problem instance $(g_1, g_1^a, g_1^b, g_2, g_2^b, g_2^c, Z)$ and outputs a bit $\nu' \in \{0, 1\}$ as a guess to decide whether Z is either equal to $e(g_1, g_2)^{abc}$ or a random element in $\mathbb{G}_T$, such that g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$.

The proof is divided into four games, namely $Game_1$, $Game_2$, $Game_3$ and $Game_4$, that differ by slight modifications. In each game, the challenger will output a bit ν' that is well defined. Let E_i be the event that the adversary is successful in $Game_i$, for $i \in [1, 4]$. Such process will allow us to conclude.

Game$_1$. This game is simply the same than the original IND-KGA security game. In the following, we describe the simulation of the challenger.

At the start of *Game*$_1$, $\mathscr{B}$ chooses three random exponents a, b and c uniformly in $\mathbb{Z}_p$. Nevertheless, in the next games, $\mathscr{B}$ will stop to use a, b and c; instead, it will require $g_1^a, g_1^b, g_2^b, g_2^c$ along with either $Z = e(g_1, g_2)^{abc}$ or $Z \in_R \mathbb{G}_T$ for the security game simulation.

The adversary and the challenger play *Game*$_1$ following several steps as in the definition of the IND-KGA security game. First, the challenger generates the public parameters *params* and gives them to $\mathscr{A}$. Then, the adversary initiates the query phase 1 and has access to the various oracles: $\mathscr{A}$ can make first certificate, update key, refreshed certificate and trapdoor queries. After this first phase of queries, the adversary chooses a challenge keyword pair (w_0, w_1). The challenger chooses a bit $\mu \in_R \{0,1\}$ at random and sets the challenge keyword as $w^* = w_\mu$. Thereafter, $\mathscr{A}$ makes other queries to the same oracles such that the trapdoors resulting from trapdoor queries have to embed a keyword that differs from w^*. Eventually, the adversary returns a bit $\mu' \in \{0,1\}$ as a guess for μ. If $\mu' = \mu$, then $\mathscr{B}$ returns $\nu' = 1$; otherwise, it returns $\nu' = 0$. This completes the description of the challenger's simulation in *Game*$_1$.

We give now more details about the trapdoor queries that $\mathscr{A}$ makes, for some keyword w and label L. Let $q_{td} \in \mathbb{N}$ be the total number of trapdoor queries. Let $\bar{\mathscr{W}}$ be the set containing all the keywords selected for the trapdoor queries. We assume that the challenge keyword does not belong to $\bar{\mathscr{W}}$ since the restriction during the challenge phase specifies that w^* has not been and will not be queried in the query phase 1 and the query phase 2, respectively. Let $\mathscr{W} \subseteq \bar{\mathscr{W}}$ be the subset of the requested keywords such that all multiples from $\bar{\mathscr{W}}$ are removed. We suppose that $|\mathscr{W}| = q_0 \leq q_{td} = |\bar{\mathscr{W}}|$ such that two keywords in $\mathscr{W}$ are necessarily different. Finally, we define $\mathscr{W}^* = \mathscr{W} \cup \{w^*\}$.

Game$_2$. This game is almost identical to *Game*$_1$ except that the challenger sets some elements differently.

The challenger computes $M = 2q_{td}$ and randomly chooses $k \in_R [1,n]$. $\mathscr{B}$ then selects at random a tuple $x = (x_0, x_1, \cdots, x_n)$ where $x_i \in_R [0, M-1]$ for $i \in [1,n]$, and a tuple $y = (y_0, y_1, \cdots, y_n)$ where $y_i \in_R \mathbb{Z}_p$ for $i \in [1,n]$. We assume that these elements are kept secret by the challenger.

Given a keyword of the form $w = (w_1, \cdots, w_n)$, let us define three functions as follows:

$$\begin{aligned} x(w) &= x_0 + \sum_{i=1}^{n} (x_i \cdot w_i) \\ y(w) &= (p - Mk) + y_0 + \sum_{i=1}^{n} (y_i \cdot w_i) \\ h(w) &= x(w) + ay(w) \end{aligned}$$

The challenger generates the hash function H with public description $h = (h_0, h_1, \cdots, h_n) \in \mathbb{G}_1^{n+1}$ as follows. $\mathscr{B}$ lets $h_0 = g_1^{x_0}(g_1^a)^{p-Mk+y_0}$ and $h_i = g_1^{x_i}(g_1^a)^{y_i}$ for $i \in [1,n]$. Observe that such setting does not affect the distribution of the outputs of the hash function H.

We now describe the techniques employed in [235, 82] to construct the security proof. As in [82], let $VIEW_{\mathscr{A}}$ be the adversary's random tape and the transcript of its interactions with its oracles in the simulation of *Game*$_2$. In other words, let us fix all the random elements that $\mathscr{A}$ is able to learn during its execution, including its random coin tosses. More precisely, we fix the public parameters *params*, the challenge bit μ and the randomness used in answering the trapdoor and other queries. This means that $\mathscr{A}$ can be seen as a deterministic algorithm, and thus the set $\mathscr{W}^*$ can be seen as fixed.

Let $\bar{\mathscr{Y}} = (y_0, y_1, \cdots, y_n, k)$ where the elements are distributed as above. Therefore, if $VIEW_{\mathscr{A}}$ is fixed and the game is conducted again, then $\bar{\mathscr{Y}}$ has the same distribution as for a run of the game without having $VIEW_{\mathscr{A}}$ as fixed. This happens due to the random "masking" values x_i.

- *Forced Abort:* Let *FAbort* be the event that one of the following conditions is true:

  1. $\mathscr{A}$ asks a trapdoor query for a keyword w and a label L such that $y(w) = 0 \mod p$;
  2. $\mathscr{A}$ chooses a keyword pair (w_0, w_1) such that neither w_0 nor w_1 is equal to $0 \mod p$. If *FAbort* occurs then the challenger aborts and ν' is chosen at random.

We will modify the next games in order to force the challenger to abort each time that the above event happens (so we call the event a forced abort). For every fixed $VIEW_{\mathscr{A}}$, we define $\tau(VIEW_{\mathscr{A}}) = Pr_{\bar{\mathscr{Y}}}[\neg FAbort]$ where $\neg FAbort$ denotes the complementary event of $FAbort$.

Let ζ_{low} and ζ_{up} be the lower and upper bounds on $\tau(VIEW_{\mathscr{A}})$ respectively. We get the following lemma:

Lemma 1. For every fixed $VIEW_{\mathscr{A}}$, we define $\zeta_{low} = \frac{1}{4(n+1)q_{td}}$ and $\zeta_{up} = \frac{1}{2q_{td}}$. Thus, we have that

$$\zeta_{low} \leq \tau(VIEW_{\mathscr{A}}) \leq \zeta_{up}.$$

This lemma is also (partly) used in [132, 235, 82]. Note that a proof of these two bounds can be found in [82]. We write the proof of this lemma at the end of this section.

We now explicit the modifications made between $Game_1$ and $Game_2$. We assume that $VIEW_{\mathscr{A}}$ is fixed since we assume that the adversary has terminated the execution. Two events occuring in $Game_2$ but not in $Game_1$ can be described as follows:

- *Forced Abort:* After the output of the adversary's bit μ' as a guess for μ, check whether *FAbort* happens or not. If it occurs, then a random bit ν' is returned and the challenger aborts; otherwise, the challenger keeps on as before.

- *Artificial Abort:* To discard some unwanted dependency on probabilities, some artificial aborts are added such that the challenger always aborts with probability approximately equal to $1 - \zeta_{low}$ and independently of $VIEW_{\mathscr{A}}$. More precisely, after the output of the adversary's bit μ' as a guess for μ, check whether *FAbort* happens or not. If it occurs, then a random bit ν' is returned and the challenger aborts; otherwise, the challenger keeps on as follows. $\mathscr{B}$ first samples an estimate $\tau'(VIEW_{\mathscr{A}})$ of the probability $\tau(VIEW_{\mathscr{A}})$ that *FAbort* does not occur. Remember that $VIEW_{\mathscr{A}}$ is fixed now, thus the sampling does not involve running the adversary again. This estimate $\tau'(VIEW_{\mathscr{A}})$ is defined as a random variable and only depends on the keywords belonging to $\mathscr{W}^*$ and the randomness used to sample.

We now explicit the two cases that the challenger can encounter:

1. If $\tau'(VIEW_{\mathscr{A}}) \leq \zeta_{low}$, then we assume that $\mathscr{B}$ keeps on as before;
2. If $\tau'(VIEW_{\mathscr{A}}) > \zeta_{low}$, then the challenger aborts and outputs a bit ν' with probability equal to $1 - \frac{\zeta_{low}}{\tau'(VIEW_{\mathscr{A}})}$. In other words, the challenger does not abort and keeps on as before with probability equal to $\frac{\zeta_{low}}{\tau'(VIEW_{\mathscr{A}})}$.

The description of *Game*$_2$ is now completed.

The following claim is postulated in [132] and proved by Fang et al. [82]. It enables to bound the probabilities of E_1 and E_2. We recall its proof at the end of this section.

Claim 1. We define $\rho(\lambda) \equiv Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda) \cdot q_{td}(n+1) > 0$. If the experiment takes $s_0(\lambda) = O(n^2(\rho(\lambda))^{-2}\log((nq_{td} \cdot \rho(\lambda))^{-1}))$ samples when computing the estimate $\tau'($ $VIEW_{\mathscr{A}})$, then

$$|Pr[E_1] - (\frac{1}{2} + (Pr[E_2] - \frac{1}{2}) \cdot 4q_{td}(n+1))| \leq \rho(\lambda).$$

Game$_3$. This game is similar to *Game*$_2$ except that the public parameters and the trapdoors are generated differently. We suppose now that Z is equal to $e(g_1, g_2)^{abc}$.

Setup. $\mathscr{B}$ computes the public parameters *params* as follows.

First, $\mathscr{B}$ chooses at random $\beta \in_R \mathbb{Z}_p$ and computes g_2^{β}. It then selects at random $\alpha \in_R \mathbb{Z}_p$ and generates $g_1^{\alpha^i}$ for $i \in [1,s] \cup [s+2,2s]$ and $g_2^{\alpha^i}$ for $i \in [1,s]$. Futhermore, it picks at random $\delta_1, \cdots, \delta_s, \omega \in_R \mathbb{Z}_p$ and computes $g_1^{\delta_1}, \cdots, g_1^{\delta_s}, g_1^{\omega}$. In addition, the challenger picks at random $A, B \in_R \mathbb{G}_2$ and chooses a strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i \in [1,s] \cup [s+2,2s]}, \{g_2^{\alpha^i}\}_{i \in [1,s]}, g_2^{\beta}, \{g_1^{\delta_i}\}_{i \in [1,s]}, g_2^{c}, g_1^{\omega}, \mathsf{OTS})$. Note that since the exponents in $\mathbb{Z}_p$ are uniformly chosen at random, these public parameters have an identical distribution to that in the actual construction.

The challenger also outputs the public description $(h_0, h_1, \cdots, h_n)$ of the hash function H as defined in *Game*$_2$.

Query Phase 1. $\mathscr{A}$ can adaptively issue queries as follows.

- *First Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the first certificate query generation oracle, then $\mathscr{B}$ picks at random $r_1 \in_R \mathbb{Z}_p$ and computes $e(g_1^{\delta_i}, g_2^{c})^{\omega r_1 L_1^R}$ and $g_2^{r_1 \delta_i}$ for $i \in [1,s]$. It sends these two elements to $\mathscr{A}$ as the first certificate $Cert_{L_1^R,i}$.
- *Update Key Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the update key generation oracle, then $\mathscr{B}$ randomly selects γ, s_j, r_j in $\mathbb{Z}_p$ and generates $g_2^{s_j}$, $g_2^{r_j}$, $(g_1^{\gamma} \cdot \prod_{k \in S_j} g_1^{\alpha^{s+1-k}})^{s_j}$, for $S_j \subseteq [1,s]$, and $e(g_1^{\alpha}, g_2^{\alpha^s})^{s_j} \cdot e(g_1^{\omega}, g_2^{c})^{r_j L_j^R - r_1 L_1^R}$. The challenger forwards these elements to $\mathscr{A}$ as the update key $UK_{L_j^R}$.
- *Refreshed Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the refreshed certificate generation oracle, then $\mathscr{B}$ picks at random $r_j \in_R \mathbb{Z}_p$ and sends the elements $e(g_1^{\delta_i}, g_2^{c})^{\omega r_j L_j^R}$ and $g_2^{\delta_i r_j}$ to $\mathscr{A}$ as the refreshed certificate $Cert_{L_j^R,i}$.

- *Trapdoor Query* $< w_i^R, L_j^R >$. Suppose that $\mathscr{A}$ queries (w_i^R, L_j^R) to the trapdoor generation oracle. If $y(w_i^R) \neq 0 \mod p$, then $\mathscr{B}$ selects v, z at random in $\mathbb{Z}_p$, generates $e(g_1^{\delta_i}, g_2^c)^{\omega r_j L_j^R} \cdot e(g_1^b, g_2^c)^{\delta_i x(w_i^R)} \cdot Z^{\delta_i y(w_i^R)}$, $e(g_1, g_2^b)^{\delta_i x(w_i^R)} \cdot e(g_1^a, g_2^b)^{\delta_i y(w_i^R)}$, $g_1^{(\beta - w_i^R)v} \cdot g_1^{\delta_i z x(w_i^R)} \cdot (g_1^a)^{\delta_i z y(w_i^R)}$, $(g_2^c)^{\delta_i z}$, $g_2^{\delta_i r_j}$ and g_1^v as the trapdoor $Trap_{w_i^R, L_j^R}$, and gives these elements to $\mathscr{A}$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a keyword pair (w_0, w_1). The challenger first chooses a random bit $\mu \in_R \{0,1\}$ and lets the challenge keyword be $w^* = w_\mu$. It also defines a label L^*. It selects v, z at random in $\mathbb{Z}_p$ and generates $T_1 = e(g_1^{\delta_i}, g_2^c)^{\omega r_j L^*} \cdot e(g_1^b, g_2^c)^{\delta_i x(w^*)} \cdot Z^{\delta_i y(w^*)}$, $T_2 = e(g_1, g_2^b)^{\delta_i x(w^*)} \cdot e(g_1^a, g_2^b)^{\delta_i y(w^*)}$, $T_3 = g_1^{(\beta - w^*)v} \cdot g_1^{\delta_i z x(w^*)} \cdot (g_1^a)^{\delta_i z y(w^*)}$, $T_4 = (g_2^c)^{\delta_i z}$, $T_5 = g_2^{\delta_i r_j}$ and $T_6 = g_1^v$ as the elements of the trapdoor $Trap^*$. The challenger sends the challenge trapdoor $Trap^* = (T_1, T_2, T_3, T_4, T_5, T_6)$ to the adversary.

When $Z = e(g_1, g_2)^{abc}$, then $Trap^*$ is a valid challenge trapdoor to $\mathscr{A}$ as in the real attack. When Z is random in $\mathbb{G}_T$, then $e(g_1^{\delta_i}, g_2^c)^{\omega r_j L_j^R} \cdot e(g_1^b, g_2^c)^{\delta_i x(w^*)} \cdot Z^{\delta_i y(w^*)}$ is a uniform element in $\mathbb{G}_T$, and thus the trapdoor gives no information about the challenger's bit μ.

Query Phase 2. $\mathscr{A}$ continues to make queries as in the query phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R > = < w_0, L^* >$ or $< w_i^R, L_j^R > = < w_1, L^* >$.

Guess. The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $\mathscr{B}$ outputs 1 meaning that $Z = e(g_1, g_2)^{abc}$; otherwise, $\mathscr{B}$ outputs 0 meaning that Z is a random element in $\mathbb{G}_T$.

We easily observe that the public parameters and the trapdoors are distributed identically in *Game*$_2$ and in *Game*$_3$. Therefore, we obtain that $Pr[E_2] = Pr[E_3]$.

Game$_4$. *Game*$_4$ is similar to *Game*$_3$ except that the value $Z = e(g_1, g_2)^{abc}$ is replaced by a random value Z in $\mathbb{G}_T$, which is chosen at the beginning of the game. Therefore, $|Pr[E_3] - Pr[E_4]| \leq Adv_{\mathscr{B}, \mathbb{G}_T}^{DBDH}(\lambda)$.

To explain why, we suppose that the tuple $(g_1, g_1^a, g_1^b, g_2, g_2^b, g_2^c, Z)$ is given to the challenger, where Z is either equal to $Z = e(g_1, g_2)^{abc}$ or to a random element in $\mathbb{G}_T$, such that g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$. When $Z = e(g_1, g_2)^{abc}$, we perfectly simulate the adversary in *Game*$_3$. When $Z \in_R \mathbb{G}_T$, then $e(g_1^{\delta_i}, g_2^c)^{\omega r_j L_j^R} \cdot e(g_1^b, g_2^c)^{\delta_i x(w^*)} \cdot Z^{\delta_i y(w^*)}$ is uniformly random in $\mathbb{G}_T$, and thus we perfectly simulate the adversary in *Game*$_4$. This means that if $\mathscr{A}$ can distinguish between *Game*$_3$ and *Game*$_4$, then the two possible values for Z can be distinguished with the same probability. Therefore, when Z is uniformly random in $\mathbb{G}_T$, we have that $Pr[E_4] = \frac{1}{2}$.

Analysis. We have described the simulations of all the games, and now, we can complete the proof by bounding the advantage of the adversary in the IND-KGA security game as follows:

$$\begin{aligned}
Adv^{IND-KGA}_{CBEKS,\mathscr{A}}(\lambda) &= |Pr[E_1]-\frac{1}{2}| \\
&\leq |(Pr[E_2]-\frac{1}{2})\cdot 4q_{td}(n+1)|+\rho(\lambda) \\
&\leq |(Pr[E_3]-\frac{1}{2})|\cdot 4q_{td}(n+1)+\rho(\lambda) \\
&\leq (|Pr[E_4]-\frac{1}{2}|+Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda))\cdot 4q_{td}(n+1)+\rho(\lambda) \\
&= Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda)\cdot 4q_{td}(n+1)+\rho(\lambda)
\end{aligned}$$

where $\rho(\lambda)=Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda)\cdot q_{td}(n+1)>0$. Thus, we obtain that

$$Adv^{IND-KGA}_{CBEKS,\mathscr{A}}(\lambda)\leq Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda)\cdot 5q_{td}(n+1).$$

Proof of the Collusion Resistance Security

Theorem. Suppose that the s-DBDHE assumption holds in $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T)$, then the CBEKS scheme is collusion resistant in the standard model.

Suppose that there exists a PPT adversary $\mathscr{A}$ that can attack the collusion resistance of the CBEKS scheme in the standard model with advantage $Adv^{CollRes}_{CBEKS,\mathscr{A}}(\lambda)\geq\varepsilon$. We build a challenger $\mathscr{B}$ that has advantage at least ε in solving the s-DBDHE problem in $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T)$. $\mathscr{B}$ receives a random s-DBDHE problem instance $(g_1,g_1^b,g_1^a,\cdots,g_1^{a^s},g_1^{a^{s+2}}$, $\cdots,g_1^{a^{2s}},g_2,g_2^b,g_2^a,\cdots,g_2^{a^s},Z)$ where Z is either $e(g_1,g_2)^{b\cdot a^{s+1}}$ or a random element in $\mathbb{G}_T$, such that g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$. We let the challenger proceed as follows.

Initialization. $\mathscr{B}$ receives from $\mathscr{A}$ the group S^* of receivers that the adversary wants to attack.

Setup. $\mathscr{B}$ computes the public parameters *params* and the private keys $sk_{R,i}$ of the receivers in $[1,s]\setminus S^*$ as follows. First, $\mathscr{B}$ chooses at random $\beta\in_R\mathbb{Z}_p$ and computes g_1^{β},g_2^{β}. It then selects at random $u\in\mathbb{Z}_p$ and generates $g_1^u\cdot(\prod_{k\in S^*}g_1^{a^{s+1-k}})^{-1}$. Suppose that it sets this value equal to g_1^{γ} for an unknown γ. It also picks at random $\delta_1,\cdots,\delta_s,\xi,\omega\in_R\mathbb{Z}_p$ and computes $g_1^{\delta_1},\cdots,g_1^{\delta_s},g_2^{\xi},g_1^{\omega}$. In addition, the challenger picks at random $A,B\in_R\mathbb{G}_2$ and chooses a strongly unforgeable one-time signature scheme $\mathsf{OTS}=(\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify})$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params=(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,g_1,$ $g_2,A,B,\{g_1^{a^i}\}_{i\in[1,s]\cup[s+2,2s]},\{g_2^{a^i}\}_{i\in[1,s]},g_2^{\beta},\{g_1^{\delta_i}\}_{i\in[1,s]},g_2^{\xi},g_1^{\omega},\mathsf{OTS})$. Note that since the exponents in $\mathbb{Z}_p$ are uniformly chosen at random, these public parameters have an identical distribution to that in the actual construction. Moreover, the exponent a from the assumption is informally set as equal to the exponent α from the real construction.

In addition, $\mathscr{B}$ forwards $\mathscr{A}$ the private keys of the receivers in $[1,s]\setminus S^*$. For all $i\notin S^*$, the challenger generates the private key $sk_{R,i}$ as $(\delta_i,g_1^{\beta},(g_1^{a^i})^u\cdot(\prod_{k\in S^*}g_1^{a^{s+1-k+i}})^{-1})$.

Note that

$$\begin{aligned}(g_1^{a^i})^u \cdot (\prod_{k \in S^*} g_1^{a^{s+1-k+i}})^{-1} &= (g_1^u \cdot (\prod_{k \in S^*} g_1^{a^{s+1-k}})^{-1})^{a^i} \\ &= (g_1^u \cdot (\prod_{k \in S^*} g_1^{a^{s+1-k}})^{-1})^{a^i} \\ &= (g_1^{\gamma})^{a^i}\end{aligned}$$

as required. Moreover, since $i \notin S^*$, the product defining the third element of i's private key does not include the element $g_1^{a^{s+1}}$. We can so observe that $\mathscr{B}$ has all the necessary values to compute the private keys $sk_{R,i}$ for $i \notin S^*$.

Query Phase 1. $\mathscr{A}$ makes the following queries:

- *First Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the first certificate query generation oracle, then $\mathscr{B}$ picks at random $r_1 \in_R \mathbb{Z}_p$ and computes $e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_1 L_1^R}$ and $g_2^{r_1 \delta_i}$ for $i \in S \subseteq S^*$. It sends these two elements to $\mathscr{A}$ as the first certificate $Cert_{L_1^R,i}$.
- *Refreshed Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries L_j^R to the refreshed certificate generation oracle, then $\mathscr{B}$ picks at random $r_j \in_R \mathbb{Z}_p$ and sends the elements $e(g_1^{\delta_i}, g_2^{\xi})^{\omega r_j L_j^R}$ and $g_2^{\delta_i r_j}$ to $\mathscr{A}$ as the refreshed certificate $Cert_{L_j^R,i}$ for $i \in S \subseteq S^*$.

Challenge. Once the adversary decides that the query phase 1 is over, it outputs a label pair (L_0, L_1). The challenger answers by choosing a random bit $\mu \in_R \{0,1\}$ and by setting the challenge label $L^* = L_\mu$. $\mathscr{B}$ picks at random $r_{j-1}, r_j \in_R \mathbb{Z}_p$ and computes the challenge update key $UK^* = (g_2^b, g_2^{r_j}, (g_1^b)^u,\ Z \cdot e(g_1^{\omega}, g_2^{\xi})^{r_j L^* - r_{j-1}(L^*-1)})$. We denote by $L^* - 1$ the label preceding the label L^*.

Therefore, UK^* is a valid update key to $\mathscr{A}$'s view, by writing $g_2^b = g_2^{s_j}$ for an unknown $s_j \in \mathbb{Z}_p$. Then, we get that $g_1^{bu} = (g_1^u \cdot (\prod_{k \in S} g_1^{a^{s+1-k}})^{-1} \cdot (\prod_{k \in S} g_1^{a^{s+1-k}}))^b = (g_1^u \cdot (\prod_{k \in S} g_1^{a^{s+1-k}})^{-1} \cdot (\prod_{k \in S} g_1^{a^{s+1-k}}))^{s_j}$.

When $Z = e(g_1, g_2)^{ba^{s+1}}$, then UK^* is a valid challenge update key to $\mathscr{A}$ as in the real attack. When Z is random in $\mathbb{G}_T$, then $Z \cdot e(g_1^{\omega}, g_2^{\xi})^{r_j L^* - r_{j-1}(L^*-1)}$ is a uniform element in $\mathbb{G}_T$, and thus the update key gives no information about the challenger's bit μ.

Query Phase 2. $\mathscr{A}$ issues a number of queries as in the query phase 1. The restriction is that $< L_j^R >$ are not allowed to be queried as first certificate or refreshed certificate queries if $< L_j^R > = < L_0 >$ or $< L_j^R > = < L_1 >$.

Guess. The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $\mathscr{B}$ outputs 1 meaning that $Z = e(g_1, g_2)^{b \cdot a^{s+1}}$; otherwise, $\mathscr{B}$ outputs 0 meaning that Z is a random element in $\mathbb{G}_T$.

Analysis. When $Z = e(g_1, g_2)^{b \cdot a^{s+1}}$, then the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$. When $Z \in_R \mathbb{G}_T$, then $Z \cdot e(g_1^{\omega}, g_2^{\xi})^{r_j L^* - r_{j-1}(L^*-1)}$ is uniformly random in $\mathbb{G}_T$, and thus $Pr[\mu' = \mu] = \frac{1}{2}$. It follows that we have $Adv_{\mathscr{B},\mathbb{G}_T}^{s-DBDHE}(\lambda) \geq \varepsilon$.

5.1.6 Additional Proofs

Proof of the Lemma 1.

We first recall the lemma: For every fixed $VIEW_{\mathscr{A}}$, we define $\zeta_{low} = \frac{1}{4(n+1)q_{td}}$ and $\zeta_{up} = \frac{1}{2q_{td}}$. Thus, we have that $\zeta_{low} \leq \tau(VIEW_{\mathscr{A}}) \leq \zeta_{up}$.

We assume that $VIEW_{\mathscr{A}}$ and the queried keywords in $\mathscr{W}^* = \{w^{(1)}, w^{(2)}, \cdots, w^{(q_0)}, w^*\}$ are fixed. Let t be an integer and $F(t)$ the following event:

$$F(t) : \bigwedge_{i=1}^{q_0} (y(w^{(i)}) \neq 0 \mod t) \wedge y(w^*) = 0 \mod t.$$

So, we get that $\tau(VIEW_{\mathscr{A}}) = Pr_{\bar{\mathscr{Y}}}[F(p)]$. Remember that we have $\bar{\mathscr{Y}} = (y_0, y_1, \cdots, y_n, k)$ distributed according to the formula $y(w) = (p - Mk) + y_0 + \sum_{i=1}^n (y_i \cdot w_i)$. Let $y(w^*) = (p - Mk) + y_0 + \sum_{i=1}^n (y_i \cdot w_i^*)$ such that $y_0 + \sum_{i=1}^n (y_i \cdot w_i^*) < (n+1) \cdot M \in [0, p)$. We set $k^* = \lfloor \frac{y_0 + \sum_{i=1}^n (y_i \cdot w_i^*)}{M} \rfloor$, and so $y(w^*) = p + (k^* - k)M$ such that $k, k^* \in [0, n+1)$, or this can be rewritten as $0 \leq (k^* - k) \cdot M < (n+1) \cdot M < p$. Therefore, if $y(w^*) = 0 \mod M$, then there exists a unique $k \in [0, n+1)$ such that $y(w^*) = 0 \mod p$. Moreover, if $k^* = k$ and $y(w^*) = 0 \mod M$, then $y(w^*) = 0 \mod p$. This can be also formulated as if $y(w^*) \neq 0$ mod M, then $y(w^*) \neq 0 \mod p$.

Thus, we obtain the following:

$$\begin{aligned}
\tau(VIEW_{\mathscr{A}}) &\geq Pr[k^* = k] \cdot Pr_{\bar{\mathscr{Y}}}[F(p) | k^* = k] \\
&= \frac{1}{n+1} Pr_{\bar{\mathscr{Y}}}[F(p) | k^* = k] \\
&\geq \frac{1}{n+1} Pr_{\bar{\mathscr{Y}}}[F(M) | k^* = k] \\
&= \frac{1}{n+1} Pr_{\bar{\mathscr{Y}}'}[F(M) | k^* = k]
\end{aligned}$$

where the probability space $\bar{\mathscr{Y}}'$ contains the random variables $(y_0, y_1, \cdots, y_n)$ that are distributed according to the formula $y(w) = (p - Mk) + y_0 + \sum_{i=1}^n (y_i \cdot w_i)$ for a fixed value k. This means that we consider k as a fixed value from this point in the proof. We define τ_M as equal to $Pr_{\bar{\mathscr{Y}}'}[F(M)]$. Since $\tau_M \geq \tau(VIEW_{\mathscr{A}})$, we get that $\frac{1}{n+1}\tau_M \leq \tau(VIEW_{\mathscr{A}}) \leq \tau_M$.

We then have to find an upper and a lower bounds for τ_M. Let $w \neq w'$ be two keywords and C, D be two elements of $\mathbb{Z}$. We show that $y(\cdot)$ and M are pairwise independent as follows:

$$Pr_{\bar{\mathscr{Y}}}[y(w) = D \mod M] = \frac{1}{M} \quad (5.1)$$

$$Pr_{\bar{\mathscr{Y}}}[y(w) = C \mod M | y(w') = D \mod M] = \frac{1}{M} \quad (5.2)$$

The equation 5.1 comes from the fact that, for any choice of $(y_0, y_1, \cdots, y_n, k)$, there is one choice for y_0 that makes the condition to hold. The equation 5.2 follows assuming there is an index $i \in [1, n]$ such that $w_i = 1$ and $w_i' = 0$. By fixing all the values y_j for $j \neq i$, except y_i, so that $y(w') = D$. Thus, we get that $Pr_{\bar{\mathscr{Y}}}[y(w) = C \mod M | y(w') = D \mod M] = \frac{1}{M}$. Note that we can use Bayes to reverse the roles of w and w' if there is no such index i.

Then, we lower bound the value τ_M as follows:

$$\begin{aligned}
\tau_M &= Pr_{\bar{\mathscr{Y}}'}[\bigwedge_{i=1}^{q_0}(y(w^{(i)}) \neq 0 \mod M)|y(w^*) = 0 \mod M] \cdot Pr_{\bar{\mathscr{Y}}'}[y(w^*) = 0 \mod M] \\
&= \frac{1}{M} Pr_{\bar{\mathscr{Y}}'}[\bigwedge_{i=1}^{q_0}(y(w^{(i)}) \neq 0 \mod M)|y(w^*) = 0 \mod M] \\
&= \frac{1}{M}(1 - Pr_{\bar{\mathscr{Y}}'}[\bigvee_{i=1}^{q_0}(y(w^{(i)}) \neq 0 \mod M)|y(w^*) = 0 \mod M]) \\
&\geq \frac{1}{M}(1 - \sum_{i=1}^{q_0} Pr_{\bar{\mathscr{Y}}'}[(y(w^{(i)}) \neq 0 \mod M)|y(w^*) = 0 \mod M]) \\
&= \frac{1}{M}(1 - \sum_{i=1}^{q_0} \frac{1}{M}) \\
&\geq \frac{1}{M}(1 - \frac{q_{td}}{M}) \\
&= \frac{1}{4q_{td}}
\end{aligned}$$

and we upper bound the value τ_M as follows:

$$\begin{aligned}
\tau_M &= \frac{1}{M}(1 - Pr_{\bar{\mathscr{Y}}'}[\bigwedge_{i=1}^{q_0}(y(w^{(i)}) \neq 0 \mod M)|y(w^*) = 0 \mod M]) \\
&\leq \frac{1}{M} \\
&= \frac{1}{2q_{td}}
\end{aligned}$$

such that the last equality comes from the definition of the value $M = 2q_{td}$. Therefore, we get that

$$\frac{1}{4(n+1)q_{td}} \leq \frac{1}{n+1}\tau_M \leq \tau(VIEW_{\mathscr{A}}) \leq \tau_M \leq \frac{1}{2q_{td}}.$$

Proof of the Claim 1.

We first recall the claim: We define $\rho(\lambda) = Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda) \cdot q_{td}(n+1) > 0$. If the experiment takes $s_0(\lambda) = O(n^2(\rho(\lambda))^{-2}\log((nq_{td} \cdot \rho(\lambda))^{-1}))$ samples when computing the estimate $\tau'(VIEW_{\mathscr{A}})$, then $|Pr[E_1] - (\frac{1}{2} + (Pr[E_2] - \frac{1}{2}) \cdot 4q_{td}(n+1))| \leq \rho(\lambda)$.

Let *AAbort* be the event that the experiment aborts at the end of the simulation in an artificial way. Let the total of aborts $TAbort = FAbort \vee AAbort$ be the event that the experiment either aborts in a forced way or aborts in an artificial way. Let us present another claim:

Claim 2. For any fixed value $VIEW_{\mathscr{A}}$, we get $|Pr[\neg TAbort] - \zeta_{low}| \leq (\zeta_{low} \cdot \rho(\lambda))/2$.

We give the proof of the above claim after the proof of the claim 1. The claim 2 holds for any fixed value $VIEW_{\mathscr{A}}$, thus it also holds for random values $VIEW_{\mathscr{A}}$ regarding either

$\mu' = \mu$ and $\mu' \neq \mu$.

$$\begin{aligned}|Pr[\neg TAbort|\mu' = \mu] - \zeta_{low}| &\leq (\zeta_{low} \cdot \rho(\lambda))/2 &(5.3)\\ |Pr[\neg TAbort|\mu' \neq \mu] - \zeta_{low}| &\leq (\zeta_{low} \cdot \rho(\lambda))/2 &(5.4)\end{aligned}$$

Therefore, we get that

$$\begin{aligned}Pr[E_2] &= Pr[\nu' = 1 \wedge TAbort] + Pr[\nu' = 1 \wedge \neg TAbort]\\ &= Pr[\nu' = 1|TAbort](1 - Pr[\neg TAbort]) + Pr[\nu' = 1 \wedge \neg TAbort]\end{aligned}$$

We recall that the challenger outputs a random bit ν' in case of an abort. If the challenger does not abort, then it outputs a bit $\nu' = 1$ if $\mu' = \mu$. Thus, we obtain that

$$\begin{aligned}Pr[E_2] &= Pr[\nu' = 1|TAbort](1 - Pr[\neg TAbort]) + Pr[\nu' = 1 \wedge \neg TAbort]\\ &= \frac{1}{2}(1 - Pr[\neg TAbort]) + Pr[\mu' = \mu \wedge \neg TAbort]\\ &= \frac{1}{2}(1 - (Pr[\mu' \neq \mu \wedge \neg TAbort] + Pr[\mu' = \mu \wedge \neg TAbort]))\\ &\quad + Pr[\mu' = \mu \wedge \neg TAbort]\\ &= \frac{1}{2} + \frac{1}{2}(Pr[\mu' = \mu \wedge \neg TAbort] - Pr[\mu' \neq \mu \wedge \neg TAbort])\\ &= \frac{1}{2} + \frac{1}{2}(Pr[\neg TAbort|\mu' = \mu]Pr[\mu' = \mu] - Pr[\neg TAbort|\mu' \neq \mu]Pr[\mu' \neq \mu])\end{aligned}$$

Since $Pr[\mu' = \mu] = Pr[E_1]$, we get that

$$\begin{aligned}Pr[E_2] - \frac{1}{2} &= \frac{1}{2}(Pr[\neg TAbort|\mu' = \mu] \cdot Pr[E_1] - Pr[\neg TAbort|\mu' \neq \mu] \cdot (1 - Pr[E_1]))\\ &= \frac{1}{2}Pr[E_1] \cdot (Pr[\neg TAbort|\mu' = \mu] + Pr[\neg TAbort|\mu' \neq \mu])\\ &\quad - \frac{1}{2}Pr[\neg TAbort|\mu' \neq \mu]\end{aligned}$$

By combining the two equations 5.3 and 5.4, we finally obtain that

$$\begin{aligned}Pr[E_2] - \frac{1}{2} - \zeta_{low}(Pr[E_1] - \frac{1}{2}) &= \frac{1}{2}Pr[E_1] \cdot ((Pr[\neg TAbort|\mu' = \mu] - \zeta_{low})\\ &\quad + (Pr[\neg TAbort|\mu' \neq \mu] - \zeta_{low}))\\ &\quad - \frac{1}{2}(Pr[\neg TAbort|\mu' \neq \mu] - \zeta_{low})\end{aligned}$$

and so,

$$\begin{aligned}|Pr[E_2] - \frac{1}{2} - \zeta_{low}(Pr[E_1] - \frac{1}{2})| &\leq \frac{1}{2}Pr[E_1] \cdot (\frac{\zeta_{low} \cdot \rho(\lambda)}{2} + \frac{\zeta_{low} \cdot \rho(\lambda)}{2}) + \frac{\zeta_{low} \cdot \rho(\lambda)}{2}\\ &\leq \zeta_{low} \cdot \rho(\lambda)\end{aligned}$$

such that the last equality comes from the fact that $Pr[E_1] \in [0, 1]$.

Proof of the Claim 2.

To prove this claim, we first recall a lemma due to Hoeffding [114]:

Lemma 2. Let $X_1, X_2, \cdots, X_{s_0}$ be independent random variables with $X_i \in [A,B]$ and let $\bar{X} = \frac{1}{s_0}\sum_{i=1}^{s_0} X_i$. Therefore, for any value $t > 0$, we have the following inequality:

$$Pr[|\bar{X} - E(\bar{X})| \geq t] \leq 2e^{-2s_0(\frac{t}{B-A})^2}$$

such that $E(\bar{X})$ denotes the expected values of $\bar{X}$.

Observe that by construction, the events *AAbort* and *FAbort* are independent, and so we get that $Pr[\neg TAbort] = Pr[\neg FAbort] \cdot Pr[\neg AAbort] = \tau(VIEW_{\mathscr{A}})Pr[\neg AAbort]$. Let $\rho'(\lambda) = \frac{1}{8}\rho(\lambda) \in (0, \frac{1}{8}]$. We obtain $s_0(\lambda)$ samples such that

$$\begin{aligned} s_0(\lambda) &= 8q_{td}^{-2}(\rho(\lambda)\zeta_{low})^{-2}\log((\frac{1}{16}\rho(\lambda)\zeta_{low})^{-1}) \\ &= O(n^2(\rho(\lambda))^{-2}\log((nq_{td}\cdot\rho(\lambda))^{-1})) \end{aligned}$$

in order to calculate an approximation $\tau'(VIEW_{\mathscr{A}})$ of $\tau(VIEW_{\mathscr{A}})$. For each sample that is independently chosen regarding the distribution $\mathscr{Y}$ from the equality $y(w) = (p - Mk) + y_0 + \sum_{i=1}^{n}(y_i \cdot w_i)$, we define each indicator variable X_i as follows: if the event *FAbort* occurs, then $X_i = 0$; otherwise, $X_i = 1$.

Note that $\zeta_{low} \leq Pr[X_i = 1] \leq \zeta_{up}$ by construction. Eventually, we make a majority decision over all the terms X_i by computing $\tau'(VIEW_{\mathscr{A}}) = \frac{1}{s_0(\lambda)}\sum_{i=1}^{s_0(\lambda)}(X_i)$. Observe that $E(\tau'(VIEW_{\mathscr{A}})) = \tau(VIEW_{\mathscr{A}}) \geq \zeta_{low}$ by construction. Using the bound due to Hoeffding [114, 133] for the estimation $\tau'(VIEW_{\mathscr{A}})$ of $\tau(VIEW_{\mathscr{A}})$ such that $D - C \leq \zeta_{up} - \zeta_{low} \leq \frac{1}{2q_{td}}$, we obtain that

$$\begin{aligned} & Pr[|\tau'(VIEW_{\mathscr{A}}) - \tau(VIEW_{\mathscr{A}})| \geq \tau(VIEW_{\mathscr{A}})\rho'(\lambda)] \\ \leq\ & 2\exp(-2s_0(\lambda)(\frac{T}{s_0(\lambda)})^2) \\ \leq\ & 2\exp(-2s_0(\lambda)(\frac{\zeta_{low}\cdot\rho(\lambda)\cdot q_{td}}{s_0(\lambda)})^2) \\ \leq\ & 2\exp(-2\cdot 8q_{td}^{-2}(\rho(\lambda)\zeta_{low})^{-2}\log((\frac{1}{16}\rho(\lambda)\zeta_{low})^{-1})(\frac{\zeta_{low}\cdot\rho(\lambda)\cdot q_{td}}{4})^2) \\ \leq\ & \frac{\zeta_{low}\cdot\rho(\lambda)}{8} \end{aligned}$$

where $T = \tau(VIEW_{\mathscr{A}})\rho(\lambda)\cdot 2q_{td}$ and $\rho'(\lambda) = \rho(\lambda)/8$. Let the approximation $\tau'(VIEW_{\mathscr{A}})$ be seen as $(good)$ if $|\tau'(VIEW_{\mathscr{A}}) - \tau(VIEW_{\mathscr{A}})| \leq \frac{\tau(VIEW_{\mathscr{A}})\rho(\lambda)}{8}$ and as (bad) otherwise. Therefore, we get that $Pr[\tau'(VIEW_{\mathscr{A}})^{(good)}] \leq \zeta_{low}\cdot\rho'(\lambda)$. We now fix $\tau'(VIEW_{\mathscr{A}})$ as $(good)$, written as $\tau'(VIEW_{\mathscr{A}})^{(good)}$, and so

$$\tau(VIEW_{\mathscr{A}})(1 - \rho'(\lambda)) \leq max\{\zeta_{low}, \tau'(VIEW_{\mathscr{A}})\} \leq \tau(VIEW_{\mathscr{A}})(1 + \rho'(\lambda)).$$

Observe that $Pr[\neg AAbort] = \frac{\zeta_{low}}{max\{\zeta_{low}, \tau'(VIEW_{\mathscr{A}})\}}$, and thus

$$\frac{\zeta_{low}}{\tau(VIEW_{\mathscr{A}})(1 + \rho'(\lambda))} \leq Pr[\neg AAbort|\tau'(VIEW_{\mathscr{A}})^{(good)}] \leq \frac{\zeta_{low}}{\tau(VIEW_{\mathscr{A}})(1 - \rho'(\lambda))}.$$

We complete the proof by giving an upper bound on $Pr[\neg TAbort]$ as follows

$$\begin{aligned}
Pr[\neg TAbort] &= \tau(VIEW_{\mathscr{A}})(Pr[\neg AAbort|\tau'(VIEW_{\mathscr{A}})^{(good)}]\cdot Pr[\tau'(VIEW_{\mathscr{A}})^{(good)}] \\
&\quad +Pr[\neg AAbort|\tau'(VIEW_{\mathscr{A}})^{(bad)}]\cdot Pr[\tau'(VIEW_{\mathscr{A}})^{(bad)}]) \\
&\leq \tau(VIEW_{\mathscr{A}})(\frac{\zeta_{low}}{\tau(VIEW_{\mathscr{A}})(1-\rho'(\lambda))}+\zeta_{low}\cdot\rho'(\lambda)) \\
&\leq \zeta_{low}(\frac{1}{\tau(VIEW_{\mathscr{A}})(1-\rho'(\lambda))}+\rho'(\lambda)) \\
&\leq \zeta_{low}(1+4\rho'(\lambda)) \\
&= \zeta_{low}(1+(\rho(\lambda)/2))
\end{aligned}$$

5.1.7 Performance

Our CBEKS scheme remains efficient and practical since the size of the ciphertexts and the trapdoors is constant. In addition, the private keys of the involved entities (namely, the receivers, the certifier and the server) have constant size. Although the public parameters have a size linear in the number s of receivers, we observe that these elements are set up only once, at the beginning of the protocol. Therefore, this is not cumbersome for our scheme.

We evaluate the efficiency of our scheme CBEKS in Tables 5.2 and 5.3. We consider cryptographic operations such as group multiplication and group division on random group elements, group exponentiation of a random group element with a random exponent, as well as pairing operation on random group elements. We use implementation results of these operations in the CHARM framework [4, 152] based on Miyaji-Nakabayashi-Takano (MNT) [162] curves. The benchmarks were executed on a quad core processor AMD A10-5800K with 16GB of RAM running Fedora 18.

We evaluated the protocol algorithms on the asymmetric bilinear group based on the MNT elliptic curve provided by CHARM. Complex multiplication is required in order to find curves of a certain embedding degree with a specified order. In particular, MNT curves have an embedding degree $k=6$.

We assume that there are $s=100$ receivers and there are $n=48$ bits coding a keyword string $w=(w_1,\cdots,w_n)\in\{0,1\}^n$ with 6 characters, such as "urgent".

	Expon. in $\mathbb{G}_1$	Expon. in $\mathbb{G}_2$	Expon. in $\mathbb{G}_T$	Pairing
Time/operation	0.5895	5.314	1.1	3.798
Setup	178.029	542.028		
Encrypt	30.0645	15.942		
CertGen	117.9	1062.8		379.8
UpdtKeyGen	2.9475	10.628		11.394
UpdtCert	0.5895	5.314		11.394
TrapGen	2.9475	5.314		7.596
Test	0.5895	5.314	1.1	7.596

Table 5.2: Timings for our CBEKS asymmetric pairing-based system. Times are in milliseconds. Expon. refers to the group exponentiation operation.

	Multip. in $\mathbb{G}_1$	Multip. in $\mathbb{G}_T$	Divis. in $\mathbb{G}_T$
Time/operation	0.002558	0.006992	0.02108
Setup			
Encrypt			
CertGen			
UpdtKeyGen	0.258358	0.006992	0.02108
UpdtCert	0.2558	0.006992	0.02108
TrapGen	0.002558	0.006992	
Test		0.006992	0.02108

Table 5.3: Timings for our CBEKS asymmetric pairing-based system based. Times are in milliseconds. Multip. refers to the multiplication operation and Divis. refers to the division operation.

We note that the total time in the algorithm Setup is substantial; however this algorithm should be run only once to generate the public parameters and the static private keys for all the receivers, certifiers and server. The algorithm Encrypt does not require too many computations from the uploader: the time is mainly a consequence from the fact that the keyword's elements have to be hashed. Then, the algorithm CertGen is significant since we consider the generation of 100 first certificates; nevertheless, the same argument from Setup applies for CertGen, i.e. first certificates are generated only once. Both the algorithms UpdtKeyGen and UpdtCert are executed fast, meaning that the certifier and the receivers can easily create update keys and update the certificates, respectively. The algorithm TrapGen is relatively efficient, meaning that a receiver can conveniently request for a medical document. Finally, the total time for running the algorithm Test is mainly due to the cost of pairing computations, and remains quick.

5.1.8 Observations

We recall that we see a label as a reference to some information about the receivers' access rights. More precisely, a label is linked to a collection of rights under the form of a unique number in $\mathbb{Z}_p$. We have to ensure that two labels are distinct if they do not refer to the same access right collection.

We argue that the certifier and the server know the labels and agree on their use by securily exchanging information among them, while none of the receivers should be aware of the labels' contents. Nevertheless, defining unique labels could turn to be a bottleneck. A more effective solution will embed time periods instead of labels. Indeed, the certifier and the server will proceed by using the current time period: the former will generate the first certificates and update keys by taking as input the current time period, while the latter will check the keyword and the validity of a receiver's certificate by using the ciphertext and the current time period. However, defining time periods does not seem straightforward at first sight. In fact, we have to ensure that a time period is unique given a starting date and an ending date and that the receivers are not able to modify their certificates themselves in order to make them valid. Moreover, a receiver's certificate can be valid for a longer time period than the one used during the server's test; hence, we have to find a way to make time periods matching as long as the time period used by the server is strictly included in the time period embedded into the receiver's certificate.

We also observe that we give power to the server by letting it know and use labels for verification processes. One might prefer instead to let the uploader decide the choices of both the keyword and the label (or the time period according to the above remarks). Hence, in this case, the server would have just to check that keywords and labels match without being aware of any pieces of information from these components.

5.1.9 Conclusion

In this section, we introduced the new primitive called certificate-based encryption with keyword search (CBEKS) in order to tackle the problem of authorising receivers in a sensitive environment to let them access and retrieve medical documents securely. We constructed a scheme and proved it secure in the standard model, according to the security models that we carefully defined. More precisely, our CBEKS construction is computationally consistent, IND-CKCA secure, IND-KGA secure and collusion resistant in the standard model.

Chapter 6

Ensuring Patient Privacy and Saving Computational and Communication Resources in Electronic Health Records-based Storage

Electronic health records (EHRs) contain sensitive personal information on patients' identity and information contents. One main concern is to ensure patients' privacy regarding non-authorised users. More precisely, only a hospital staff member who is granted with the right to access patients' EHRs knows their identities and information contents, whereas everyone else should acquire no information. In addition, by guaranteeing privacy, we should avoid to increase computational and communication burdens. We recall that the involved parties, namely patients and hospital staff members, have limited technology and storage resources.

We propose two primitives to enable privacy in EHRs. A first one allows a user to use the recipient's credentials to encrypt medical information. A cloud server, acting as a proxy, is required to re-encrypt the encrypted documents regarding the credentials of another recipient (when the first beneficiary is no longer available for instance). We require that the proxy receives no information about the identities and the information contents of the two beneficiaries. In addition of the privacy-preserving feature, the primitive enables a user to perform the main workload offline (for the first encryption before forwarding to the cloud server), and finalise the process online (for the second encryption done by the cloud server).

A second one permits a user to use some personal characteristics to encrypt delicate medical information. In particular, we focus on the DNA properties: the DNA sequences are shared between relatives, and such feature helps us to build our protocol. Observe that we use features already existing in the nature, and thus simple and effortless to use. A cloud server, acting as a proxy, participates into the process as the entity releasing the encrypted documents at a given time (that the encryptor decided beforehand). We require that this proxy gets no information about neither the sender nor the recipient.

6.1 On-line/Off-line Ciphertext-Policy Attribute-Based Proxy Re-Encryption

We now focus on the e-Health environment through a hospital staff member oriented approach. The EHRs could be seen as the solution for better management of an individual's health, and as the tool empowering the hospital staff members by giving them the competences to access the medical history of the patients.

Hospital staff member access to the patients' records is important while patients' privacy regarding the sensitive information that they contain, such as the diagnosis, prognosis or implications of diagnostic tests, should be maintained. We are motivated to empower EHRs with the emerging cloud computing solution. Especially, we would like to equip hospital staff members with the ability to control the access to EHRs efficiently and easily, while they are away from their desktop computing environment.

Let us consider the following scenario. A patient, Alice, comes to the hospital to have her blood sampled for a test. Later, the clinical laboratory encrypts the result of this blood test and uploads it on the hospital's private cloud server. Note that the cloud server is not able to decrypt this ciphertext without the clinical laboratory's private key, which is inaccessible by the cloud server. Technically, we can view this ciphertext as an *intermediate ciphertext*.

Now that the blood test result is ready to collect, Alice should take an appointment with a general practitioner (GP) of her choice, as well as her possible dates, to discuss about the blood test result. To do so, she creates a re-encryption key based on her requirements and provides it to the cloud server. Note that this re-encryption key does not enable the cloud server to decrypt the message, but rather lets the cloud server to "re-encrypt" the intermediate ciphertext such that the resulting ciphertext is decryptable only by the requested GP, Greg. Thus, the policy that Alice constructs can be something like $\{$Greg $\wedge$ from Monday 12/09/2016 to Friday 16/09/2016$\}$ where the dates specify the time range that allows the medical practitioner to retrieve the blood test result. Subsequently, the cloud server re-encrypts the intermediate ciphertext under Alice's policy and keeps it securely on its storage. Only Greg is able to decrypt this ciphertext in the determined time range. This scenario is depicted in Figure 6.1.

Moreover, we assume that Alice does not need big computational and communication resources to send her requirements to the cloud server in order to finalise the encryption of her blood test result. For instance, we suppose that Alice can proceed on-line, using her smartphone connected to the hospital network. To enable such property, we require that much of the encryption of her blood test result is done off-line by the clinical laboratory, that holds the necessary technical resources.

6.1.1 Contributions

Motivated by the above scenario, we propose a new primitive called on-line/off-line ciphertext attribute-based proxy re-encryption (OO-CP-AB-PRE). We carefully define this cryptosystem along with the related security models. Based on the techniques for OO-CP-ABE systems [115] and the methods for CP-AB-PRE systems [145], we provide a single-hop unidirectional OO-CP-AB-PRE scheme supporting monotonic access policy and dealing with on-line/off-line encryption. We then show that the OO-CP-AB-PRE construction is selectively IND-CCA secure and selectively collusion resistant in the random oracle model.

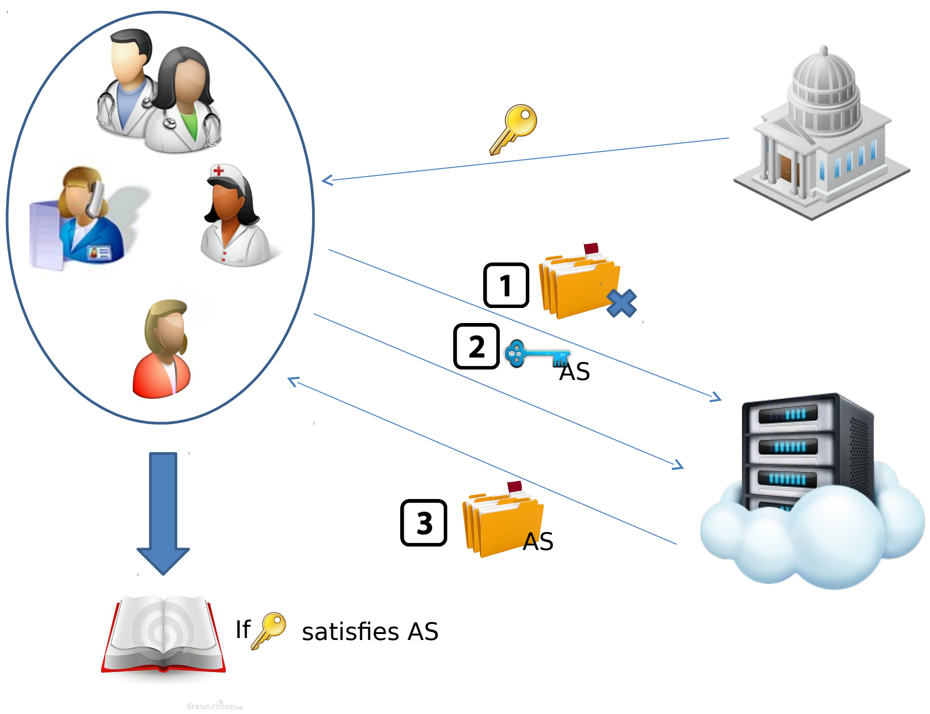


Figure 6.1: On-line/Off-line Ciphertext-Policy Attribute-Based Proxy Re-Encryption. The medical institute, acting as a certifier, generates the public and private key pairs for the hospital staff members. An intermediate cipertext *IntC* is computed by one of the hospital staff member as an encryption of a medical document *m* and regarding the policy defined by this member. This intermediate ciphertext *IntC* is sent to the cloud server. A re-encryption key *rk* is generated by another user (either a hospital staff member or a patient) according to another policy, and is also sent to the cloud server. Finally, the cloud server computes the final ciphertext *C* using *IntC* and *rk*, and makes it available for all the hospital staff members. A hosptial staff member can retrieve *m* if and only if his/her credentials satisfy the policy embedded in *rk* and so *C*.

6.1.2 Protocol Definition

An on-line/off-line ciphertext-policy attribute-based proxy re-encryption (OO-CP-AB-PRE) scheme is composed of the following six algorithms:

- $\mathsf{Setup}(\lambda, \mathscr{U}) \to (params, msk)$. On inputs a security parameter λ and an attribute universe $\mathscr{U}$, output the public parameters *params* and the master secret key *msk*.
- $\mathsf{KeyGen}(params, msk, S) \to sk_S$. On inputs the public parameters *params*, the master secret key *msk* and an attribute set S, output a private key sk_S, such that sk_S is associated with the attribute set S.

  Note that sk_S may contain a description of the attribute set S.
- $\mathsf{OffEncrypt}(params, m) \to IntC$. On inputs the public parameters *params* and a message m, output an intermediate ciphertext *IntC*.
- $\mathsf{ReKeyGen}(params, S, sk_S, \mathbb{AS}) \to rk_{S\to\mathbb{AS}}$. On inputs the public parameters *params*, a attribute set S, a private key sk_S associated with S, and an access structure $\mathbb{AS}$ for attributes over $\mathscr{U}$, output a re-encryption key $rk_{S\to\mathbb{AS}}$ that can be used to transform an intermediate ciphertext to a ciphertext under $\mathbb{AS}$, such that the attribute S satisfies the access structure $\mathbb{AS}$.

Note that $rk_{S\rightarrow\mathbb{AS}}$ is associated with the attribute set S and may contain a description of it.

- $\mathsf{OnEncrypt}(params, rk_{S\rightarrow\mathbb{AS}}, IntC) \rightarrow C_{\mathbb{AS}}$. On inputs the public parameters *params*, a re-encryption key $rk_{S\rightarrow\mathbb{AS}}$ and an intermediate ciphertext *IntC*, output a ciphertext $C_{\mathbb{AS}}$ under $\mathbb{AS}$.

  We assume that the access structure $\mathbb{AS}$ is included in the ciphertext $C_{\mathbb{AS}}$.

- $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) \rightarrow m/\perp$. On inputs the public parameters *params*, an attribute set S, the corresponding private key sk_S and a ciphertext $C_{\mathbb{AS}}$, output m if S satisfies $\mathbb{AS}$; output $\perp$ otherwise, indicating either $C_{\mathbb{AS}}$ is invalid or S does not satisfy $\mathbb{AS}$.

Correctness. We require that an on-line/off-line ciphertext-policy attribute-based proxy re-encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda, \mathscr{U})$, $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$, $IntC \leftarrow \mathsf{OffEncrypt}(params, m)$, $rk_{S\rightarrow\mathbb{AS}} \leftarrow \mathsf{ReKeyGen}(params, S, sk_S, \mathbb{AS})$ and $C_{\mathbb{AS}} \leftarrow \mathsf{OnEncrypt}(params, rk_{S\rightarrow\mathbb{AS}}, IntC)$, and if S satisfies $\mathbb{AS}$, we have that $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) = m$.

N.B. Note that in our protcol defintion, the output $rk_{S\rightarrow\mathbb{AS}}$ of the algorithm ReKeyGen is not a conventional re-encryption key. Indeed, in a classic CP-AB-PRE system, the re-encryption key serves to transform a ciphertext under $\mathbb{AS}'$ to another ciphertext under $\mathbb{AS}$ such that S satisfies $\mathbb{AS}'$ and the two access structures $\mathbb{AS}$ and $\mathbb{AS}'$ are disjoint. Observe that in our case, $rk_{S\rightarrow\mathbb{AS}}$ can be used to transform an intermediate ciphertext to a ciphertext under $\mathbb{AS}$, such that the attribute S satisfies the access structure $\mathbb{AS}$, since the intermediate ciphertext is not linked to any access structure.

6.1.3 Security Models

Selective Indistinguishability against Chosen-Ciphertext Attacks

In the following, we define the security notion for selective indistinguishability against chosen-ciphertext attacks (IND-CCA). An OO-CP-AB-PRE scheme is selectively IND-CCA secure if no PPT adversary $\mathscr{A}$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}$ as follows:

Initialization. The adversary $\mathscr{A}$ outputs a challenge access structure $\mathbb{AS}^*$ to the challenger $\mathscr{B}$.

Setup. The challenger $\mathscr{B}$ runs the algorithm Setup and gives the public parameters *params* to the adversary $\mathscr{A}$.

Query Phase 1. The adversary $\mathscr{A}$ is given access to the following oracles:

1. *Private Key Query* $< S >$. On input an attribute set S, the challenger $\mathscr{B}$ runs $\mathsf{KeyGen}(params, msk, S)$ to obtain sk_S and returns it to the adversary.
2. *Re-Encryption Key Query* $< S, \mathbb{AS} >$. On inputs an attribute set S and an access structure $\mathbb{AS}$, $\mathscr{B}$ runs $\mathsf{ReKeyGen}(params, S, sk_S, \mathbb{AS})$ to obtain $rk_{S\rightarrow\mathbb{AS}}$ and returns it to $\mathscr{A}$, where $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$.

3. *Online Encryption Query* $< S, \mathbb{AS}, IntC >$. On inputs an attribute set S, an access structure $\mathbb{AS}$ and an intermediate ciphertext $IntC$, $\mathscr{B}$ runs $\mathsf{OnEncrypt}(params, rk_{S\to\mathbb{AS}}, IntC)$ to obtain $C_{\mathbb{AS}}$ and returns it to $\mathscr{A}$, where $rk_{S\to\mathbb{AS}} \leftarrow \mathsf{ReKeyGen}(params, S, sk_S, \mathbb{AS})$ and $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$.
4. *Ciphertext Decryption Query* $< S, C_{\mathbb{AS}} >$. On inputs an attribute set S and a ciphertext $C_{\mathbb{AS}}$, the challenger $\mathscr{B}$ returns $m \leftarrow \mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}})$ to the adversary $\mathscr{A}$, where $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$ and S satisfies $\mathbb{AS}$.

Note that if the queries to the ciphertext decryption oracle are invalid, then the challenger simply outputs $\perp$. In this phase, the following queries are forbidden to issue: private key queries for any S satisfying $\mathbb{AS}^*$ and re-encryption key queries for any S satisfying $\mathbb{AS}^*$.

Challenge. The adversary $\mathscr{A}$ submits two messages m_0 and m_1 of equal length. The challenger $\mathscr{B}$ chooses a random bit $\mu \in_R \{0,1\}$ and encrypts m_μ under $\mathbb{AS}^*$. The ciphertext $C^*_{\mathbb{AS}^*}$ is given to the adversary $\mathscr{A}$.

Query Phase 2. The query phase 1 is repeated except that the following queries are forbidden to issue:

- private key queries for any S satisfying $\mathbb{AS}^*$,
- re-encryption key queries for any S satisfying $\mathbb{AS}^*$,
- online encryption queries for any invalid intermediate ciphertext or invalid ciphertext where S satisfies $\mathbb{AS}^*$,
- ciphertext decryption queries for any invalid ciphertext or $C_{\mathbb{AS}} = C^*_{\mathbb{AS}^*}$, where S satisfies $\mathbb{AS}^*$.

Guess. The adversary $\mathscr{A}$ outputs a guess $\mu' \in \{0,1\}$ for μ. If $\mu' = \mu$, then the adversary $\mathscr{A}$ wins.

The advantage of the adversary $\mathscr{A}$ is defined as $Adv^{S-IND-CCA}_{OO-CP-AB-PRE,\mathscr{A}}(\lambda, \mathscr{U}) = |Pr[\mu' = \mu] - 1/2|$.

Selective Collusion Resistant

We consider insider attacks which deal with selective collusion resistance. The collusion resistance is also known as the master key security in the literature [52, 145, 146, 149].

In this security model, we omit the descriptions of the online encryption and ciphertext decryption oracles since they are straightforward: any re-encryption key could be correctly generated from the re-encryption key oracle.

In the following, we define the security notion for selective collusion resistance. An OO-CP-AB-PRE scheme is selectively collusion resistant if no PPT adversary $\mathscr{A}$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}$ as follows:

Initialization. $\mathscr{A}$ outputs an attribute set S^* to $\mathscr{B}$.

Setup. $\mathscr{B}$ runs the algorithm Setup and gives the public parameters *params* to $\mathscr{A}$.

Query Phase. $\mathscr{A}$ is given access to the following oracles:

1. *Private Key Query* $< S >$. On input an attribute set $S \neq S^*$, $\mathscr{B}$ runs $\mathsf{KeyGen}(params, msk, S)$ to obtain sk_S and returns it to $\mathscr{A}$.
2. *Re-Encryption Key Query* $< S, \mathbb{AS} >$. On inputs an attribute set S and an acces structure $\mathbb{AS}$, $\mathscr{B}$ runs $\mathsf{ReKeyGen}(params, S, sk_S, \mathbb{AS})$ to obtain $rk_{S \to \mathbb{AS}}$ and returns it to $\mathscr{A}$, where $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$.

Output. $\mathscr{A}$ submits a private key $sk^*_{S^*}$ for the attribute set S^*.

The advantage of $\mathscr{A}$ is defined as $Adv^{S-CollRes}_{OO-CP-AB-PRE,\mathscr{A}}(\lambda, \mathscr{U}) = Pr[\mathscr{A}$ succeeds$]$.

6.1.4 Construction

Our scheme is based on the OO-CP-ABE system developped by Hohenberger and Waters [115] and the CP-AB-PRE system proposed by Liang et al. [145]. One can now create an intermediate ciphertext during the off-line phase and then translate it to a ciphertext for a hitherto unknown access structure, under a re-encryption key generated from the sender's private key. The Hohenberger and Waters' scheme [115] is based on the unbounded CP-ABE construction implemented by Rouselakis and Waters [195]. We assume the existence of a bound P on the maximum number of rows in the LSSS access structure that will be used to encrypt messages in our construction.

Following Boneh and Boyen's IBE system [36], we proceed as follows during the off-line phase. A ciphertext is generated by encrypting a message using an exponent $x \in_R \mathbb{Z}_p$ chosen at random with another random element $c \in_R \mathbb{Z}_p$. This ciphertext has elements $C_1 = g^c$ and $C_2 = (u^x h)^c$. Moreover, the public parameters are a description of the bilinear group $\mathbb{G}_1$ of order p, along with the tuple $(g, u, h, e(g,g)^\alpha)$, and the encapsulated key is $K = e(g,g)^{\alpha c}$. The intermediate ciphertext $IntC = (C_1, C_2, x, c)$ is stored by the off-line algorithm. During the on-line phase, given an identity $id \in \mathbb{Z}_p$, the encryptor adds a "correction factor" equal to $c \cdot (id - x) \in \mathbb{Z}_p$ to the components C_1 and C_2.

- $\mathsf{Setup}(\lambda, \mathscr{U}) \to (params, msk)$. Let λ be the security parameter and $\mathscr{U}$ be the attribute universe viewed as $\mathbb{Z}_p$. Choose two bilinear cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of prime order p, along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Select at random some generators $g, h, u, v, w \in_R \mathbb{G}_1$ and an exponent $\alpha \in_R \mathbb{Z}_p$. Define five hash functions $H_1 : \mathbb{G}_T \to \{0,1\}^{2\lambda}$, $H_2 : \{0,1\}^{2\lambda} \to \mathbb{Z}_p$, $H_3 : \{0,1\}^{\lambda} \to \mathbb{Z}_p$, $H_4 : \{0,1\}^* \to \mathbb{G}_1$ and $H_5 : \{0,1\}^* \to \mathbb{G}_1$.

  Eventually, set the public parameters $params$ as $(p, \mathbb{G}_1, \mathbb{G}_T, e, g, h, u, v, w, e(g,g)^\alpha, H_1, H_2, H_3, H_4, H_5)$ and the master secret key msk as α.

- $\mathsf{KeyGen}(params, msk, S) \to sk_S$. Let $S = \{A_j\}_{j \in [1,k]} \subseteq \mathbb{Z}_p$ be an attribute set. Choose at random values $r, r_1, r_2, \cdots, r_k \in_R \mathbb{Z}_p$. Compute $K_0 = g^\alpha w^r$, $K_1 = g^r$ and $K_{j,2} = g^{r_j}$, $K_{j,3} = (u^{A_j} h)^{r_j} v^{-r}$ for $j \in [1,k]$. The private key sk_S is $(S, K_0, K_1, \{K_{j,2}, K_{j,3}\}_{j \in [1,k]})$.

- $\mathsf{OffEncrypt}(params, m) \to IntC$. We assume there exists a maximum bound of P rows in any LSSS access structure used to generate a (intermediate) ciphertext.

  First, pick at random $\beta \in_R \{0,1\}^\lambda$ and compute $c = H_2(\beta, m)$, $B_1 = (m \| \beta) \oplus H_1(e(g,g)^{\alpha c})$, $C_{0,1} = g^c$ and $C_{0,2} = h^c$. Then, for $j \in [1,P]$, choose at random $\lambda'_j, t_j, x_j \in_R \mathbb{Z}_p$ and compute $C_{j,1} = w^{\lambda'_j} v^{t_j}$, $C_{j,2} = (u^{x_j} h)^{-t_j}$ and $C_{j,3} = g^{t_j}$. This process can be seen as encrypting a random attribute x_j with a random "share" λ'_j of c, and it will be corrected in the on-line phase with the attributes defined in the algorithm ReKeyGen. We notice that the work done in this off-line phase is almost equivalent to the one of the regular encryption algorithm in [195]. Moreover, compute $D = H_4(B_1, C_{0,1}, C_{0,2}, (C_{1,1}, C_{1,2}, C_{1,3}), \cdots, (C_{P,1}, C_{P,2}, C_{P,3}))^c$. Eventually, the intermediate ciphertext $IntC$ is $(c, B_1, C_{0,1}, C_{0,2}, \{\lambda'_j, t_j, x_j, C_{j,1}, C_{2,j}, C_{3,j}\}_{j\in[1,P]}, D)$.

- $\mathsf{ReKeyGen}(params, S, sk_S, (M,\rho)) \rightarrow rk_{S\rightarrow(M,\rho)}$. Let (M,ρ) be a LSSS access structure, where M is a $l\times n$ matrix and $l \leq P$.

  First, pick at random $\tilde{\beta}, \delta \in_R \{0,1\}^\lambda$ and compute $\tilde{c} = H_2(\tilde{\beta}, \delta)$, $\tilde{B}_1 = (\delta\|\tilde{\beta}) \oplus H_1(e(g,g)^{\alpha\tilde{c}})$, and $\tilde{C}_0 = g^{\tilde{c}}$. Select at random $\tilde{y}_2, \cdots, \tilde{y}_n \in \mathbb{Z}_p$, set the vector $\vec{\tilde{y}} = (\tilde{c}, \tilde{y}_2, \cdots, \tilde{y}_n)^T$, where T denotes the transpose of the matrix, and compute a vector of shares of $\tilde{c}$ as $(\tilde{\lambda}_1, \cdots, \tilde{\lambda}_l)^T = M \cdot \vec{\tilde{y}}$. Then, for $j \in [1,l]$, choose at random $\tilde{t}_j \in \mathbb{Z}_p$ and compute $\tilde{C}_{j,1} = w^{\tilde{\lambda}_j} v^{\tilde{t}_j}$, $\tilde{C}_{j,2} = (u^{\rho(j)}h)^{-\tilde{t}_j}$ and $\tilde{C}_{j,3} = g^{\tilde{t}_j}$. Moreover, compute $\tilde{D} = H_5(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M,\rho))^{\tilde{c}}$. Eventually, the intermediate ciphertext $Int\tilde{C}$ for the re-encryption key is $((M,\rho), \tilde{B}_1, \tilde{C}_0, \{\tilde{C}_{j,1}, \tilde{C}_{j,2}, \tilde{C}_{j,3}\}_{j\in[1,l]}, \tilde{D})$.

  Second, select at random an exponent $\theta \in_R \mathbb{Z}_p$ and compute $RK_{0,1} = K_0^{H_3(\delta)} h^\theta$, $RK_{0,2} = g^\theta$, $RK_1 = K_1^{H_3(\delta)}$ and $RK_{j,2} = K_{j,2}^{H_3(\delta)}$, $RK_{j,3} = K_{j,3}^{H_3(\delta)}$, for $j \in [1,k]$.

  The re-encryption key $rk_{S\rightarrow(M,\rho)}$ is $(S, RK_{0,1}, RK_{0,2}, RK_1, \{RK_{j,2}, RK_{j,3}\}_{j\in[1,k]}, Int\tilde{C})$.

- $\mathsf{OnEncrypt}(params, rk_{S\rightarrow(M,\rho)}, IntC) \rightarrow C_{(M,\rho)}$. First, check the validity of the intermediate ciphertext $IntC$ as follows:

$$\begin{aligned} & e(C_{0,1}, h) \stackrel{?}{=} e(g, C_{0,2}) \\ & \forall j \in [1,P], \quad e(C_{j,1}, g) \stackrel{?}{=} e(w,g)^{\lambda'_j} \cdot e(v, C_{j,3}) \\ & \forall j \in [1,P], \quad e(C_{j,2}, g) \stackrel{?}{=} e(u, C_{j,3}^{-1})^{x_j} \cdot e(h, C_{j,3}^{-1}) \\ & e(C_{0,1}, H_4(B_1, C_{0,1}, C_{0,2}, (C_{1,1}, C_{1,2}, C_{1,3}), \cdots, (C_{P,1}, C_{P,2}, C_{P,3}))) \stackrel{?}{=} e(g, D) \end{aligned} \tag{6.1}$$

  and the validity of the re-encryption key $rk_{S\rightarrow(M,\rho)}$ as follows:

$$e(\tilde{C}_0, H_5(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M,\rho))) \stackrel{?}{=} e(g, \tilde{D}).$$

  Then, take the element c from $IntC$ and pick at random $y_2, \cdots, y_n \in_R \mathbb{Z}_p$, set the vector $\vec{y} = (c, y_2, \cdots, y_n)^T$, where T denotes the transpose of the matrix, and compute a vector of shares of c as $(\lambda_1, \cdots, \lambda_l)^T = M \cdot \vec{y}$. Let $I \subset [1,l]$ be defined as $I = \{i : \rho(i) \in S\}$, and $\{w_i\}_{i\in I}$ be a set of constants in $\mathbb{Z}_p$ such that $\sum_{i\in I} w_i \cdot \lambda_i = c$. For $j \in [1,l]$, compute $C_{j,4} = \lambda_j - \lambda'_j$ and $C_{j,5} = t_j \cdot (\rho(j) - x_j)$. Intuitively, this corrects to the proper attributes and shares of c. Finally, compute the value B_2 as the quotient:

$$B_2 = \frac{e(C_{0,1}, RK_{0,1})/e(C_{0,2}, RK_{0,2})}{e(w^{\sum_{i\in I} C_{i,4}\cdot w_i}, RK_1) \cdot \prod_{i\in I}(e(C_{i,1}, RK_1) \cdot e(C_{i,2} \cdot u^{C_{i,5}}, RK_{j,2}) \cdot e(C_{i,3}, RK_{j,3}))^{w_i}},$$

  where j is the index of the attribute $\rho(i)$ in S (it depends on i).

  Eventually, set the ciphertext $C_{(M,\rho)}$ as $((M,\rho), C_{0,1}, B_2, Int\tilde{C})$.

- $\mathsf{Decrypt}(params, S, sk_S, C_{(M,\rho)}) \rightarrow m/\perp$. Parse the ciphertext $C_{(M,\rho)}$ as $((M,\rho), C_{0,1}, B_2, Int\tilde{C})$ for an access structure (M,ρ) and a private key sk_S as $(S, K_0, K_1, \{K_{i,2}, K_{i,3}\}_{i\in[1,k]})$ for an attribute set S. If S does not satisfy (M,ρ), then output $\perp$. Otherwise, proceed as follows. Set $I \subset [1,l]$ as $I = \{i\}_{\rho(i)\in S}$ and find the set of constants $\{\tilde{w}_i\}_{i\in I}$ in $\mathbb{Z}_p$ such that $\sum_{i\in I} \tilde{w}_i \cdot \tilde{\lambda}_i = \tilde{c}$ (let M_i be the i-th row of the matrix M, thus we have that $\sum_{i\in I} \tilde{w}_i \cdot M_i = (1,0,\cdots,0)$). Then, check that:

  $$\begin{aligned} & S \text{ satisifies } (M,\rho)? \\ & e(\tilde{C}_0, H_5(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M,\rho))) \overset{?}{=} e(g, \tilde{D}) \end{aligned} \tag{6.2}$$

  If the above equations hold, then proceed; otherwise, output $\perp$.

  To recover the message m, first recover the value $e(g,g)^{\alpha\tilde{c}}$ by calculating:

  $$\frac{e(\tilde{C}_0, K_0)}{\prod_{i\in I}(e(\tilde{C}_{i,1}, K_1)\cdot e(\tilde{C}_{i,2}, K_{j,2})\cdot e(\tilde{C}_{i,3}, K_{j,3}))^{\tilde{w}_i}}$$

  where j is the index of the attribute $\rho(i)$ in S (it depends on i). Afterwards, compute $H_1(e(g,g)^{\alpha\tilde{c}}) \oplus \tilde{B}_1 = H_1(e(g,g)^{\alpha\tilde{c}}) \oplus (\delta\|\tilde{\beta}) \oplus H_1(e(g,g)^{\alpha\tilde{c}}) = \delta\|\tilde{\beta}$. If $\tilde{C}_0 = g^{H_2(\tilde{\beta},\delta)}$, then proceed; otherwise, output $\perp$. Finally, compute $H_1(B_2^{1/H_3(\delta)}) \oplus B_1 = H_1(e(g,g)^{\alpha\cdot c}) \oplus m\|\beta \oplus H_1(e(g,g)^{\alpha\cdot c}) = m\|\beta$, and if $C_{0,1} = g^{H_2(\beta,m)}$, output m; otherwise, output $\perp$.

Correctness. If the attribute set S of the ciphertext satisfies the access structure (M,ρ), then $\sum_{i\in I} w_i \cdot \lambda_i = c$ and $\sum_{i\in I} \tilde{w}_i \cdot \tilde{\lambda}_i = \tilde{c}$. We recall that $\rho(i) = A_j$. Therefore, the value B_2 is recovered as follows:

$$\begin{aligned} B_2 &= \frac{e(C_{0,1}, RK_{0,1})/e(C_{0,2}, RK_{0,2})}{e(w^{\sum_{i\in I} C_{i,4}\cdot w_i}, RK_1)\cdot\prod_{i\in I}(e(C_{i,1}, RK_1)\cdot e(C_{i,2}\cdot u^{-C_{i,5}}, RK_{j,2})\cdot e(C_{i,3}, RK_{j,3}))^{w_i}} \\ &= \frac{e(C_{0,1}, K_0^{H_3(\delta)} h^{\theta})/e(C_{0,2}, g^{\theta})}{e(w^{\sum_{i\in I} C_{i,4}\cdot w_i}, K_1^{H_3(\delta)})\cdot\prod_{i\in I}(e(C_{i,1}, K_1^{H_3(\delta)})\cdot e(C_{i,2}\cdot u^{-C_{i,5}}, K_{j,2}^{H_3(\delta)})\cdot e(C_{i,3}, K_{j,3}^{H_3(\delta)}))^{w_i}} \\ &= \frac{(e(g^c, g^{\alpha} w^r)^{H_3(\delta)}\cdot e(g^c, h^{\theta}))/e(h^c, g^{\theta})}{\prod_{i\in I}(e(w^{\lambda_i'} v^{t_i}, g^r)\cdot e((u^{x_i}h)^{-t_i}\cdot u^{-t_i(\rho(i)-x_i)}, g^{r_j})\cdot e(g^{t_i}, (u^{A_j}h)^{r_j} v^{-r}))^{H_3(\delta)\cdot w_i}} \\ &\quad \cdot\frac{1}{e(w^{\sum_{i\in I}(\lambda_i-\lambda_i')w_i}, g^r)^{H_3(\delta)}} \\ &= \frac{e(g,g)^{\alpha c\cdot H_3(\delta)}\cdot e(g,w)^{cr\cdot H_3(\delta)}}{\prod_{i\in I}(e(g,w)^{\lambda_i' r} e(g,v)^{t_i r} e(g,u)^{-\rho(i)t_i r_j} e(g,h)^{-t_i r_j} e(g,u)^{A_j t_i r_j} e(g,h)^{t_i r_j} e(g,v)^{-t_i r})^{H_3(\delta)\cdot w_i}} \\ &\quad \cdot\frac{1}{(e(g,w)^{\sum_{i\in I}(\lambda_i-\lambda_i')r})^{H_3(\delta)\cdot w_i}} \\ &= \frac{e(g,g)^{\alpha c\cdot H_3(\delta)}\cdot e(g,w)^{cr\cdot H_3(\delta)}}{(\prod_{i\in I} e(g,w)^{\lambda_i' r}\cdot e(g,w)^{\sum_{i\in I}(\lambda_i-\lambda_i')r})^{H_3(\delta)\cdot w_i}} \\ &= \frac{e(g,g)^{\alpha c\cdot H_3(\delta)}\cdot e(g,w)^{cr\cdot H_3(\delta)}}{e(g,w)^{r\cdot H_3(\delta)\sum_{i\in I}\lambda_i w_i}} = e(g,g)^{\alpha c\cdot H_3(\delta)} \end{aligned}$$

and the value $e(g,g)^{\alpha\tilde{c}}$ is recovered as follows:

$$
\begin{aligned}
& \frac{e(\tilde{C}_0,K_0)}{\prod_{i\in I}(e(\tilde{C}_{i,1},K_1)\cdot e(\tilde{C}_{i,2},K_{j,2})\cdot e(\tilde{C}_{i,3},K_{j,3}))^{\tilde{w}_i}} \\
= & \frac{e(g^{\tilde{c}},g^{\alpha}w^{r})}{\prod_{i\in I}(e(w^{\tilde{\lambda}_i}v^{\tilde{t}_i},g^{r})\cdot e((u^{\rho(i)}h)^{-\tilde{t}_i},g^{r_j})\cdot e(g^{\tilde{t}_i},(u^{A_j}h)^{r_j}v^{-r}))^{\tilde{w}_i}} \\
= & \frac{e(g,g)^{\alpha\tilde{c}}\cdot e(g,w)^{\tilde{c}r}}{(\prod_{i\in I}e(g,w)^{\tilde{\lambda}_i r}e(g,v)^{\tilde{t}_i r}e(g,u)^{-\rho(i)\tilde{t}_i r_j}e(g,h)^{-\tilde{t}_i r_j}e(g,u)^{A_j\tilde{t}_i r_j}e(g,h)^{\tilde{t}_i r_j}e(g,v)^{-\tilde{t}_i r})^{\tilde{w}_i}} \\
= & \frac{e(g,g)^{\alpha\tilde{c}}\cdot e(g,w)^{\tilde{c}r}}{\prod_{i\in I}e(g,w)^{\tilde{\lambda}_i r\tilde{w}_i}} = \frac{e(g,g)^{\alpha\tilde{c}}\cdot e(g,w)^{\tilde{c}r}}{e(g,w)^{r\sum_{i\in I}\tilde{\lambda}_i\tilde{w}_i}} = e(g,g)^{\alpha\tilde{c}}
\end{aligned}
$$

6.1.5 Security Proofs

Selective IND-CCA Security Proof

Theorem. Suppose that the s-type assumption holds in $(\mathbb{G}_1,\mathbb{G}_T)$ and the hash functions H_1, H_2, H_3, H_4 and H_5 are seen as random oracles, then the OO-CP-AB-PRE scheme is selectively IND-CCA secure in the random oracle model.

Suppose that there is an adversary $\mathscr{A}$ who breaks the selective IND-CCA security of the OO-CP-AB-PRE scheme. We then construct a challenger $\mathscr{B}$ to decide whether Z is either equal to $e(g,g)^{a^{s+1}\cdot c}$ or to a random element in $\mathbb{G}_T$. The challenger $\mathscr{B}$ plays the IND-CCA game with $\mathscr{A}$ as follows.

$\mathscr{B}$ takes as inputs $(p,\mathbb{G}_1,\mathbb{G}_T,e)\leftarrow\mathscr{G}(\lambda)$, a generator g of $\mathbb{G}_1$ and a s-type problem instance

$$
\begin{aligned}
& g^{c} \\
& \forall i,j\in[1,s],\quad g^{a^i},g^{b_j},g^{c\cdot b_j},g^{a^i\cdot b_j},g^{a^i/b_j^2} \\
& \forall i\in[1,2s],j\in[1,s],i\neq s+1,\quad g^{a^i/b_j} \\
& \forall i\in[1,2s],j,k\in[1,s],j\neq k,\quad g^{a^i b_j/b_k^2} \\
& \forall i\in[1,2s],j,k\in[1,s],j\neq k,\quad g^{c\cdot a^i b_j/b_k},g^{c\cdot a^i b_j/b_k^2}
\end{aligned}
$$

and Z which is either equal to $e(g,g)^{a^{s+1}\cdot c}$ or to a random element in $\mathbb{G}_T$.

Initialization. The adversary gives the challenge access structure (M^*,ρ^*) to $\mathscr{B}$, where M^* has l rows and n columns such that $l\leq P,s$ and $n\leq s$.

Setup. The challenger chooses at random $\alpha'\in_R\mathbb{Z}_p$ and implicitly sets $\alpha=\alpha'+a^{s+1}$ by letting $e(g,g)^{\alpha}=e(g^a,g^{a^s})\cdot e(g,g)^{\alpha'}$ ($\alpha=\alpha'+a^{s+1}$ cannot be computed by $\mathscr{B}$). We note that the element α is correctly distributed. $\mathscr{B}$ also picks at random

$\hat{v}, \hat{u}, \hat{h} \in_R \mathbb{Z}_p$ and gives the following public parameters to $\mathscr{A}$:

$$\begin{aligned} w &= g^a \\ v &= g^{\hat{v}} \cdot \prod_{\substack{j\in[1,l] \\ k\in[1,n]}} (g^{a^k/b_j})^{M^*_{j,k}} \\ u &= g^{\hat{u}} \cdot \prod_{\substack{j\in[1,l] \\ k\in[1,n]}} (g^{a^k/b_j^2})^{M^*_{j,k}} \\ h &= g^{\hat{h}} \cdot \prod_{\substack{j\in[1,l] \\ k\in[1,n]}} (g^{a^k/b_j^2})^{-\rho^*(j)M^*_{j,k}} \end{aligned}$$

Since a is information-theoretically hidden from $\mathscr{A}$'s view, the value w is properly uniformly random in $\mathbb{G}_1$. The values v, u, h are properly distributed due to $\hat{v}, \hat{u}, \hat{h}$ respectively. We note that all these values are calculated by $\mathscr{B}$ using terms from the assumption instance and from (M^*, ρ^*) given by $\mathscr{A}$.

Then, the challenger chooses the hash functions H_1, H_2, H_3, H_4, H_5 as in the real scheme, and sends the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_T, e, g, h, , u, v, w, e(g,g)^\alpha$, $H_1, H_2, H_3, H_4, H_5)$ to $\mathscr{A}$. We note that the public parameters are identical to those given in the real scheme for the adversary.

At any time, $\mathscr{A}$ can adaptively query the random oracles H_j for $j \in [1,5]$, which are controlled by $\mathscr{B}$ as follows. The challenger maintains the lists L_{H_j}, for $j \in [1,5]$, which are initially empty, and answers the queries to the random oracles as follows.

- H_1: on receipt of an H_1 query on $R \in \mathbb{G}_T$, if there is a tuple $(R, \delta_1) \in L_{H_1}$, $\mathscr{B}$ forwards the predefined value δ_1 to $\mathscr{A}$, where $\delta_1 \in \{0,1\}^{2\lambda}$. Otherwise, $\mathscr{B}$ sets $H_1(R) = \delta_1$, responds δ_1 to $\mathscr{A}$ and adds the tuple (R, δ_1) to L_{H_1}, where $\delta_1 \in_R \{0,1\}^{2\lambda}$.
- H_2: on receipt of an H_2 query on (β, m), if there is a tuple $(\beta, m, c) \in L_{H_2}$, $\mathscr{B}$ forwards the predefined value c to $\mathscr{A}$, where $c \in \mathbb{Z}_p$. Otherwise, $\mathscr{B}$ sets $H_2(\beta, m) = c$, responds c to $\mathscr{A}$ and adds the tuple (β, m, c) to L_{H_2}, where $c \in_R \mathbb{Z}_p$.
- H_3: on receipt of an H_3 query on $\delta \in \{0,1\}^\lambda$, if there is a tuple (δ, ξ_1) in L_{H_3}, $\mathscr{B}$ forwards the predefined value ξ_1 to $\mathscr{A}$, where $\xi_1 \in \mathbb{Z}_p$. Otherwise, $\mathscr{B}$ sets $H_3(\delta) = \xi_1$ to $\mathscr{A}$ and adds the tuple (δ, ξ_1) to L_{H_3}, where $\xi_1 \in_R \mathbb{Z}_p$.
- H_4: on receipt of an H_4 query on $(B_1, C_{0,1}, C_{0,2}, (C_{1,1}, C_{1,2}, C_{1,3}), \cdots, (C_{l,1}, C_{l,2}, C_{l,3}))$, if there is a tuple $(B_1, C_{0,1}, C_{0,2}, (C_{1,1}, C_{1,2}, C_{1,3}), \cdots, (C_{l,1}, C_{l,2}, C_{l,3}), \xi_2, \delta_2) \in L_{H_4}$, $\mathscr{B}$ forwards the predefined value δ_2 to $\mathscr{A}$, where $\xi_2 \in \mathbb{Z}_p, \delta_2 \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ sets $\delta_2 = g^{\xi_2}$, responds δ_2 to $\mathscr{A}$ and adds the tuple $(B_1, C_{0,1}, C_{0,2}, (C_{1,1}, C_{1,2}, C_{1,3}), \cdots, (C_{l,1}, C_{l,2}, C_{l,3}), \xi_2, \delta_2)$ in L_{H_4}, where $\xi_2 \in_R \mathbb{Z}_p$.
- H_5: on receipt of an H_5 query on $(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M, \rho))$, if there is a tuple $(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M, \rho), \xi_3, \delta_3) \in L_{H_5}$, $\mathscr{B}$ forwards the predefined value δ_2 to $\mathscr{A}$, where $\xi_3 \in \mathbb{Z}_p, \delta_3 \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ sets $\delta_3 = g^{\xi_3}$, responds δ_3 to $\mathscr{A}$ and adds the tuple $(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M, \rho), \xi_3, \delta_3)$ in L_{H_5}, where $\xi_3 \in_R \mathbb{Z}_p$.

In addition, $\mathscr{B}$ maintains the lists L_{sk}, L_{rk} and L_{one} which are initially empty as follows:

- L_{sk} records the tuples (S, sk_S) which are the results of the queries to the private key oracle.
- L_{rk} records the tuples $(S, (M,\rho), \delta, \tilde{\beta}, rk_{S\to(M,\rho)}, tag_1, tag_2, tag_3)$ which are the results of the queries to the re-encryption key oracle, where tag_1, tag_2 and tag_3 denote that the re-encryption key is randomly chosen, generated as an online encryption query or generated as a re-encryption key query respectively.
- L_{one} records the tuples $(S, (M,\rho), C_{(M,\rho)}, tag_1, tag_2, tag_3)$ which are the results of the queries to the online encryption oracle, where tag_1, tag_2 and tag_3 denote that the ciphertext is generated under a valid re-encryption key, under a randomly chosen re-encryption key or generated without any re-encryption key respectively.

Query Phase 1. The challenger answers to $\mathscr{A}$'s queries as follows.

- *Private Key Query* $< S >$. $\mathscr{B}$ has to produce private keys for attribute sets that do not satisfy (M^*, ρ^*) requested by $\mathscr{A}$. $\mathscr{B}$ proceeds in creating a key for an attribute set $S = \{A_j\}_{j\in[1,|S|}$ as follows.

  Since S does not satisfy (M^*, ρ^*), there exists a vector $\vec{w} = (w_1, \cdots, w_n)^T \in \mathbb{Z}_p^n$ such that $w_1 = -1$ and $M_i^* \cdot \vec{w} = 0$ for all $i \in I = \{i\}_{\rho^*(i)\in S} \subseteq [1,l]$ (if S satisfies (M^*, ρ^*), then the challenger outputs a random bit in $\{0,1\}$ and aborts the simulation). $\mathscr{B}$ then picks at random $\hat{r} \in_R \mathbb{Z}_p$ and implicitly sets

  $$r = \hat{r} + w_1 a^s + \cdots + w_n a^{s+1-n} = \hat{r} + \sum_{i=1}^{n} w_i a^{s+1-i}$$

  (note that this is properly distributed due to $\hat{r}$) by computing

  $$\begin{aligned} K_0 &= g^{\alpha} w^r = g^{a^{s+1}} g^{\alpha'} g^{a\hat{r}} \prod_{i=1}^{n} g^{w_i a^{s+2-i}} = g^{\alpha'} (g^a)^{\hat{r}} \prod_{i=2}^{n} (g^{a^{s+2-i}})^{w_i} \\ K_1 &= g^r = g^{\hat{r}} \prod_{i=1}^{n} (g^{a^{s+1-i}})^{w_i} \end{aligned}$$

  Moreover, for all $j \in [1, |S|]$, $\mathscr{B}$ calculates the values $K_{j,2} = g^{r_j}$ and $K_{j,3} =$

$(u^{A_j}h)^{r_j}v^{-r}$ as follows:

$$\begin{aligned}
v^{-r} &= v^{\hat{r}} \cdot (g^{\hat{v}} \prod_{\substack{j'\in[1,l]\\ k\in[1,n]}} g^{a^k M^*_{j',k}/b_{j'}})^{-\Sigma_{i=1}^{n} w_i a^{s+1-i}} \\
&= v^{\hat{r}} \cdot \prod_{i=1}^{n} (g^{a^{s+1-i}})^{\hat{v}w_i} \cdot \prod_{\substack{i,k\in[1,n]\\ j'\in[1,l]}} g^{-w_i M^*_{j',k} a^{s+1+k-i}/b_{j'}} \\
&= v^{\hat{r}} \cdot \prod_{i=1}^{n} (g^{a^{s+1-i}})^{\hat{v}w_i} \cdot \prod_{\substack{i,k\in[1,n],i\neq k\\ j'\in[1,l]}} (g^{a^{s+1+k-i}/b_{j'}})^{-w_i M^*_{j',k}} \cdot \prod_{\substack{i\in[1,n]\\ j'\in[1,l]}} g^{-w_i M^*_{j',k} a^{s+1}/b_{j'}} \\
&= Q \cdot \prod_{j'=1}^{l} g^{-\vec{w}\cdot M^*_{j'} a^{s+1}/b_{j'}} \\
&= Q \cdot \prod_{j'\in[1,l,]\rho(j)\notin S} g^{-\vec{w}\cdot M^*_{j'} a^{s+1}/b_{j'}}
\end{aligned}$$

such that

$$Q = v^{\hat{r}} \prod_{i=1}^{n} (g^{a^{s+1-i}})^{\hat{v}w_i} \cdot \prod_{\substack{i,k\in[1,n],i\neq k\\ j'\in[1,l]}} (g^{a^{s+1+k-i}/b_{j'}})^{-w_i M^*_{j',k}}.$$

The value Q can be calculated by $\mathscr{B}$ using the problem instance and since $\prod_{j'\in[1,l],\rho(j)\notin S} g^{-\vec{w}\cdot M^*_{j'} a^{s+1}/b_{j'}}$ should be wiped out by $(u^{A_j}h)^{r_j}$. Therefore, for all attribute $A_j \in S$, $\mathscr{B}$ implicitly computes

$$\begin{aligned}
r_j &= \hat{r}_j + r \cdot \sum_{i'\in[1,n],\rho^*(i')\notin S} \frac{b_{i'}}{A_j - \rho^*(i')} \\
&= \hat{r}_j + \hat{r} \cdot \sum_{i'\in[1,n],\rho^*(i')\notin S} \frac{b_{i'}}{A_j - \rho^*(i')} + \sum_{\substack{i\in[1,n]\\ i'\in[1,l],\rho^*(i')\notin S}} \frac{w_i b_{i'} a^{s+1-i}}{A_j - \rho^*(i')}
\end{aligned}$$

such that $\hat{r}_j$ is randomly chosen in $\mathbb{Z}_p$ and thus, r_j is properly distributed. The values b_i at the numerator cancel out with the values b_i^2 at the denominator, and thus cancel out the unknown part of v^{-r}.

Moreover, we note that r_j is well defined only for the attributes in the set S (that does not satisfy (M^*, ρ^*)) or the unrelated attributes (not in the policy), since the sum is over the i''s such that $\rho^*(i') \notin S$. Thus, for all $A_j \in S$ or all $A_j \notin \{\rho^*(i)\}_{i\in[1,l]}$, the denominators $A_j - \rho^*(i')$ are non-zero. If $\mathscr{B}$ tries to put more attributes in the policy, and possibly create a key for an unauthorised set, then it would have to divide by zero. Then, we obtain $K_{j,3} = (u^{A_j}h)^{r_j}v^{-r}$

such that

$$\begin{aligned}
(u^{A_j}h)^{r_j} &= (u^{A_j}h)^{\hat{r}_j} \cdot (g^{\hat{u}A_j+\hat{h}} \prod_{\substack{i\in[1,l]\\ k\in[1,n]}} g^{(A_j-\rho^*(i))M^*_{i,k}a^k/b_i^2})^{\hat{r}\cdot\sum_{i'\in[1,l],\rho^*(i')\notin S}\frac{b_{i'}}{A_j-\rho^*(i')}} \\
&\quad \cdot(g^{\hat{u}A_j+\hat{h}} \prod_{\substack{i\in[1,l]\\ k\in[1,n]}} g^{(A_j-\rho^*(i))M^*_{i,k}a^k/b_i^2})^{\sum_{\substack{i'\in[1,l],\rho^*(i')\notin S\\ j'\in[1,n]}}\frac{w_{j'}b_{i'}a^{s+1-j'}}{A_j-\rho^*(i')}} \\
&= (u^{A_j}h)^{\hat{r}_j} \cdot (K_{j,2}/g^{\hat{r}_j})^{\hat{u}A_j+\hat{h}} \\
&\quad \cdot \prod_{\substack{i,i'\in[1,l],\rho^*(i')\notin S\\ k\in[1,n]}} g^{\hat{r}(A_j-\rho^*(i))M^*_{i,k}b_{i'}a^k/(A_j-\rho^*(i'))b_i^2} \\
&\quad \cdot \prod_{\substack{i,i'\in[1,l],\rho^*(i')\notin S\\ j',k\in[1,n]}} g^{(A_j-\rho^*(i))w_{j'}M^*_{i,k}b_{i'}a^{s+1+k-j'}/(A_j-\rho^*(i'))b_i^2} \\
&= Q' \cdot \prod_{\substack{i\in[1,l],\rho^*(i)\notin S\\ i'\in[1,n]}} g^{(A_j-\rho^*(i))w_{i'}M^*_{i,i'}b_i a^{s+1+i'-i'}/(A_j-\rho^*(i))b_i^2} \\
&= Q' \cdot \prod_{i\in[1,l],\rho^*\notin S} g^{(\vec{w}\cdot M^*_i)a^{s+1}/b_i}
\end{aligned}$$

such that

$$\begin{aligned}
Q' &= (u^{A_j}h)^{\hat{r}_j} \cdot (K_{j,2}/g^{\hat{r}_j})^{\hat{u}A_j+\hat{h}} \cdot \prod_{\substack{i,i'\in[1,l],\rho^*(i')\notin S\\ k\in[1,n]}} (g^{b_{i'}a^k/b_i^2})^{\hat{r}(A_j-\rho^*(i))M^*_{i,k}/(A_j-\rho^*(i'))} \\
&\quad \cdot \prod_{\substack{i,i'\in[1,l]\\ j',k\in[1,n]\\ \rho^*(i')\notin S,(i\neq i'\vee j'\neq k)}} (g^{b_{i'}a^{s+1+k-j'}/b_i^2})^{(A_j-\rho^*(i))w_{j'}M^*_{i,k}/(A_j-\rho^*(i'))} \\
K_{j,2} &= g^{r_j} = g^{\hat{r}_j} \cdot \prod_{i'\in[1,l],\rho^*(i')\notin S} (g^{b_{i'}})^{\hat{r}/(A_j-\rho^*(i'))} \\
&\quad \cdot \prod_{\substack{i\in[1,n]\\ i'\in[1,l],\rho^*(i')\notin S}} (g^{b_{i'}a^s+1-i})^{w_i/(A_j-\rho^*(i'))}
\end{aligned}$$

The values Q' and $K_{j,2}$ can be calculated using the values from the problem instance. The second part of $(u^{A_j}h)^{r_j}$ cancels out with the unknown part of v^{-r}. Thus, $\mathscr{B}$ can compute $K_{j,2}$ and $K_{j,3}$ for all the attributes A_j in S and returns the private key $sk_S = (S, K_0, K_1, \{K_{j,2}, K_{j,3}\}_{j\in[1,|S|]})$ to $\mathscr{A}$.

- *Re-Encryption Key Query* $< S, (M,\rho) >$. If $(S, (M,\rho), \delta, \tilde{\beta}, rk_{S\to(M,\rho)}, *, 0, 1) \in L_{rk}$, then the challenger returns $rk_{S\to(M,\rho)}$ to the adversary. Otherwise, $\mathscr{B}$ proceeds as follows.
  - If S satisfies (M^*,ρ^*) and $(S, sk_S) \in L_{sk}$, then $\mathscr{B}$ returns a random bit in $\{0,1\}$ and aborts the simulation.
  - If S satisfies (M^*,ρ^*) and $(S, sk_S) \notin L_{sk}$, then $\mathscr{B}$ verifies whether $(S, (M,\rho), \delta, \tilde{\beta}, rk_{S\to(M,\rho)}, 1, 1, 0) \in L_{rk}$. If yes, $\mathscr{B}$ returns sk_S to the adversary and

resets $tag_2 = 0, tag_3 = 1$. Otherwise, the challenger selects at random $\theta, \sigma, \sigma_1, \cdots, \sigma_{|S|} \in_R \mathbb{Z}_p$, $\tilde{\beta}, \delta \in_R \{0,1\}^{\lambda}$, $\bar{K}_0, \bar{K}_1, \cdots, \bar{K}_{|S|} \in_R \mathbb{G}_1$. It then sets $RK_{0,1} = \bar{K}_0 \cdot h^{\theta}$, $RK_{0,2} = g^{\theta}$, $RK_1 = g^{\sigma}$ and for $j \in [1, |S|]$, $RK_{j,2} = g^{\sigma_j}$ and $RK_{j,3} = \bar{K}_j^{\sigma_j} \cdot v^{-\sigma}$, and constructs $Int\tilde{C}$ using $\tilde{\beta}$ and δ as in the real scheme. Eventually, $\mathscr{B}$ outputs $rk_{S \to (M,\rho)}$ to $\mathscr{A}$, and adds $(S, (M, \rho), \delta, \tilde{\beta}, rk_{S \to (M,\rho)}, 1, 0, 1)$ to L_{rk}.

  - Otherwise, if $(S, (M, \rho), \delta, \tilde{\beta}, rk_{S \to (M,\rho)}, 0, 1, 0) \in L_{rk}$, $\mathscr{B}$ gives $rk_{S \to (M,\rho)}$ to $\mathscr{A}$, and resets $tag_2 = 0, tag_3 = 1$. Otherwise, $\mathscr{B}$ first constructs the private key sk_S for the attribute set S as for the private key queries. The challenger then generates $rk_{S \to (M,\rho)}$ as in the real scheme, gives it to the adversary, and adds $(S, (M, \rho), \delta, \tilde{\beta}, rk_{S \to (M,\rho)}, 0, 0, 1)$ to L_{rk}.

- *Online Encryption Query* $< S, (M, \rho), IntC >$. The challenger checks whether the equation 6.1 holds. If not, meaning that either the intermediate ciphertext $IntC$ is invalid or S does not satisfy (M, ρ), then it returns $\perp$. Otherwise, $\mathscr{B}$ works as follows.

  - If S satisfies (M^*, ρ^*) and $(S, sk_S) \notin L_{sk}$, then the challenger first constructs the re-encryption key as the second case for the re-encryption key queries, re-encrypts $IntC$ and gives it to the adversary. It then adds $(S, (M, \rho),\ \delta, \tilde{\beta}, rk_{S \to (M,\rho)}, 1, 1, 0)$ to L_{rk} and $(S, (M, \rho), C_{(M,\rho)}, 0, 1, 0)$ to L_{one}.
  - Otherwise, the challenger first constructs $rk_{S \to (M,\rho)}$ as the third case for the re-encryption key queries, re-encrypts $IntC$ and gives it to the adversary. It then adds $(S, (M, \rho), \delta, \tilde{\beta}, rk_{S \to (M,\rho)}, 0, 1, 0)$ to L_{rk} and $(S, (M, \rho), C_{(M,\rho)}, 1, 0, 0)$ to L_{one}.
  - Otherwise, $\mathscr{B}$ verifies whether $(\beta, m, c) \in L_{H_2}$ such that $C_{0,2} = h^c = g^{\hat{h} \cdot c} \cdot \prod_{\substack{j \in [1,l] \\ k \in [1,n]}} (g^{a^k / b_j^2})^{-\rho^*(j) M^*_{j,k} \cdot c}$. If no such tuple exists, then the challenger returns $\perp$. Otherwise, it verifies whether $(S, (M, \rho), \delta, \tilde{\beta}, \perp, \perp, 1, \perp) \in L_{rk}$ such that S satisfies (M^*, ρ^*). If no, the challenger selects at random $\tilde{\beta}, \delta \in \{0,1\}^{\lambda}$, generates $Int\tilde{C}$ as in the real scheme, in order to hide δ and $\tilde{\beta}$. It then constructs $B_2 = (e(g^a, g^{a^q}) \cdot e(g, g^{\alpha'}))^{c \cdot \xi_1}$ where $\xi_1 = H_3(\delta)$. Eventually, $\mathscr{B}$ outputs $C_{(M,\rho)} = ((M, \rho), C_{0,1}, B_2, Int\tilde{C})$, gives it to the adversary, and adds $(S, (M, \rho), \delta, \tilde{\beta}, \perp, \perp, 1, \perp)$ to L_{rk} and $(S, (M, \rho), C_{(M,\rho)}, 0, 0, 1)$ to L_{one}.

- *Ciphertext Decryption Query* $< S, C_{(M,\rho)} >$. $\mathscr{B}$ first checks if there are tuples $(\tilde{\beta}, \delta, \tilde{c})$ and (β, m, c) in L_{H_2} such that $\tilde{C}_0 = g^{\tilde{c}}$ and $C_{0,2} = h^c$. If not, the challenger outputs $\perp$. Otherwise, it checks whether the equation 6.2 holds. It not, meaning that either $Int\tilde{C}$ is invalid or S does not satisfy (M, ρ), then it outputs $\perp$. Otherwise, $\mathscr{B}$ proceeds as follows:

  - If $(S, (M, \rho), \delta, \tilde{\beta}, rk_{S \to (M,\rho)}, 1, 0, 1) \in L_{rk}$ or $(S, (M, \rho), C_{(M,\rho)}, 0, 1, 0) \in L_{one}$, it verifies

$$\begin{aligned} &\forall j \in [1, l], \quad e(\tilde{C}_{j,1}, g) \overset{?}{=} e(w, g)^{\tilde{\lambda}_j} \cdot e(v, \tilde{C}_{j,3}) \\ &\forall j \in [1, l], \quad e(\tilde{C}_{j,2}, g) \overset{?}{=} e(u, \tilde{C}_{j,3}^{-1})^{\tilde{x}_j} \cdot e(h, \tilde{C}_{j,3}^{-1}) \end{aligned} \tag{6.3}$$

where given $I = \{i\}_{\rho(i)\in S}$ and M, it can find the vector $\vec{\tilde{w}} = \{\tilde{w}_i\}_{i\in I}$ of constants in $\mathbb{Z}_p$ such that $\sum_{i\in I} \tilde{w}_i \cdot \tilde{\lambda}_i = \tilde{c}$. If the above equations do not hold, then the challenger returns $\perp$. Otherwise, it sets $C_{0,1} = g^c$ and checks the equation 6.1. If the equation does not hold, then it returns $\perp$. Otherwise, $\mathscr{B}$ recovers the random re-encryption key $rk_{S\rightarrow(M,\rho)} = (S, RK_{0,1}, RK_{0,2}, RK_1, \{RK_{i,2}, RK_{i,3}\}_{i\in[1,k]}, Int\tilde{C})$ from L_{rk} and verifies the validity of B_2 as

$$B_2 \overset{?}{=} \frac{e(C_{0,1}, RK_{0,1})/e(C_{0,2}, RK_{0,2})}{e(w^{\sum_{i\in I} C_{i,4}\cdot w_i}, RK_1)\cdot \prod_{i\in I}(e(C_{i,1}, RK_1)\cdot e(C_{i,2}\cdot u^{C_{i,5}}, RK_{j,2})\cdot e(C_{i,3}, RK_{j,3}))^{w_i}}$$

where I and w_i are defined in the re-encryption phase. If the previous equation does not hold, then it returns $\perp$. Otherwise, the challenger verifies whether $(R, \delta_1) \in L_{H_1}$ such that $B_1 = (\beta\|m)\oplus\delta_1$ and $R = e(g,g)^{\alpha\cdot c}$. If no such tuple exists, $\mathscr{B}$ returns $\perp$. Otherwise, it outputs m and gives it to $\mathscr{A}$.

- Otherwise, if $(S, sk_S) \in L_{sk}$, the challenger recovers m as in the real scheme using sk_S. Otherwise, $\mathscr{B}$ verifies whether both equations 6.2 and 6.3 hold. If not, it returns $\perp$. Otherwise, it verifies whether $(R, \delta_1) \in L_{H_1}$ such that $B_1 = (\beta\|m)\oplus\delta_1$ and $R = e(g,g)^{\alpha\cdot c}$, and $B_2 = e(g,g)^{\alpha\cdot c\cdot\xi_1}$. If no such tuple exists and the equations do not hold, $\mathscr{B}$ returns $\perp$. Otherwise, it outputs m and gives it to $\mathscr{A}$.

Challenge. The challenge ciphertext is constructed as follows. The adversary gives two messages m_0 and m_1 of equal length to the challenger. $\mathscr{B}$ chooses a bit $\mu \in_R \{0,1\}$ and answers to $\mathscr{A}$ as follows. $IntC$ is generated as in the real scheme, meaning that $IntC = (c, B_1, C_{0,1}, C_{0,2}, \{\lambda'_j, t_j, x_j, C_{j,1}, C_{2,j}, C_{3,j}\}_{j\in[1,l]}, D)$ We suppose that the value $C_{0,2}$ is not necessarily output.

Then, $\mathscr{B}$ chooses at random $\tilde{\beta}^*, \delta^* \in_R \{0,1\}^\lambda$, issues an H_3 query on δ^* to obtain ξ_1^*. We notice that in the previous step, (β, m_μ) must be issued to H_2 such that the tuple (β, m_μ, c) is already stored in L_{H_2}, where $\beta \in_R \{0,1\}^\lambda$ and $c \in_R \mathbb{Z}_p$. Thus, it recovers (β, m_μ, c) from L_{H_2} and computes $B_2^* = (e(g^a, g^{a^s})\cdot e(g, g^{\alpha'}))^{c\cdot\xi_1^*}$. Moreover, $\mathscr{B}$ chooses at random $\tilde{B}_1^* \in_R \{0,1\}^{2\lambda}$ and implicitly sets $H_1(Z\cdot e(g^{\tilde{c}}, g^{\alpha'})) = \tilde{B}_1^* \oplus (\delta^*\|\tilde{\beta}^*)$ and $\tilde{C}_0^* = g^{\tilde{c}}$.

$\mathscr{B}$ can choose the secret that is splitted in order to cancel out the terms. For that, it chooses at random $\tilde{y}'_2, \cdots, \tilde{y}'_n$ and shares the secret using the vector $\vec{\tilde{y}} = (\tilde{c}, \tilde{c}a + \tilde{y}'_2, \cdots, \tilde{c}a^{n-1} + \tilde{y}'_n) \in \mathbb{Z}_p^n$. We note that the secret $\tilde{c}$ and the vector $\vec{\tilde{y}}$ are properly distributed since $\tilde{c}$ is information-theoretically hidden from $\mathscr{A}$ and the $\tilde{y}'_j$'s are uniformly randomly chosen. Since $\vec{\tilde{\lambda}} = M^*\cdot\vec{\tilde{y}}$, we have that $\tilde{\lambda}_j = \sum_{i\in[1,n]} M^*_{j,i}\tilde{c}a^{i-1} + \sum_{i\in[2,n]} M^*_{j,i}\tilde{y}'_i = \sum_{i\in[1,n]} M^*_{j,i}\tilde{c}a^{i-1} + \tilde{\lambda}'_j$ for each row $j \in [1,l]$. We note that the values $\tilde{\lambda}'_j = \sum_{i\in[2,n]} M^*_{j,i}\tilde{y}'_i$ are known to $\mathscr{B}$.

For each row, $\mathscr{B}$ implicitly sets $\tilde{t}_j = -\tilde{c}b_j$, which is properly distributed since the b_j's are information-theoretically hidden from $\mathscr{A}$. Then, $\mathscr{B}$ computes the following:

$$\begin{aligned}
\tilde{C}^*_{j,1} &= w^{\tilde{\lambda}_j} v^{\tilde{t}_j} = w^{\tilde{\lambda}'_j} \prod_{i=1}^{n} g^{M^*_{j,i}\tilde{c}a^i} \cdot (g^{\tilde{c}b_j})^{-\hat{v}} \cdot \prod_{\substack{i'\in[1,l]\\ k\in[1,n]}} g^{-M^*_{i',k}a^k\tilde{c}b_j/b_{i'}} \\
&= w^{\tilde{\lambda}'_j} \cdot (g^{\tilde{c}b_j})^{\hat{v}} \cdot \prod_{i=1}^{n} g^{M^*_{j,i}\tilde{c}a^i} \cdot \prod_{k=1}^{n} g^{M^*_{j,k}a^k\tilde{c}b_j/b_j} \cdot \prod_{\substack{i'\in[1,l],i'\neq j\\ k\in[1,n]}} g^{-M^*_{i',k}a^k\tilde{c}b_j/b_{i'}} \\
&= w^{\tilde{\lambda}'_j} \cdot (g^{\tilde{c}b_j})^{\hat{v}} \cdot \prod_{\substack{i'\in[1,l],i'\neq j\\ k\in[1,n]}} (g^{\tilde{c}a^k b_j/b_{i'}})^{-M^*_{i',k}} \\
\tilde{C}^*_{j,2} &= (u^{\rho^*(j)}h)^{\tilde{t}_j} = (g^{\tilde{c}b_j})^{-(\hat{u}\rho^*(j)+\hat{h})} \cdot (\prod_{\substack{i'\in[1,l]\\ k\in[1,n]}} g^{(\rho^*(j)-\rho^*(i'))M^*_{i',k}a^k/b^2_{i'}})^{-\tilde{c}b_j} \\
&= (g^{\tilde{c}b_j})^{-(\hat{u}\rho^*(j)+\hat{h})} \cdot \prod_{\substack{i'\in[1,l],i'\neq j\\ k\in[1,n]}} (g^{\tilde{c}a^k b_j/b^2_{i'}})^{-(\rho^*(j)-\rho^*(i'))M^*_{i',k}})^{-\tilde{c}b_j} \\
C^*_{j,3} &= g^{\tilde{t}_j} = (g^{\tilde{c}b_j})^{-1}
\end{aligned}$$

Since $\tilde{t}_j = -\tilde{c}b_j$, we “raised” the exponents of the value so that they cancel out with the exponents of $w^{\tilde{\lambda}_j}$. The challenger then issues a H_5 query on $(\tilde{B}^*_1, \tilde{C}^*_0, (\tilde{C}^*_{1,1}, \tilde{C}^*_{1,2}, \tilde{C}^*_{1,3}), \cdots, (\tilde{C}^*_{l,1}, \tilde{C}^*_{l,2}, \tilde{C}^*_{l,3}), (M^*, \rho^*))$ to obtain $(\tilde{B}^*_1, \tilde{C}^*_0, (\tilde{C}^*_{1,1}, \tilde{C}^*_{1,2}, \tilde{C}^*_{1,3}), \cdots, (\tilde{C}^*_{l,1}, \tilde{C}^*_{l,2}, \tilde{C}^*_{l,3}), (M^*, \rho^*), \xi^*_3, \delta^*_3)$, and sets $\tilde{D}^* = (g^{\tilde{c}})^{\xi^*_3}$. Then, $\mathscr{B}$ sets $Int\tilde{C}^* = ((M^*, \rho^*), B^*_1, C^*_0, \{C^*_{j,1}, C^*_{j,2}, C^*_{j,3}\}_{j\in[1,l]}, D^*)$.

Finally, $\mathscr{B}$ returns the ciphertext $C^*_{(M^*,\rho^*)} = ((M^*, \rho^*), C_{0,1}, B^*_2, Int\tilde{C})$ to $\mathscr{A}$.

If $Z = e(g,g)^{a^{s+1}\cdot c}$, then $C^*_{(M^*,\rho^*)}$ is a valid ciphertext. Indeed, the components corresponding to $IntC$ are valid. Since $IntC$ is encrypted to $C^*_{(M^*,\rho^*)}$ under a valid re-encryption key $rk_{S\to(M,\rho)}$, the on-line encryption must be valid, meaning that the construction of B^*_2 is valid. Moreover, it is easy to see that the rest of the components are valid too. Nevertheless, if Z is a random element in $\mathbb{G}_T$, then the challenge ciphertext is independent of the bit μ for $\mathscr{A}$.

Query Phase 2. Same as in the query phase 1, except that we consider the constraints defined in Section 6.1.3.

Guess. $\mathscr{A}$ outputs a bit $\mu' \in \{0,1\}$ for μ. If $\mu' = \mu$, then $\mathscr{B}$ outputs 0, meaning that it claims that the challenge Z is equal to $e(g,g)^{a^{s+1}\cdot c}$; otherwise, it outputs 1.

If $Z = e(g,g)^{a^{s+1}\cdot c}$, then $\mathscr{A}$ played the proper security game, since $B_1 = (m_\mu \| \beta) \oplus H_1(e(g, g^c)^{\alpha'} \cdot Z) = (m_\mu \| \beta) \oplus H_1(e(g,g)^{\alpha\cdot c})$. If Z is a random term in $\mathbb{G}_T$, then all the information about the message m_μ is lost in the challenge ciphertext. Thus, the advantage of $\mathscr{A}$ is exactly 0. Therefore, if $\mathscr{A}$ breaks the proper security game with a non-negligible advantage, then $\mathscr{B}$ has a non-negligible advantage in breaking the s-type assumption.

Analysis. The simulations of the random oracles are perfect except for H_1, H_2 and H_3. Let H^*_1 and H^*_2 be the events that $\mathscr{A}$ has queried $R^* = e(g,g)^{\alpha c}$ to H_1 (with probability of a successful query equal to $q_{H_1}/2^{2\lambda}$) and (β^*, m_μ) to H_2 (with probability of a successful query be equal to q_{H_2}/p) before the challenge phase. Let H_3^* denotes the event that $\mathscr{A}$ has queried δ^* to H_3 before the challenge phase (this happens with probability q_{H_3}/p). In the simulation of private key oracle, the responses to $\mathscr{A}$ are perfect.

In the simulation of ciphertext decryption oracle, it might be possible that the challenger cannot provide a decryption for a valid ciphertext. Suppose the adversary can generate a valid ciphertext without querying $e(g,g)^{\alpha c}$ to H_1, where $c = H_2(\beta, m)$. Let $Valid$ be the event that the ciphertext is valid, $QueryH_1$ be the event that the adversary has queried $e(g,g)^{\alpha c}$ to H_1 and $QueryH_2$ be the event that the adversary has queried (β, m) to H_2. From the simulation, $Pr[Valid \mid \neg QueryH_1] \leq Pr[QueryH_2 \mid QueryH_1] + Pr[Valid \mid QueryH_1 \wedge \neg QueryH_2] \leq q_{H_2}/2^{2\lambda} + 1/p$ and $Pr[Valid \mid \neg QueryH_2] \leq q_{H_1}/2^{2\lambda} + 1/p$, where q_{H_1}, q_{H_2} are the total numbers of queries to the random oracles H_1 and H_2. Let $Pr[ErrDec]$ be the probability that the event $Valid \mid (\neg QueryH_1 \vee \neg QueryH_2)$ occurs, then $Pr[ErrDec] \leq (\frac{q_{H_1}+q_{H_2}}{2^{2\lambda}} + \frac{2}{p})q_d$, where q_d is the total number of decryption queries.

Let Bad denote the event that $(H_2^* \mid \neg H_1^*) \vee H_1^* \vee H_3^* \vee ErrDec$, then

$$\begin{aligned} Adv^{S-IND-CCA}_{OO-CP-AB-PRE,\mathscr{A}} &= \mid Pr[\mu' = \mu] - 1/2 \mid \leq \frac{1}{2} Pr[Bad] \\ &= \frac{1}{2} Pr[(H_2^* \mid \neg H_1^*) \vee H_1^* \vee H_3^* \vee ErrDec] \\ &\leq \frac{1}{2}(\frac{q_{H_3} + 2q_d}{p} + \frac{q_{H_2} + (q_{H_1} + q_{H_2})q_d}{2^{2\lambda}}) \end{aligned}$$

Therefore, $Adv^{s\text{-type}}_{\mathscr{B}} \geq \frac{1}{q_{H_1}}(2Adv^{S-IND-CCA}_{OO-CP-AB-PRE,\mathscr{A}} - \frac{q_{H_2}+(q_{H_1}+q_{H_2})q_d}{2^{2\lambda}} - \frac{q_{H_3}-2q_d}{p})$.

Selective Collusion Resistance Security Proof

Theorem. Suppose that the BDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$ and the hash functions H_1, H_2, H_3, H_4 and H_5 are seen as random oracles, then the OO-CP-AB-PRE scheme is selectively collusion resistant in the random oracle model.

Suppose there is an adversary $\mathscr{A}$ who breaks the collusion resistance of the OO-CP-AB-PRE scheme with advantage $Adv^{S-CollRes}_{OO-CP-AB-PRE,\mathscr{A}}$ greater than ε. We then construct a challenger $\mathscr{B}$ that should output $e(g,g)^{abd}$ playing the selectively collusion resistance game with $\mathscr{A}$ as follows.

$\mathscr{B}$ takes as inputs $(p, \mathbb{G}_1, \mathbb{G}_T, e)$, a generator g of $\mathbb{G}_1$ and a BDH problem instance (g^a, g^b, g^d) for some unknown exponents $a, b, d \in \mathbb{Z}_p$. The goal of the challenger is to output $e(g,g)^{abd}$.

Initialization. The adversary gives the challenge set S^* to $\mathscr{B}$.

Setup. The challenger sets $w = g^b$ and $e(g,g)^\alpha = e(g^a, g^b) = e(g,g)^{ab}$. It also chooses at random $u, h, v \in_R \mathbb{G}_1$. It gives these public parameters to $\mathscr{A}$.

We note that the public parameters are identical to those in the real scheme for the adversary. At any time, $\mathscr{A}$ can adaptively query the random oracles H_j, for $j \in [1,5]$, which are controlled by $\mathscr{B}$ as follows. The challenger maintains the lists H_j^{List}, for $j \in [1,5]$, which are initially empty, and answers the queries to the random oracles as follows.

- H_1: on receipt of an H_1 query on $R \in \mathbb{G}_T$, if there is a tuple $(R, \delta_1) \in L_{H_1}$, $\mathscr{B}$ forwards the predefined value δ_1 to $\mathscr{A}$, where $\delta_1 \in \{0,1\}^{2\lambda}$. Otherwise, $\mathscr{B}$ sets $H_1(R) = \delta_1$, responds δ_1 to $\mathscr{A}$ and adds the tuple (R, δ_1) to L_{H_1}, where $\delta_1 \in_R \{0,1\}^{2\lambda}$.
- H_2: on receipt of an H_2 query on (β, m), if there is a tuple $(\beta, m, c) \in L_{H_2}$, $\mathscr{B}$ forwards the predefined value c to $\mathscr{A}$, where $c \in \mathbb{Z}_p$. Otherwise, $\mathscr{B}$ sets $H_2(\beta, m) = c$, responds c to $\mathscr{A}$ and adds the tuple (β, m, c) to L_{H_2}, where $c \in_R \mathbb{Z}_p$.
- H_3: on receipt of an H_3 query on $\delta \in \{0,1\}^{\lambda}$, if there is a tuple (δ, ξ_1) in L_{H_3}, $\mathscr{B}$ forwards the predefined value ξ_1 to $\mathscr{A}$, where $\xi_1 \in \mathbb{Z}_p$. Otherwise, $\mathscr{B}$ sets $H_3(\delta) = \xi_1$ to $\mathscr{A}$ and adds the tuple (δ, ξ_1) to L_{H_3}, where $\xi_1 \in_R \mathbb{Z}_p$.
- H_4: on receipt of an H_4 query on $(B_1, C_{0,1}, C_{0,2}, (C_{1,1}, C_{1,2}, C_{1,3}), \cdots, (C_{l,1}, C_{l,2}, C_{l,3}))$, if there is a tuple $(B_1, C_{0,1}, C_{0,2}, (C_{1,1}, C_{1,2}, C_{1,3}), \cdots, (C_{l,1}, C_{l,2}, C_{l,3}), \xi_2, \delta_2) \in L_{H_4}$, $\mathscr{B}$ forwards the predefined value δ_2 to $\mathscr{A}$, where $\xi_2 \in \mathbb{Z}_p, \delta_2 \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ sets $\delta_2 = g^{\xi_2}$, responds δ_2 to $\mathscr{A}$ and adds the tuple $(B_1, C_{0,1}, C_{0,2}, (C_{1,1}, C_{1,2}, C_{1,3}), \cdots, (C_{l,1}, C_{l,2}, C_{l,3}), \xi_2, \delta_2)$ in L_{H_4}, where $\xi_2 \in_R \mathbb{Z}_p$.
- H_5: on receipt of an H_5 query on $(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M, \rho))$, if there is a tuple $(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M, \rho), \xi_3, \delta_3) \in L_{H_5}$, $\mathscr{B}$ forwards the predefined value δ_2 to $\mathscr{A}$, where $\xi_3 \in \mathbb{Z}_p, \delta_3 \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ sets $\delta_3 = g^{\xi_3}$, responds δ_3 to $\mathscr{A}$ and adds the tuple $(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M, \rho), \xi_3, \delta_3)$ in L_{H_5}, where $\xi_3 \in_R \mathbb{Z}_p$.

Query Phase. The challenger answers to $\mathscr{A}$'s queries as follows.

- *Private Key Query* $< S >$. Let $S = \{A_i\}_{i \in [1,k]}$ be an attribute set such that $S \neq S^*$. We suppose there is an attribute j such that either $(j \in S \wedge y \notin S^*)$ or $(j \notin S \wedge y \in S^*)$. Without loss of generality, we assume that $(j \notin S \wedge y \in S^*)$. For $i \in [1,k]$, $\mathscr{B}$ randomly chooses $r'_i \in_R \mathbb{Z}_p$ and implicitly sets the elements r and r_i as follows:

$$\begin{aligned} r_i &= r'_i, \text{ for } i \in [1,k], \quad i \neq j \\ r_j &= -a + r'_j \\ r &= -a + \sum_{i=1}^{k} r'_i \end{aligned}$$

  Therefore, the components of the private key sk_S can be computed as follows:

$$\begin{aligned} K_0 &= g^{\alpha} w^{r} = g^{ab} \cdot g^{b(-a+\sum_{i=1}^{k} r'_i)} = g^{b(\sum_{i=1}^{k} r'_i)} \\ K_1 &= g^{r} = g^{-a+\sum_{i=1}^{k} r'_i} \\ K_{i,2} &= g^{r'_i}, \text{ for } i \in [1,k], \quad i \neq j \\ K_{i,3} &= (u^{A_i} h)^{r'_i} \cdot v^{a-\sum_{i=1}^{k} r'_i}, \text{ for } i \in [1,k], \quad i \neq j \\ K_{j,2} &= g^{-a+r'_j} \\ K_{j,3} &= (u^{A_j} h)^{-a+r'_j} \cdot v^{a-\sum_{i=1}^{k} r'_i} \end{aligned}$$

- *Re-Encryption Key Query* $< S, (M, \rho) >$. Let S be an attribute set. If $S \neq S^*$, then $\mathscr{A}$ gets the private key sk_S from $\mathsf{KeyGen}(params, msk, S)$ and the re-encryption key $rk_{S \to (M,\rho)}$ from $\mathsf{ReKeyGen}(params, S, sk_S, (M, \rho))$. Otherwise, the adversary proceeds as follows.

First, $\mathscr{B}$ computes $Int\tilde{C}$: it picks at random $\tilde{\beta}, \delta \in_R \{0,1\}^{\lambda}$ and issues a H_2 query on $(\tilde{\beta}, \delta)$ to obtain $\tilde{c}$ and a H_1 query on $e(g,g)^{\alpha \cdot \tilde{c}} = e(g,g)^{ab \cdot \tilde{c}}$ to obtain δ_1. It then computes $\tilde{B}_1 = (\delta \| \tilde{\beta}) \oplus \delta_1$ and $\tilde{C}_0 = g^{\tilde{c}}$. It also picks at random $\tilde{y}_2, \cdots, \tilde{y}_n \in_R \mathbb{Z}_p$, sets the vector $\vec{\tilde{y}} = (\tilde{c}, \tilde{y}_2, \cdots, \tilde{y}_n)^T$, and computes a vector of shares of $\tilde{c}$ as $(\tilde{\lambda}_1, \cdots, \tilde{\lambda}_l)^T = M \cdot \vec{\tilde{y}}$, where M is a $l \times n$ matrix and $l \leq P$. Then, for $j \in [1,l]$, it chooses at random $\tilde{t}_j \in_R \mathbb{Z}_p$ and computes $\tilde{C}_{j,1} = w^{\tilde{\lambda}_j} v^{\tilde{t}_j}$, $\tilde{C}_{j,2} = (u^{\rho(j)} h)^{-\tilde{t}_j}$ and $\tilde{C}_{j,3} = g^{\tilde{t}_j}$.
Moreover, it issues a H_5 query on $(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M, \rho))$ to obtain $(\tilde{B}_1, \tilde{C}_0, (\tilde{C}_{1,1}, \tilde{C}_{1,2}, \tilde{C}_{1,3}), \cdots, (\tilde{C}_{l,1}, \tilde{C}_{l,2}, \tilde{C}_{l,3}), (M, \rho), \xi_3, \delta_3)$, and sets $\tilde{D} = (g^{\tilde{c}})^{\xi_3}$. Eventually, the intermediate ciphertext for the re-encryption key is $Int\tilde{C} = ((M, \rho), \tilde{B}_1, \tilde{C}_0, \{\tilde{C}_{j,1}, \tilde{C}_{2,j}, \tilde{C}_{3,j}\}_{j \in [1,l]}, \tilde{D})$.
It then issues a H_3 query on δ to obtain ξ_1 and randomly chooses $\theta \in_R \mathbb{Z}_p$. It computes $RK_{0,1} = K_0^{\xi_1} = g^{b(\Sigma_{i=1}^{k} r'_i)\xi_1} h^{\theta}$, $RK_{0,2} = g^{\theta}$, and $RK_1 = K_1^{\xi_1} = g^{(-a+\Sigma_{i=1}^{k} r'_i)\xi_1}$. It also chooses an index j in $[1,k]$ and computes $RK_{i,2}$ and $RK_{i,3}$ for all $i \in [1,k]$ as follows:

$$
\begin{aligned}
RK_{i,2} &= K_{i,2}^{\xi_1} = g^{r'_i \xi_1}, \text{ for } i \in [1,k], \quad i \neq j \\
RK_{i,3} &= K_{i,3}^{\xi_1} = ((u^{A_i} h)^{r'_i} \cdot v^{a - \Sigma_{i=1}^{k} r'_i})^{\xi_1}, \text{ for } i \in [1,k], \quad i \neq j \\
RK_{j,2} &= K_{j,2}^{\xi_1} = g^{(-a+r'_j)\xi_1} \\
RK_{j,3} &= K_{j,3}^{\xi_1} = ((u^{A_j} h)^{-a+r'_j} \cdot v^{a - \Sigma_{i=1}^{k} r'_j})^{\xi_1}
\end{aligned}
$$

$\mathscr{B}$ returns the re-encryption key $rk_{S \to (M,\rho)} = (S, RK_{0,1}, RK_{0,2}, RK_1, \{RK_{i,2}, RK_{i,3}\}_{i \in [1,k]}, Int\tilde{C})$ to $\mathscr{A}$.

Output. The challenge private key $sk^*_{S^*} = (S^*, K^*_0, K^*_1, \{K^*_{i,2}, K^*_{i,3}\}_{i \in [1,k^*]})$ for the attribute set $S^* = \{A^*_i\}_{i \in [1,k^*]}$ is constructed as follows. If $sk^*_{S^*}$ is a valid key, then it should satisfy the following equation:

$$
\frac{e(g, K^*_0) \cdot \prod_{i=1}^{k^*} e(u^{A^*_i} h, g)}{e(wv, K^*_1) \cdot e(g, K^*_{i,3})} = e(g,g)^{ab} = e(g,g)^{\alpha}.
$$

Then, $\mathscr{B}$ outputs:

$$
\frac{e(g^d, K^*_0) \cdot \prod_{i=1}^{k^*} e(u^{A^*_i} h, g^d)}{e((wv)^d, K^*_1) \cdot e(g^d, K^*_{i,3})} = e(g,g)^{d \cdot \alpha} = e(g,g)^{abd}
$$

and solves the BDH problem.

Analysis. In the simulation of private key oracle, the responses to $\mathscr{A}$ are perfect. The simulations of the random oracles are perfect except for H_3. Let H_3^* be the event that $\mathscr{A}$ has queried δ to H_3 before the output phase (this event happens with probability q_{H_3}/p). Therefore, $Adv_{\mathscr{B}}^{BDH} \geq \varepsilon - \frac{q_{H_3}}{p}$.

6.1.6 Performance

In the following table, we evaluate the efficiency of our OO-CP-AB-PRE symmetric pairing-based scheme. We use results of cryptographic operation implementations (group exponentiation of a random group element with a random exponent and pairing operations on random group elements) using the MIRACL framework [203, 196] for a 128-bit security level. All the following experiments are based on a dual core IntelR XeonR CPU W3503@2.40GHz with 2.0GB RAM running Ubuntu R10.04. The elliptic curve utilised for all the benchmarks was the super-singular symmetric curve $y^2 = x^3 + 1 \mod p$ with embedding degree 2 for suitable primes p.

We do not take into account multiplication/division in $\mathbb{G}_1$ and $\mathbb{G}_T$ as well as exponentiation in $\mathbb{G}_T$ as these operations have timings negligible compared to the ones for exponentiation in $\mathbb{G}_1$ and pairing operation.

	Group Exponentiation in $\mathbb{G}_1$	Pairing
Time/operation	34.4	176.6
Setup		176.6
KeyGen	1,135.2	
OffEncrypt	1,823.2	176.6
ReKeyGen	2,648.8	176.6
OnEncrypt	1,410.4	17,483.4
Decrypt	412.8	5,827.8

Table 6.1: Timings for our OO-CP-AB-PRE symmetric pairing-based system. Times are in milliseconds. We assume that there are up to 10 attributes involved in the protocol.

The algorithms Setup and KeyGen should be run only once to generate the public parameters and the static private keys for all the users. The execution of the algorithm KeyGen depends on the number of attributes of a given user. Attributes are also involved in algorithms OffEncrypt, ReKeyGen, OnEncrypt and Decrypt, and timings significantly result from their number. For each algorithm execution, we consider a maximum number of 10 attributes: such number seems realistic to allow users to find specifications and requirements decribing other users (when generating private keys) and medical information (when generating ciphertexts). Finally, timings are substancial in algorithms ReKeyGen, OnEncrypt and Decrypt due to the verification process in each of them. We can propose to pre-compute the values needed for the verification of the components. Indeed, if the verification process is skipped, we notice that the computation of the ciphertext components in the algorithm OnEncrypt only requires 6,550.2 milliseconds. In the algorithm Decrypt, it only takes 5,676.4 milliseconds to recover the message when not processing the verification part.

6.1.7 Conclusion

We proposed the notion of on-line/off-line ciphertext attribute-based proxy re-encryption (OO-CP-AB-PRE) to directly solve the e-Health scenario. We provided a clear definition of this primitive along with the corresponding security models. The OO-CP-AB-PRE construction is efficient and practical, and is built in the random oracle model. We offered a selective access structure and chosen-ciphertext security game for the OO-CP-AB-PRE system, to cover the attacks taken by an outsider. We also showed that the OO-CP-AB-PRE scheme is selectively collusion resistant, to cover the attacks triggered by an insider.

6.2 Ciphertext-Policy DNA-Based Encryption

Consider the following scenario. A patient, Alice, has the possibility to make her Electronic Health Records (EHRs) reachable only to her descendents in case of unability to access them (because of illness, disability or death for instance). Without loss of generality, we assume that Alice has only one son, Charles. To do so, the hospital suggests her to encrypt the EHR using her DNA sequences as a key, and to send the ciphertext to the cloud server of the hospital, acting as a proxy.

Let us suppose now that Alice is unable to access her EHR herself. Thus, she can ask Charles to use his DNA to conduct a successful decryption and retrieve her EHR in order to take decisions about her health situation. The decryption is successful if and only if Charles is a true descendent of Alice, meaning that he passes the DNA parentage test. Nevertheless, a random individual, Bob, will not be able to conduct a successful decryption, even if he colludes with other people who are not Alice's relatives, since Bob does not have the required DNA sequences, and hence, he will fail the DNA parentage test. Additionally, Bob will not learn anything else, except that the decryption process is unsuccessful. We depict the scenario in Figure 6.2.

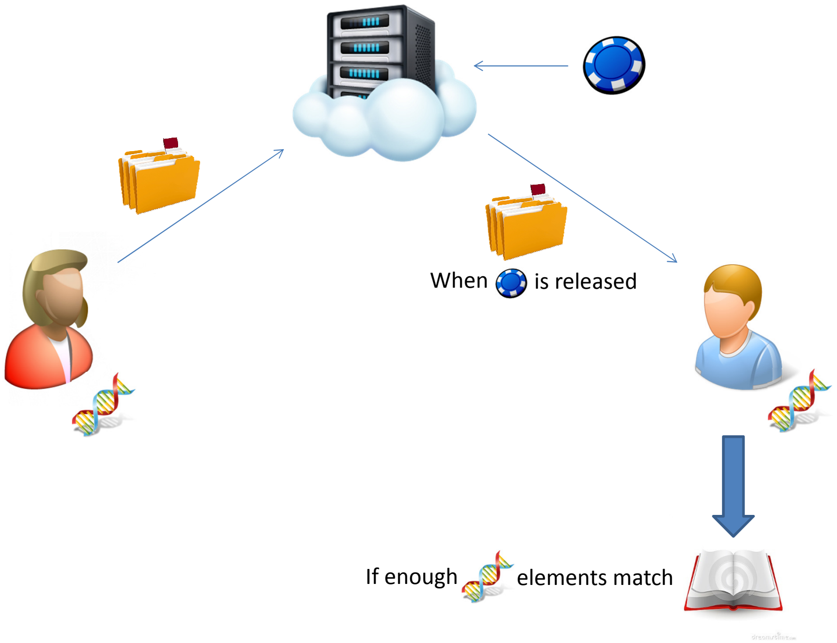


Figure 6.2: Ciphertext-Policy DNA-Based Encryption. A patient, Alice, encrypts a document m using her DNA and sends the resulting ciphertext C to a cloud server, acting as a proxy. The latter has to keep C on its local storage until a token is released (when Alice becomes unable to proceed for instance). When this happens, the proxy forwards the ciphertext C to the patient Charles, Alice's son. Charles uses his DNA (close enough to Alice's one) to retrieve m.

DNA Parentage Test. Ostrowski [177] gave some definitions and explanations related to the paternity indices. DNA parental test is currently the most advanced and accurate technology to determine parental relationship. It uses genetic fingerprints to determine whether two individuals have a biological parentchild relationship. Hence, it establishes genetic evidence whether a man (respectively woman) is the biological father (respectively mother) of an individual. When testing parental relationship, the result, called “probability of parentage”, is 0 if the parent is not biologically related to the child, and is typically close to 1 otherwise.

Some but rare individuals are called “chimeras“ due to the fact that they have at least two different sets of genes. Nevertheless, almost all individuals have a single and distinct set of genes. Therefore, we meet several cases of DNA profiling that falsely “proved” that a mother was unrelated to her children. Since this has a negligible probability, we assume that these exceptions are not applicable in our protocol.

In the following, we give some useful definitions to explain the vocabulary on genetics that will be used throughout this section:

- *DNA.* The deoxyribonucleic acid (DNA) is a molecule that encodes the genetic information found in all human organisms as a sequence of nucleotides using letters G, A, T, and C, corresponding to guanine, adenine, thymine, and cytosine respectively. The molecule is formed as double-stranded helices, consisting of two long polymers of simple units called nucleotides (i.e. the molecules with backbones made of alternating sugars and phosphate groups). These characters allow the DNA to be well-suited for biological information storage.

- *Chromosome.* A chromosome is an organized structure of DNA found in cells. This single piece of coiled DNA contains many genes, regulatory elements and other nucleotide sequences. It also holds DNA-bound proteins, which serve to package the DNA and control its functions.

- *Allele.* An allele is the alternative forms of the same gene for a character, producing different effects. For instance, different alleles can result in different observable phenotypic traits, such as a different pigmentation. However, many variations at the genetic level result in little or no observable variation.

- *Minisatellites.* The minisatellites or also called variable number tandem repeats (VNTR), are repeated combined sequences, such that the size of one sequence is from 10 to 60 nucleotides. They are present in all species, are particularly studied in humans, and largely found in the genome. We can easily observe replication errors in these minisatellites, including replication slippage, which are at the origin of interindividual on the number of repetition variations. This variability has many applications, such as the DNA parentage test.

Next, we explain the principle of the DNA parentage test. Each individual has, in his/her chromosomes, some portions of DNA that encode genes and other portions for which no usefulness has yet been discovered. This last part has an interesting characteristic: some sequences of nucleotide pairs follow on from each other, identically repeated. These repeated sequences are called *minisatellites*. The size of these minisatellites, corresponding to the number of repetitions of DNA sequences, highly varies from person to person, comprised from 5 to 55.

This size is a characteristic of the chromosome. Therefore, it inherits in the same way than for its alleles: for each chromosome pair of an individual, there will be one that will match the characteristics of the father and another one that will match with those of the mother.

Thus, parental testing is to assess the size of some specific minisatellites. For each chromosome, we need to compare these characteristics between the chromosomes of the child and those of the prospective parents. We illustrate the aforementioned idea in the example below and in Figure 6.3.

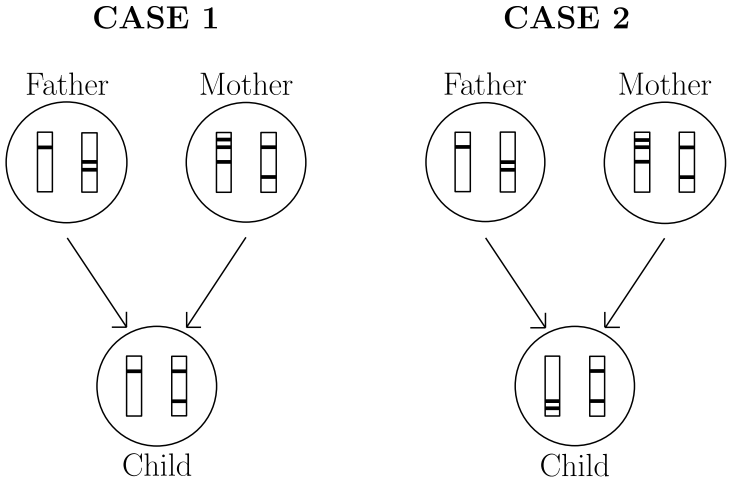


Figure 6.3: DNA Parentage Tes. In case 1, the father and the mother have biological parent-child relationship with the child since the child's chromosomes match one chromosome of each parents' pair. However, in case 2, the father is not the biological father of the child since they do not share any common chromosome.

Example. Let us consider the following example. The table contains genetic maps of three people: the mother Alice and two presumed sons, labelled as Bob and Charles respectively. In the table, the size of the minisatellite ACGCC is indicated for each chromosome pair. For instance, Alice has the sequence ACGCCACGCCACGCCACGCCACGC-CACGCCACGCC (number of repetitions is equal to 7) on its first chromosome 14.

	Size of the minisatellite		
Analyzed chromosome pair - left, right	Alice	Bob	Charles
Chromosome pair 1	18, 15	5, 15	18, 13
Chromosome pair 8	13, 14	17, 20	55, 14
Chromosome pair 13	34, 15	34, 9	34, 14
Chromosome pair 14	7, 24	5, 14	12, 7

We note that for each chromosome pair, Alice and Charles have one element in common. However, Alice and Bob have only one element in common for the chromosome pair number 1 and the chromosome pair number 13, thus Bob is not Alice's biological son. Moreover, we can conclude with a high probability that Charles is the biological son of Alice.

6.2.1 Contributions

Motivated by the above scenario, we provide a new primitive called ciphertext-policy DNA-based encryption (CB-DBE), to encrypt documents using the DNA information of an individual. We allow the receiver to use his/her DNA information to decrypt the ciphertext, if and only if his/her DNA sequences match the original DNA sequences embedded in the ciphertext. The matching criteria is done via the parentage test since the DNA has an interesting and useful characteristic, namely the number of sequence repetitions is specific for each individual. The size of the minisatellites of each pair of chromosomes will be treated as attributes and used in the encryption and decryption processes. Furthermore, the solution is privacy preserving, which means that a party learns nothing else regarding the DNA information.

Note that the CP-DBE primitive is closely related to the well-known CP-ABE primitive, such that the DNA elements are seen as attributes. We carefully define the security models for this cryptosystem, and we propose a CP-DBE construction that satisfies these security models. More precisely, we prove that the CP-DBE scheme is selectively IND-CCA secure and selectively collusion resistant in the random oracle.

6.2.2 Protocol Definition

A ciphertext-policy DNA-based encryption (CP-DBE) scheme consists of the following four algorithms:

- $\mathsf{Setup}(\lambda, \mathscr{U}) \to (params, msk)$. On inputs a security parameter λ and an attribute universe $\mathscr{U}$, output the public parameters *params* and the master secret key *msk*.

- $\mathsf{KeyGen}(params, msk, S) \to sk_S$. On inputs the public parameters *params*, the master secret key *msk* and an attribute set S, output a private key sk_S, such that sk_S is associated with the attribute set S.

- $\mathsf{Encrypt}(params, \mathbb{AS}, S', m) \to C_{\mathbb{AS}}$. On inputs the public parameters *params*, an access structure $\mathbb{AS}$ for attributes over $\mathscr{U}$, an attribute set S' satisfying the access structure $\mathbb{AS}$ and a message m, output a ciphertext $C_{\mathbb{AS}}$.

  We assume that the access structure $\mathbb{AS}$ is included in the ciphertext $C_{\mathbb{AS}}$.

- $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) \to m/\perp$. On inputs the public parameters *params*, an attribute set S, the corresponding private key sk_S and a ciphertext $C_{\mathbb{AS}}$, output m if $\mathscr{S}$ satisfies $\mathbb{AS}$, where $\mathscr{S}$ is defined as the intersection set between the attribute set S from the algorithm KeyGen and the attribute set S' from the algorithm Encrypt; output $\perp$ otherwise, indicating either $C_{\mathbb{AS}}$ is invalid or $\mathscr{S}$ does not satisfy $\mathbb{AS}$.

Correctness. We require that a ciphertext-policy DNA-based encryption scheme is correct if for any $\lambda \in \mathbb{N}$, $(params, msk) \leftarrow \mathsf{Setup}(\lambda, \mathscr{U})$, $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$ and $C_{\mathbb{AS}} \leftarrow \mathsf{Encrypt}(params, \mathbb{AS}, S', m)$, and if $\mathscr{S}$ satisfies $\mathbb{AS}$, we have that $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}}) = m$, where $\mathscr{S}$ is defined as the intersection set between the attribute set S from the algorithm KeyGen and the attribute set S' from the algorithm Encrypt.

6.2.3 Security Models

Selective Indistinguishability against Chosen-Ciphertext Attacks

In the following, we define the security notion for selective IND-CCA. A CP-DBE scheme is selectively IND-CCA secure if no PPT adversary $\mathscr{A}$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}$ as follows.

Initialization. The adversary $\mathscr{A}$ outputs a challenge access structure $\mathbb{AS}^*$ to the challenger $\mathscr{B}$.

Setup. The challenger $\mathscr{B}$ runs the algorithm Setup and gives the public parameters $params$ to the adversary $\mathscr{A}$.

Query Phase 1. The adversary $\mathscr{A}$ is given access to the following oracles:

1. *Private Key Query* $< S >$. On input an attribute set S, the challenger $\mathscr{B}$ runs $\mathsf{KeyGen}(params, msk, S)$ to obtain sk_S.
2. *Ciphertext Decryption Query* $< S, C_{\mathbb{AS}} >$. On inputs an attribute set S and a ciphertext $C_{\mathbb{AS}}$, the challenger $\mathscr{B}$ returns $m \leftarrow \mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{AS}})$ to the adversary $\mathscr{A}$, where $sk_S \leftarrow \mathsf{KeyGen}(params, msk, S)$ and $\mathscr{S}$ satisfies $\mathbb{AS}$ such that $\mathscr{S}$ is defined as the intersection set between the attribute set S from the algorithm KeyGen and the attribute set S' from the algorithm $\mathsf{Encrypt}$.
3. *Set Intersection Query* $< S, S', C_{\mathbb{AS}} >$. On inputs an attribute set S, another attribute set S' from the algorithm $\mathsf{Encrypt}$ and a ciphertext $C_{\mathbb{AS}} \leftarrow \mathsf{Encrypt}($ $params, \mathbb{AS}, S', m)$, the challenger $\mathscr{B}$ returns the intersection set $\mathscr{S} = S \cap S'$ to the adversary $\mathscr{A}$.

Note that if the queries to the ciphertext decryption oracle are invalid, then the challenger simply outputs $\perp$. In this phase, the following query is forbidden to issue: private key queries for any S satisfying $\mathbb{AS}^*$.

Challenge. The adversary $\mathscr{A}$ submits two messages m_0 and m_1 of equal length. The challenger $\mathscr{B}$ chooses a random bit $\mu \in_R \{0,1\}$ and encrypts m_μ under $\mathbb{AS}^*$. The ciphertext $C^*_{\mathbb{AS}^*}$ is given to the adversary $\mathscr{A}$.

Query Phase 2. The query phase 1 is repeated except that the following queries are forbidden to issue:

- private key queries for any S satisfying $\mathbb{AS}^*$,
- ciphertext decryption queries for any $\mathscr{S}$ satisfying $\mathbb{AS}^*$, where $\mathscr{S}$ is defined as the intersection set between the attribute set S from the algorithm KeyGen and the attribute set S' from the algorithm $\mathsf{Encrypt}$.

Guess. The adversary $\mathscr{A}$ outputs a guess $\mu' \in \{0,1\}$ for μ. If $\mu' = \mu$, then the adversary $\mathscr{A}$ wins.

The advantage of the adversary $\mathscr{A}$ is defined as $Adv^{S-IND-CCA}_{CP-DBE,\mathscr{A}}(\lambda, \mathscr{U}) = |Pr[\mu' = \mu] - 1/2|$.

Selective Collusion Resistance

In the following, we define the security notion for selective collusion resistance. A CP-DBE scheme is selectively collusion resistant if no PPT adversary $\mathscr{A}$ can win the game below with non-negligible advantage. Let $\mathscr{B}$ be a challenger that plays the game with the adversary $\mathscr{A}$ as follows:

Initialization. $\mathscr{A}$ outputs an attribute set S^* to $\mathscr{B}$.

Setup. $\mathscr{B}$ runs the algorithm Setup and gives the public parameters *params* to $\mathscr{A}$.

Query Phase. $\mathscr{A}$ is given access to the following oracles:

1. *Private Key Query* $< S >$. On input an attribute set $S \neq S^*$, $\mathscr{B}$ runs $\mathsf{KeyGen}($ $params, msk, S)$ to obtain sk_S and returns it to $\mathscr{A}$.
2. *Set Intersection Query* $< S, S', C_{\mathbb{AS}} >$. On inputs an attribute set $S \neq S^*$, another attribute set $S' \neq S^*$ from the algorithm Encrypt and a ciphertext $C_{\mathbb{AS}} \leftarrow \mathsf{Encrypt}(params, \mathbb{AS}, S', m)$, the challenger $\mathscr{B}$ returns the intersection set $\mathscr{S} = S \cap S'$ to the adversary $\mathscr{A}$.

Output. $\mathscr{A}$ submits a private key $sk^*_{S^*}$ for the attribute set S^*.

The advantage of $\mathscr{A}$ is defined as $Adv^{S-CollRes}_{CP-DBE,\mathscr{A}}(\lambda, \mathscr{U}) = Pr[\mathscr{A}$ succeeds$]$.

6.2.4 Construction

We now present a CP-DBE construction in the random oracle model. To build this CP-DBE construction, we start from the Waters' ABE scheme [237], that is proven selectively IND-CCA secure and collusion resistant. In Waters' scheme, the encryption algorithm takes as input a LSSS access matrix M and distributes a random exponent $c \in \mathbb{Z}_p$ according to M. The main advantage is the possibility to realize expressive functionality.

Then, we extend it to obtain a CP-DBE scheme with the same levels of security, namely IND-CCA security and collusion resistance in the random oracle model. Now, the receiver owns an attribute set S and is performs a set intersection operation on the sender's attribute set S', without learning any extra information except the output of the operation.

Let $\mathscr{U}$ be the attribute universe and $n' \cdot U$ be its cardinality (such that $n' \cdot U$ is assumed to be a large public number and $n' \geq 2$ an integer). Let S be an attribute set such that $S \subseteq \mathscr{U}$ and $|S| = 2 \cdot U$.

- $\mathsf{Setup}(\lambda, \mathscr{U}) \rightarrow (params, msk)$. Given a security parameter λ, run $(p, \mathbb{G}_1, \mathbb{G}_T, e) \leftarrow \mathscr{G}(\lambda)$. Choose two random values $\alpha, a \in_R \mathbb{Z}_p$, two random generators $g, g_1 \in_R \mathbb{G}_1$ and compute $h = g^a$. Pick at random U values $q_1, \cdots, q_U \in_R \mathbb{Z}_p$, and compute $Q_1 = g^{q_1}, \cdots, Q_U = g^{q_U}$. Moreover, set the following TCR hash functions $H_1 : \{0,1\}^{2\lambda} \rightarrow \mathbb{Z}_p$, $H_2 : \mathbb{G}_T \rightarrow \{0,1\}^{2\lambda}$, $H_3 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_4 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_5 : \mathbb{G}_T \rightarrow \mathbb{G}_1$.

  Finally, set the public parameters as $params = (p, \mathbb{G}_1, \mathbb{G}_T, e, g, g_1, h, e(g,g)^\alpha, Q_1, \cdots, Q_U, H_1, H_2, H_3, H_4, H_5)$ and the master secret key as $msk = (g^\alpha, q_1, \cdots, q_U)$.

- $\mathsf{KeyGen}(params, msk, S) \rightarrow sk_S$. Choose $t, \tilde{t} \in_R \mathbb{Z}_p$, compute $K = g^\alpha h^t$, $L = g^t$ and $K_x = H_3(x)^t$ for all $x \in S$. Then, compute $\tilde{Q} = g^{\tilde{t}}, \tilde{Q}_1 = g^{-q_1} h^{\tilde{t}}, \cdots, \tilde{Q}_U = g^{-q_U} h^{\tilde{t}}$ and set the private key $sk_S = (K, L, \{K_x\}_{x \in S}, \tilde{Q}, \tilde{Q}_1, \cdots, \tilde{Q}_U)$.

- $\mathsf{Encrypt}(params, \mathbb{AS}, S', m) \rightarrow C_{\mathbb{AS}}$. Let (M, ρ) be a LSSS access structure where M is a $l \times n$-matrix and the function ρ associates rows of M to attributes. Let the attribute set $S' = \{(x_{1,1},x_{1,2})\cdots,(x_{U,1},x_{U,2})\}$ satisfy (M,ρ) and the message m be in $\{0,1\}^{\lambda}$.

First, pick at random $N_{j,0}$ and $N_{j,1},N_{j,2}$ in $\mathbb{Z}_p$ for $j \in [1,U]$ (for $i \in [1,2]$, we suppose that the elements $N_{j,i}$ are not equal each other and not equal to $x_{j,1},x_{j,2}$). For $j \in [1,U]$, construct the polynomials $P_j(x) = N_{j,0}(x-x_{j,1})(x-x_{j,2})(x-N_{j,1})(x-N_{j,2}) = \sum_{i=0}^{4} \nu_{j,i}x^i$. Then, for $j \in [1,U], i \in [0,4]$, compute $g_{j,i} = g^{\nu_{j,i}}Q_j = g^{\nu_{j,i}+q_j}$ and $X_{j,i} = H_5(e(g^{\nu_{j,i}},g)) \oplus g^{\nu_{j,i}}$.

Second, choose $\beta \in_R \{0,1\}^{\lambda}$, set $c = H_1(m \parallel \beta)$ and a random vector $\vec{v} = (c, y_2, \cdots, y_n) \in_R \mathbb{Z}_p^n$, where $y_2,\cdots,y_n \in_R \mathbb{Z}_p$. For $i = [1,l]$, set $\lambda_i = v \cdot M_i$, where M_i is the vector corresponding to the i-th row of M and choose $r_1,\cdots,r_l \in_R \mathbb{Z}_p$. Then, set

$$
\begin{aligned}
&A_1 = (m \parallel \beta) \oplus H_2(e(g,g)^{\alpha c})\\
&A_2 = g^c\\
&A_3 = g_1^c,\\
&(B_1 = h^{\lambda_1}H_3(\rho(1))^{-r_1}, C_1 = g^{r_1}), \cdots, (B_l = h^{\lambda_l}H_3(\rho(l))^{-r_l}, C_1 = g^{r_l})\\
&D = H_4(A_1,A_3,(B_1,C_1),\cdots,(B_l,C_l),(M,\rho))^c
\end{aligned}
$$

Finally, output the ciphertext $C_{(M,\rho)} = ((M,\rho), A_1, A_2, A_3, (B_1,C_1), \cdots, (B_l,C_l), D, g_{1,0}, X_{1,0}), ((g_{1,1},X_{1,1}), (g_{1,2},X_{1,2}), (g_{1,3},X_{1,3}), (g_{1,4},X_{1,4}), \cdots, (g_{U,0},X_{U,0}), (g_{U,1},X_{U,1}), (g_{U,2},X_{U,2}), (g_{U,3},X_{U,3}), (g_{U,4},X_{U,4}))$.

Note that $\{\rho(i)\}_{i\in[1,l]}$ are the attributes used in the access structure (M,ρ). As in [237], we allow an attribute to be associated with multiple rows of matrix M, i.e. the function ρ is not injective.

- $\mathsf{Decrypt}(params, S, sk_S, C_{\mathbb{A}\mathbb{S}}) \to m/\perp$. Parse the ciphertext $C_{(M,\rho)}$ as $((M,\rho), A_1, A_2, A_3, (B_1,C_1), \cdots, (B_l,C_l), D, (g_{1,0},X_{1,0}), (g_{1,1},X_{1,1}), (g_{1,2},X_{1,2}), (g_{1,3},X_{1,3}), (g_{1,4},X_{1,4}), \cdots, (g_{U,0},X_{U,0}), (g_{U,1},X_{U,1}), (g_{U,2},X_{U,2}), (g_{U,3},X_{U,3}), (g_{U,4},X_{U,4}))$ and the private key sk_S for the attribute set S as $(K, L, \{K_x\}_{x\in S}, \tilde{Q}, \tilde{Q}_1, \cdots, \tilde{Q}_U)$.

  Let S' be the attribute set used in the algorithm Encrypt. First, start by setting the intersection set $\mathscr{S} = S \cap S'$. If $|S| > 2U$, then output $\perp$; otherwise proceed as follows. For $j \in [1,U]$ and $i \in [0,4]$, compute $g_{j,i}\tilde{Q}_j = g^{\nu_{j,i}+q_j}g^{-q_j}h^{\tilde{t}} = g^{\nu_{j,i}}h^{\tilde{t}}$ and

  $$Y = \frac{e(g^{\nu_{j,i}}h^{\tilde{t}}, g)}{e(\tilde{Q}, h)} = \frac{e(g^{\nu_{j,i}}g^{a\tilde{t}}, g)}{e(g^{\tilde{t}}, g^a)} = e(g^{\nu_{j,i}}, g).$$

  Finally, for $j \in [1,U]$ and $i \in [0,4]$, output $H_5(Y) \oplus X_{j,i} = H_5(e(g^{\nu_{j,i}},g)) \oplus H_5(e(g^{\nu_{j,i}}, g)) \oplus g^{\nu_{j,i}} = g^{\nu_{j,i}}$.

  Second, for $j \in [1,U]$, compute $g^{P_j(x)} = g^{\nu_{j,0}} \cdot (g^{\nu_{j,1}})^x \cdot (g^{\nu_{j,2}})^{x^2} \cdot (g^{\nu_{j,3}})^{x^3} \cdot (g^{\nu_{j,4}})^{x^4}$. For $j \in [1,U]$, for any pair $(\hat{x}_{j,1}, \hat{x}_{j,2})$ in S,

  - compute $g^{P_j(\hat{x}_{j,1})}$: if $g^{P_j(\hat{x}_{j,1})} = 1$ then $\hat{x}_{j,1} \in S'$ and stop; otherwise $\hat{x}_{j,1} \notin S'$ and proceed.
  - compute $g^{P_j(\hat{x}_{j,2})}$: if $g^{P_j(\hat{x}_{j,2})} = 1$ then $\hat{x}_{j,2} \in S'$ and stop; otherwise $\hat{x}_{j,2} \notin S'$ and output $\perp$.

  This process is repeated for all $j \in [1,U]$ until finally obtaining $\mathscr{S} = S' \cap S$. If $\mathscr{S}$ satisfies (M,ρ) and $|\mathscr{S}| \geq U$ such that for all $j \in [1,U]$, $(\hat{x}_{j,1} \in \mathscr{S} \vee \hat{x}_{j,2} \in \mathscr{S})$, then proceed; otherwise output $\perp$.

We recall that $I = \{i\}_{\rho(i) \in \mathscr{S}} \subseteq [1,l]$ and let $\{w_i\}_{i \in I}$ be the set of constants in $\mathbb{Z}_p$ such that $\sum_{i \in I} w_i \cdot \lambda_i = c$. Verify the validity of the ciphertext

$$\begin{aligned}
& e(A_2, g_1) \overset{?}{=} e(g, A_3) \\
& e(A_3, H_4(A_1, A_3, (B_1, C_1), \cdots, (B_l, C_l), (M, \rho))) \overset{?}{=} e(g_1, D) \\
& \mathscr{S} \text{ satisfies } (M, \rho) \text{ ?} \\
& e(\prod_{i \in I} B_i^{w_i}, g) \overset{?}{=} e(A_2, g^{\alpha}) \prod_{i \in I} e(C_i^{-1}, H_3(\rho(i))^{w_i})
\end{aligned} \tag{6.4}$$

If the equations 6.4 do not hold, then output $\perp$; otherwise, proceed as follows. Compute $T = e(A_2, K) / \prod_{i \in I}(e(B_i, L) \cdot e(C_i, K_{\rho(i)}))^{w_i}$ and $m \parallel \beta = H_2(T) \oplus A_1$. If $A_3 = g_1^{H_1(m \parallel \beta)}$, then output m; otherwise, output $\perp$.

Correctness. We first check the correctness of the intersection set. Let S' be the attribute set from the algorithm Encrypt and S be the attribute set from the algorithm KeyGen. For $j \in [1, U]$, for any pair $(\hat{x}_{j,1}, \hat{x}_{j,2}) \in S'$, if $g^{P_j(\hat{x}_{j,i})} = 1$, then $P_j(\hat{x}_{j,i}) = 0$ which means that $\hat{x}_{j,i}$ is a root of P_j and there is an attribute $x_{j,i'} \in S$ such that $\hat{x}_{j,i} = x_{j,i'}$, for $i' \in [1,2]$, and so we conclude that $\hat{x}_{j,i} \in S$. Reciprocally, if $\hat{x}_{j,i} \in S$, then there is an attribute $x_{j,i'} \in S$ such that $\hat{x}_{j,i} = x_{j,i'}$, for $i' \in [1,2]$, and $P_j(\hat{x}_{j,i}) = 0$, and so we conclude that $g^{P_j(\hat{x}_{j,i})} = 1$.

We then check the validity of the value T. Let $I = \{i\}_{\rho(i) \in \mathbb{S}} \subseteq [1,l]$ and $\{w_i\}_{i \in I}$ be the set of constants in $\mathbb{Z}_p$ such that $\sum_{i \in I} w_i \cdot \lambda_i = c$. We get:

$$\begin{aligned}
T &= \frac{e(A_2, K)}{\prod_{i \in I}(e(B_i, L) \cdot e(C_i, K_{\rho(i)}))^{w_i}} \\
&= \frac{e(g^c, g^{\alpha} h^t)}{\prod_{i \in I}(e(h^{\lambda_i} H_3(\rho(i))^{-r_i}, g^t) \cdot (g^{r_i}, H_3(\rho(i))^t)^{w_i}} \\
&= \frac{e(g^c, g^{\alpha} g^{at})}{e(g, g^{at})^{\sum_{i \in I} \lambda_i w_i}} = e(g^c, g^{\alpha})
\end{aligned}$$

Therefore, $H_2(T) \oplus A_1 = H_2(e(g^c, g^{\alpha})) \oplus (m \parallel \beta) \oplus H_2(e(g^c, g^{\alpha})) = m \parallel \beta$.

6.2.5 Security Proofs

Selective IND-CCA Security Proof

We start this section by giving an overview of the formal security analysis presented below. Note that this proof follows the one given in [145]. Let the challenge message be m_{μ} for a bit $\mu \in_R \{0,1\}$, and the challenge ciphertext be $C^*_{(M^*, \rho^*)} = ((M^*, \rho^*), A_1^*, A_2^*, A_3^*, (B_1^*, C_1^*), \cdots, (B_{l^*}^*, C_{l^*}^*), D^*, (g_{1,0}^*, X_{1,0}^*), \cdots, (g_{U,4}^*, X_{U,4}^*))$. Let $\mathscr{A}$ be an adversary that attacks the above scheme, following the constraints defined in the security model. $\mathscr{A}$ tries to guess the value of the bit μ by calling the ciphertext decryption oracle. To do so, $\mathscr{A}$ will modify the challenge ciphertext $C^*_{(M^*, \rho^*)}$ and submit the resulting ciphertext to the ciphertext decryption oracle. Since $A_1^*, A_3^*, (B_1^*, C_1^*), \cdots, (B_{l^*}^*, C_{l^*}^*)$ are bound by the element D^*, as well as the description of (M^*, ρ^*), modifying the ciphertext is noticeable with non-negligible probability from the equations 6.4. Indeed, note that we can view D^* as a signature on these components. In addition, observe that the integrity of A_2^* is bound by A_3^*. Hence, if the ciphertext is modified, then the equations 6.4 do not hold. Therefore, $\mathscr{A}$ is not given any special advantage in guessing the bit μ.

Theorem. Suppose that the s-DPBDHE assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$ and H_1, H_2, H_3, H_4 and H_5 are TCR hash functions, then the CP-DBE scheme is selectively IND-CCA secure in the random oracle model.

Suppose there is an adversary $\mathscr{A}$ who can break the selective IND-CCA security of the CP-DBE scheme. We then construct a challenger $\mathscr{B}$ that can decide whether Z is either equal to $e(g,g)^{a^{s+1} \cdot c}$ or to a random element in $\mathbb{G}_T$. The simulator $\mathscr{B}$ plays the selective IND-CCA game with $\mathscr{A}$ as follows.

$\mathscr{B}$ takes as inputs $(p, \mathbb{G}_1, \mathbb{G}_T, e) \leftarrow \mathscr{G}(\lambda)$, a generator g of $\mathbb{G}_1$, and a s-DPBDHE instance

$$\begin{aligned}
&g^c, g_1^a, \cdots, g^{a^s}, g^{a^{s+2}}, \cdots, g^{a^{2s}} \\
&\forall j \in [1,s], \quad g^{c \cdot b_j}, g^{a/b_j}, \cdots, g^{a^s/b_j}, g^{a^{s+2}/b_j}, \cdots, g^{a^{2s}/b_j} \\
&\forall j,k \in [1,s], k \neq j, \quad g^{a \cdot c \cdot b_k/b_j}, \cdots, g^{a^s \cdot s \cdot b_k/b_j}
\end{aligned}$$

and Z which is either equal to $e(g,g)^{a^{s+1} \cdot c}$ or to a random element in $\mathbb{G}_T$.

Initialization. The adversary gives the challenge access structure (M^*, ρ^*) to $\mathscr{B}$, where M^* has l^* rows and n^* columns such that $l^*, n^* \leq s$.

Setup. The challenger chooses at random $\alpha', \gamma \in_R \mathbb{Z}_p$, sets $g_1 = g^\gamma$ and implicitly sets $\alpha = \alpha' + a^{s+1}$ by letting $e(g,g)^\alpha = e(g^a, g^{a^s}) e(g,g)^{\alpha'}$ ($\alpha = \alpha' + a^{s+1}$ cannot be computed by $\mathscr{B}$).

Then $\mathscr{B}$ chooses the five TCR hash functions H_1, H_2, H_3, H_4, H_5 and the exponents $q_1, \cdots, q_U$ as in the real scheme. It sends the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_T, e, g, g_1, h = g^a, e(g,g)^\alpha, Q_1 = g^{q_1}, \cdots, Q_U = g^{q_U}, H_1, H_2, H_3, H_4, H_5)$ to $\mathscr{A}$. We note that the public parameters are identical to those in the real scheme, for the adversary's view. At any time, $\mathscr{A}$ can adaptively query the random oracles H_j, for $j \in [1,5]$, which are controlled by $\mathscr{B}$. The challenger maintains the lists H_j^{List} for $j \in [1,5]$, which are initially empty, and answers the queries to the aforementioned random oracles as follows:

- H_1: on receipt of a H_1 query on (m, β), if there is a tuple $(m, \beta, c) \in L_{H_1}$, $\mathscr{B}$ forwards the predefined value c to $\mathscr{A}$, where $c \in \mathbb{Z}_p$. Otherwise, $\mathscr{B}$ sets $H_1(m, \beta) = c$, responds c to $\mathscr{A}$ and adds the tuple (m, β, c) to L_{H_1}, where $c \in_R \mathbb{Z}_p$.
- H_2: on receipt of a H_2 query on $R \in \mathbb{G}_T$, if there is a tuple $(R, \delta_1) \in L_{H_2}$, $\mathscr{B}$ forwards the predefined value δ_1 to $\mathscr{A}$, where $\delta_1 \in \{0,1\}^{2\lambda}$. Otherwise, $\mathscr{B}$ sets $H_2(R) = \delta_1$, responds δ_1 to $\mathscr{A}$ and adds the tuple (R, δ_1) to L_{H_2}, where $\delta_1 \in_R \{0,1\}^{2\lambda}$.
- H_3: on receipt of a H_3 query on $x \in \mathscr{U}$, if there is a tuple $(x, z_x, \delta_{2,x}) \in L_{H_3}$, $\mathscr{B}$ forwards the predefined value $\delta_{2,x}$ to $\mathscr{A}$, where $z_x \in \mathbb{Z}_p$ and $\delta_{2,x} \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ constructs $\delta_{2,x}$ as follows. Let X denote the set of indices i such that $\rho^*(i) = x$ where $i \in [1, l^*]$. Namely, X contains the indices of rows of matrix M^* that corresponds to the same attribute x. $\mathscr{B}$ chooses $z_x \in_R \mathbb{Z}_p$ and sets
$$\delta_{2,x} = g^{z_x} \prod_{i \in X} g^{a M^*_{i,1}/b_i + a^2 M^*_{i,2}/b_i + \cdots + a^{n^*} M^*_{i,n^*}/b_i}.$$

If $X = \emptyset$, $\mathscr{B}$ sets $\delta_{2,x} = g^{z_x}$. $\mathscr{B}$ responds $\delta_{2,x}$ to $\mathscr{A}$ and adds the tuple $(x, z_x, \delta_{2,x})$ to L_{H_3}.

- H_4: on receipt of a H_4 query on $(A_1, A_3, (B_1, C_1), \cdots, (B_l, C_l), (M, \rho))$, if there is a tuple $(A_1, A_3, (B_1, C_1), \cdots, (B_l, C_l), (M, \rho), \xi_1, \delta_3) \in L_{H_4}$, $\mathscr{B}$ forwards the predefined value δ_3 to $\mathscr{A}$, where $\xi_1 \in \mathbb{Z}_p$ and $\delta_3 \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ sets $\delta_3 = g^{\xi_1}$, responds δ_3 to $\mathscr{A}$ and adds the tuple $(A_1, A_3, (B_1, C_1), \cdots, (B_l, C_l), (M, \rho), \xi_1, \delta_3)$ in L_{H_4}, where $\xi_1 \in_R \mathbb{Z}_p$.
- H_5: on receipt of a H_5 query on $\delta \in \mathbb{G}_T$, if there is a tuple $(\delta, \xi_2) \in L_{H_5}$, $\mathscr{B}$ forwards the predefined value ξ_2 to $\mathscr{A}$, where $\xi_2 \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ sets $H_5(\delta) = \xi_2$ to $\mathscr{A}$ and adds the tuple (δ, ξ_2) to L_{H_5}, where $\xi_2 \in_R \mathbb{G}_1$.

In addition, $\mathscr{B}$ maintains the list L_{sk} which is initially empty as follows: L_{sk} records the tuples (S, sk_S) which are the results of the queries to the private key oracle.

Query Phase 1. The challenger answers to $\mathscr{A}$'s queries as follows.

- *Private Key Query* $< S >$. We suppose that a private key sk_S for an attribute set S is given to $\mathscr{B}$, where S does not satisfy M^* (if S satisfies M^* then $\mathscr{B}$ outputs a random bit μ in $\{0,1\}$ and aborts the simulation).

  First, the challenger chooses $r \in_R \mathbb{Z}_p$ at random. Then, $\mathscr{B}$ finds a vector $\vec{w} = (w_1, \cdots, w_{n^*}) \in \mathbb{Z}_p^{n^*}$ such that $w_1 = -1$ and for all i such that $\rho^*(i) \in S$, we have that $\vec{w} \cdot M_i^* = 0$. By the definition of a LSSS, such a vector must exist. We note that if such a vector did not exist, then the vector $(1, 0, \cdots, 0)$ would be in the span of S.

  Second, $\mathscr{B}$ defines implicitly the exponent t as $r + w_1 a^s + w_2 a^{s-1} + \cdots + w_{n^*} a^{s-n^*+1}$, by setting $L = g^r \prod_{i \in [1,n^*]} (g^{a^{s+1-i}})^{w_i} = g^t$. By definition of the exponent t, $h^t = g^{at}$ contains a term of $g^{-a^{s+1}}$. This term will cancel out with the unknown term in g^{α} when K will be computed. Thus, $\mathscr{B}$ creates K as

$$\begin{aligned} K &= g^{\alpha'} g^{ar} \prod_{i=2}^{n^*} (g^{a^{s+2-i}})^{w_i} \\ &= g^{\alpha'} g^{a^{s+1}} g^{-a^{s+1}} \prod_{i=2}^{n^*} (g^{a^{s+2-i}})^{w_i} \\ &= g^{\alpha} (g^r \prod_{i=1,\cdots,n^*} (g^{a^{s+2-i}})^{w_i})^a \\ &= g^{\alpha} L^a = g^{\alpha} g^{at} = g^{\alpha} h^t \end{aligned}$$

  Third, the simulator calculates K_x for all $x \in S$. There is $x \in S$ for which there is no i such that $\rho^*(i) = x$, and so we compute $K_x = L^{z_x}$. Therefore, we obtain that $K_x = L^{z_x} = \delta_{2,x}^t = H_3(x)^t$.

  Nevertheless, it remains more difficult to compute the key components K_x for attributes $x \in S$ for which there is one i such that $\rho^*(i) = x$ (note that x is used in the access structure). We need to check that there is no term of the form g^{a^{s+1}/b_i} that cannot be simulated; if we have $M_i^* \cdot \vec{w} = 0$, then all of these terms will cancel out.

We define X as $\{i\}_{\rho^*(i)=x}$. Thus, $\mathscr{B}$ creates K_x as follows:

$$\begin{aligned}
K_x &= L^{z_x} \prod_{i\in X} \prod_{j=1}^{n^*} \left(g^{\frac{a^j}{b_i} r} \prod_{\substack{k\in[1,n^*]\\ k\neq j}} (g^{a^{s+1+j-k}/b_i})^{w_k} \right)^{M^*_{i,j}} \\
&= L^{z_x} \prod_{i\in X} \prod_{j=1}^{n^*} \left(g^{\frac{a^j}{b_i} r} \prod_{\substack{k\in[1,n^*]\\ k\neq j}} (g^{a^{s+1+j-k}/b_i})^{w_k} \right)^{M^*_{i,j}} \prod_{i\in X} \prod_{j=1}^{n^*} \left(g^{(a^{s+1}/b_i)} \right)^{w_j M^*_{i,j}} \\
&= \left(g^r \prod_{i=1}^{n^*} g^{(a^{s+1-i} w_i)} \right)^{z_x} \prod_{i\in X} \prod_{j=1}^{n^*} \left(g^{\frac{a^j}{b_i} r} \prod_{k=1}^{n^*} (g^{a^{s+1+j-k}/b_i})^{w_k} \right)^{M^*_{i,j}} \\
&= \left(g^r \prod_{i\in X} g^{a \frac{M^*_{i,1}}{b_i} + a^2 \frac{M^*_{i,2}}{b_i} + \cdots + a^{n^*} \frac{M^*_{i,n^*}}{b_i}} \right)^{\kappa} \text{ such that } \kappa = (r + \sum_{i=1}^{n^*} w_i a^{s-i+1}) \\
&= \delta_{2,x}^{(r + w_1 a^q + \cdots + w_{n^*} a^{s-n^*+1})} = \delta_{2,x}^t = H_3(x)^t
\end{aligned}$$

We recall that if S does not satisfy (M^*,ρ^*), then $w \cdot M^*_i = 0$.

Thus, we have $\prod_{i\in X} \prod_{j=1}^{n^*} (g^{a^{s+1}/b_i})^{w_j M^*_{i,j}} = g^{a^{s+1}(\sum_{i\in X} \sum_{j=1}^{n^*} w_j M^*_{i,j}/b_i)} = g^0 = 1$.

In addition, the challenger chooses $\tilde{t} \in_R \mathbb{Z}_p$, and computes $\tilde{Q}$ and $\tilde{Q}_1, \cdots, \tilde{Q}_U$ as in the real scheme. Finally, $\mathscr{B}$ adds the tuple (S, sk_S) to L_{sk} and returns sk_S to the adversary.

- *Ciphertext Decryption Query* $< S, CT_{(M,\rho)} >$. The challenger checks whether the equation 6.4 holds. If not, then either the ciphertext is invalid or $\mathscr{S} = S \cap S'$ does not satisfy (M,ρ), such that S' is the attribute set from the algorithm Encrypt, and $\mathscr{B}$ outputs $\perp$. Otherwise, $\mathscr{B}$ proceeds as follows.
  - If $(S, sk_S) \in L_{sk}$ for any S such that $\mathscr{S} = S \cap S'$ satisfying (M,ρ), $\mathscr{B}$ recovers m as in the real scheme using sk_S.
  - Otherwise, $\mathscr{B}$ verifies whether $(m,\beta,c) \in L_{H_1}$ and $(R,\delta_1) \in L_{H_2}$ such that $A_3 = g_1^c$, $A_1 = (m \parallel \beta) \oplus \delta_1$ and $R = e(g,g)^{\alpha c}$. If no such tuple exists, the challenger outputs $\perp$; otherwise, it outputs m.
- *Set Intersection Query* $< S, S', C_{(M,\rho)} >$. The challenger parses the ciphertext $C_{(M,\rho)}$ as $((M,\rho), A_1, A_2, A_3, (B_1,C_1), \cdots, (B_l,C_l), D, (g_{1,0}, X_{1,0}), \cdots, (g_{U,4}, X_{U,4}))$.
  - If S does not satisfy (M,ρ) then the $\mathscr{B}$ outputs $\perp$. Note that S' is the attribute set used to define (M,ρ) in the algorithm Encrypt.
  - Otherwise, $\mathscr{B}$ constructs $\mathscr{S}$ as in the real scheme. If $\mathscr{S} = S \cap S'$ does not satisfy (M,ρ), then the challenger outputs $\perp$. If $|\mathscr{S}| \geq U$, then the challenger outputs $\mathbb{S}$.

Challenge. The adversary gives two messages m_0 and m_1 of equal length to the challenger. $\mathscr{B}$ picks at random a bit $\mu \in_R \{0,1\}$. For each row i of the matrix M^*, $\mathscr{B}$ sets $x^* = \rho^*(i)$ and issues a H_3 query on x^* to obtain the tuple $(x^*, z_{x^*}, \delta_{2,x^*})$.

Let us focus on the simulation of the values B_i since the terms that must be canceled out are contained in them. Nevertheless, $\mathscr{B}$ can choose a secret c to be split in order to cancel out these terms. In other words, $\mathscr{B}$ chooses at random $y'_2, \cdots, y'_{n^*}$ and shares the secret c using the vector $\vec{v} = (c, ca + y'_2, \cdots, ca^{n-1} + y'_{n^*}) \in \mathbb{Z}_p^{n^*}$. Moreover, values $r'_1, \cdots, r'_l$ are randomly chosen.

For $i \in [1, n^*]$, let R_i be $\{k \neq i$ such that $\rho^*(k) = \rho^*(i)\}$. Intuitively, R_i is the set of all other row indices that have the same attribute as row index i. Then, we generate the components of the challenge ciphertext as follows:

$$\begin{aligned} B_i &= \delta_{2,x^*}^{-r'_i} \cdot (\prod_{j=2}^{n^*} h^{M^*_{i,j} y'_j}) \cdot g^{cb_i \cdot (-z_{x^*})} \cdot \left(\prod_{k \in R_i} \prod_{j=1}^{n^*} (g^{a^j c (b_i / b_k)})^{M^*_{k,j}} \right)^{-1} \\ C_i &= g^{r'_i} g^{cb_i} \end{aligned}$$

Then, $\mathscr{B}$ chooses $\beta^* \in_R \{0,1\}^{\lambda}$ and $A_1^* \in_R \{0,1\}^{2\lambda}$ and sets $H_2(Z \cdot e(g^c, g^{\alpha'})) = A_1^* \oplus (m_\mu \parallel \beta^*)$. It finally computes $A_2^* = g^c$ and $A_3^* = (g^c)^{\gamma}$.

Moreover, $\mathscr{B}$ issues a H_4 query on $A_1^*, A_3^*, (B_1^*, C_1^*), \cdots, (B_{l^*}^*, C_{l^*}^*), (M^*, \rho^*)$ to obtain $(A_1^*, A_3^*, (B_1^*, C_1^*), \cdots, (B_{l^*}^*, C_{l^*}^*), (M^*, \rho^*), \xi_1^*, \delta_3^*)$, and sets $D^* = (g^c)^{\xi_1^*}$.

Thereafter, the challenger computes the values $g_{j,i}^*$ as follows. From the attribute set $S^* = \{(x_{1,1}^*, x_{1,2}^*), \cdots, (x_{U,1}^*, x_{U,2}^*)\}$, it picks at random $N_{j,0}^*$ and $N_{j,1}^*, N_{j,2}^*$ in $\mathbb{Z}_p$, for $j \in [1, U]$. Then, for $j \in [1, U]$, it computes the polynomials $P_j^*(x) = N_{j,0}^*(x - x_{j,1}^*)(x - x_{j,2}^*)(x - N_{j,1}^*)(x - N_{j,2}^*) = \sum_{i=0}^{4} \nu_{j,i}^* x^i$. Finally, for $j \in [1, U]$ and $i \in [0, 4]$, it computes $g_{j,i}^* = g^{\nu_{j,i}^*} Q_j$, issues an H_5 query on $\delta^* = e(g^{\nu_{j,i}^*}, g)$ to obtain (δ^*, ξ_2^*) and sets $X_{j,i}^* = \xi_2^* \oplus g^{\nu_{j,i}^*}$.

Eventually, $\mathscr{B}$ outputs the challenge ciphertext $C^*_{(M^*, \rho^*)} = ((M^*, \rho^*), A_1^*, A_2^*, A_3^*, (B_1^*, C_1^*), \cdots, (B_{l^*}^*, C_{l^*}^*), D^*, (g_{1,0}^*, X_{1,0}^*), \cdots, (g_{U,4}^*, X_{U,4}^*))$ and gives it to the adversary.

Observe that if $Z = e(g, g)^{a^{s+1} \cdot c}$, then $C^*_{(M^*, \rho^*)}$ is a valid ciphertext. By letting

$H_1(m_\mu, \beta^*) = c$ and $r_i = r'_i + cb_i$, we can check:

$$\begin{aligned}
A_1^* &= A_1^* \oplus (m_\mu \parallel \beta^*) \oplus (m_\mu \parallel \beta^*) = H_2(Ze(g^c, g^{\alpha'})) \oplus (m_\mu \parallel \beta^*) \\
&= H_2(e(g,g)^{\alpha c}, ZK^*) \oplus (m_\mu \parallel \beta^*) \\
A_2^* &= g^c \\
A_3^* &= g^{c\gamma} = g_1^c \\
D^* &= (g^c)^{\xi_1^*} = H_4(A_1^*, A_3^*, (B_1^*, C_1^*), \cdots, (B_{l^*}^*, C_{l^*}^*), (M^*, \rho^*))^c \\
B_i^* &= \delta_{2,x^*}^{-r'_i} \cdot (\prod_{j=2}^{n^*} h^{M_{i,j}^* y'_j}) \cdot g^{-z_x^* cb_i} \cdot (\prod_{k \in R_i} \prod_{j=1}^{n^*} g^{a^j c(b_i/b_k) M_{k,j}^*})^{-1} \\
&= \delta_{2,x^*}^{-r'_i} \cdot (\prod_{j=2}^{n^*} g^{aM_{i,j}^* y'_j}) \cdot (\prod_{j=1}^{n^*} g^{a^j cM_{i,j}^*}) \cdot (\prod_{j=1}^{n^*} g^{a^j cM_{i,j}^*})^{-1} g^{-z_x^* cb_i}) \\
&\quad \cdot (\prod_{k \in R_i} \prod_{j=1}^{n^*} g^{a^j c(b_i/b_k) M_{k,j}^*})^{-1} \\
&= \delta_{2,x^*}^{-r'_i} g^{a\lambda_i} \cdot (\prod_{j=1}^{n^*} g^{a^j cM_{i,j}^*})^{-1} \cdot g^{-z_x^* cb_i} \cdot (\prod_{k \in R_i} \prod_{j=1}^{n^*} g^{a^j s(b_i/b_k) M_{k,j}^*})^{-1} \\
&= g^{a\lambda_i} g^{-r'_i z_x^*} g^{-z_x^* cb_i} \cdot (\prod_{i \in X} g^{aM_{i,1}^*/b_i + a^2 M_{i,2}^*/b_i + \cdots + a^{n^*} M_{i,n^*}^*/b_i})^{-r'_i} \cdot (\prod_{j=1}^{n^*} g^{a^j cM_{i,j}^*})^{-1} \\
&\quad \cdot (\prod_{k \in R_i} \prod_{j=1}^{n^*} g^{a^j c(b_i/b_k) M_{k,j}^*})^{-1} \\
&= g^{a\delta_i} \delta_{2,x^*}^{-r'_i - cb_i} = g^{a\delta_i} \delta_{2,x^*}^{-r_i} = g^{a\delta_i} H_3(x^*)^{-r_i} \\
&= g^{a\delta_i} H_3(\rho^*(i))^{-r_i} = h^{\delta_i} H_3(\rho^*(i))^{-r_i} \\
C_i^* &= g^{r'_i + cb_i} = g^{r_i}
\end{aligned}$$

Nevertheless, if $Z \in_R \mathbb{G}_T$, then the challenge ciphertext is independent of the bit μ for $\mathscr{A}$'s view.

Query Phase 2. Same as in the query phase 1, except that we consider the constraints defined in Section 6.2.3.

Guess. The adversary will eventually output a guess $\mu' \in \{0,1\}$ for μ. If $\mu' = \mu$, the challenger then outputs 0 to guess that $Z = e(g,g)^{a^{s+1}c}$; otherwise, it outputs 1 to indicate that it believes Z is a random group element in $\mathbb{G}_T$.

Analysis. The simulations of the random oracles are perfect except for H_1 and H_2. Let H_1^* and H_2^* be the events that $\mathscr{A}$ has queried (m_μ, β^*) to H_1 (with probability of successful query $Adv_{H_1^*,\mathscr{A}}^{TCR}$) and $R^* = e(g,g)^{\alpha c}$ to H_2 (with probability of successful query $Adv_{H_2^*,\mathscr{A}}^{TCR}$) before the challenge phase.

In the simulation of the private key oracle, the responses to $\mathscr{A}$ are perfect. In the simulation of the ciphertext decryption oracle, it might be possible that the challenger cannot provide a decryption for a valid ciphertext. Suppose that the adversary can generate a valid ciphertext without querying $e(g,g)^{\alpha c}$ to H_2 such that $c = H_1(m, \beta)$. Let $Valid$ be the event that the ciphertext is valid, $QueryH_1$ be the event that the adversary has queried (m, β)

to H_1 and $QueryH_2$ be the event that the adversary has queried $e(g,g)^{\alpha c}$ to H_2. From the simulation, we get that $Pr[Valid \mid \neg QueryH_2] \leq Pr[QueryH_1 \mid QueryH_2] + Pr[Valid \mid QueryH_1 \wedge \neg QueryH_2] \leq q_{H_1}/2^{2\lambda} + 1/p$ and $Pr[Valid \mid \neg QueryH_1] \leq q_{H_2}/2^{2\lambda} + 1/p$, where q_{H_1} and q_{H_2} are the total numbers of queries to the random oracles H_1 and H_2 respectively. Let $Pr[DecErr]$ be the probability that the event $Valid \mid (\neg QueryH_1 \vee \neg QueryH_2)$ occurs, then we obtain that $Pr[DecErr] \leq (\frac{q_{H_1}+q_{H_2}}{2^{2\lambda}} + \frac{2}{p})q_d$, where q_d is the total number of queries to the ciphertext decryption oracle.

Let Bad denote the event that $(H_1^* \mid \neg H_2^*) \vee H_2^* \vee DecErr$, then

$$\begin{aligned} Adv^{S-IND-CCA}_{CP-DBE,\mathscr{A}} &= \mid Pr[\mu' = \mu] - 1/2 \mid \leq \frac{1}{2}Pr[Bad] = \frac{1}{2}Pr[(H_1^* \mid \neg H_2^*) \vee H_2^* \vee DecErr] \\ &\leq \frac{1}{2}(Adv^{TCR}_{H_2^*,\mathscr{A}} + \frac{2q_d}{p} + \frac{q_{H_1} + (q_{H_1} + q_{H_2})q_d}{2^{2\lambda}}) \end{aligned}$$

Therefore, $Adv^{DBDHE}_{\mathscr{B}} \geq \frac{1}{q_{H_2}} Adv^{TCR}_{H_2^*,\mathscr{A}} \geq \frac{1}{q_{H_2}}(2Adv^{S-IND-CCA}_{CP-DBE,\mathscr{A}} - \frac{q_{H_1}+(q_{H_1}+q_{H_2})q_d}{2^{2k}} - \frac{2q_d}{p})$.

Selective Collusion Resistance Security Proof

As in Waters ABE scheme [237], one of the main challenges in designing our scheme is to prevent its security against colluding user attacks.

The receiver owns a private key which is associated with an attribute set S. This receiver can decrypt a ciphertext if and only if the access structure associated with the ciphertext is satisfied by his/her attributes. In order to avoid collusions in our construction, each key should be randomized with a freshly chosen exponent t in $\mathbb{Z}_p$. During the decryption process, each share of the secret is multiplied by t in the exponent. This factor should bind the components of one receiver's key together, and so combining them with another receiver's key components is not possible. Thus, these randomized shares are only useful for one given key.

In addition, other collusion attacks could occur when constructing the intersection set of attributes $\mathscr{S}$: several receivers could combine their sets in order to find enough elements to determine $\mathscr{S}$. For instance, we may imagine that some grandchildren try to decrypt the message of their grandparent (instead of a relation child-parent). In order to avoid such attacks, a receiver's private key receives extra elements that are randomized with another freshly chosen exponent $\tilde{t}$ in $\mathbb{Z}_p$. These elements will be useful to construct $\mathscr{S}$ during the decryption process, since they should enable tone receiver to recover the coefficients of the polynomials constructed from the sender's attributes set and to test whether the elements in the receiver's attribute set are roots of these polynomials.

Theorem. Suppose that the BDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$ and H_1, H_2, H_3, H_4 and H_5 are TCR hash functions, then the CP-DBE scheme is selectively collusion resistant in the random oracle model.

Suppose there is an adversary $\mathscr{A}$ who breaks the collusion resistance of the CP-DBE scheme with advantage $Adv^{S-CollRes}_{CP-DBE,\mathscr{A}}$ greater than ε. We then construct a challenger $\mathscr{B}$ to output Z equal to $e(g,g)^{abc}$. The challenger $\mathscr{B}$ plays the collusion resistance game with $\mathscr{A}$ as follows.

$\mathscr{B}$ takes as inputs $(p, \mathbb{G}_1, \mathbb{G}_T, e)$, a generator g of $\mathbb{G}_1$ and a BDH problem instance (g^a, g^b, g^c) for some unknown exponents $a, b, c \in \mathbb{Z}_p$. The goal of the challenger is to output g^{ab}.

Initialization. The adversary gives the challenge set S^* to $\mathscr{B}$.

Setup. The challenger sets $h = g^a$ and $e(g,g)^\alpha = e(g^a, g^b) = e(g,g)^{ab}$. It also chooses at random $g_1 \in_R \mathbb{G}_1$.

Then $\mathscr{B}$ chooses the five TCR hash functions H_1, H_2, H_3, H_4, H_5 and the exponents $q_1, \cdots, q_U$ as in the real scheme. It sends the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_T, e, g, g_1, h = g^a, e(g,g)^\alpha, Q_1 = g^{q_1}, \cdots, Q_U = g^{q_U}, H_1, H_2, H_3, H_4, H_5)$ to $\mathscr{A}$. We note that the public parameters are identical to those in the real scheme, for the adversary's view. At any time, $\mathscr{A}$ can adaptively query the random oracles H_j for $j \in [1,5]$, which are controlled by $\mathscr{B}$. The challenger maintains the lists H_j^{List} for $j \in [1,5]$, which are initially empty, and answers the queries to the aforementioned random oracles as follows:

- H_1: on receipt of a H_1 query on (m, β), if there is a tuple $(m, \beta, c) \in L_{H_1}$, $\mathscr{B}$ forwards the predefined value c to $\mathscr{A}$, where $c \in \mathbb{Z}_p$. Otherwise, $\mathscr{B}$ sets $H_1(m, \beta) = c$, responds c to $\mathscr{A}$ and adds the tuple (m, β, c) to L_{H_1}, where $c \in_R \mathbb{Z}_p$.
- H_2: on receipt of a H_2 query on $R \in \mathbb{G}_T$, if there is a tuple $(R, \delta_1) \in L_{H_2}$, $\mathscr{B}$ forwards the predefined value δ_1 to $\mathscr{A}$, where $\delta_1 \in \{0,1\}^{2\lambda}$. Otherwise, $\mathscr{B}$ sets $H_2(R) = \delta_1$, responds δ_1 to $\mathscr{A}$ and adds the tuple (R, δ_1) to L_{H_2}, where $\delta_1 \in_R \{0,1\}^{2\lambda}$.
- H_3: on receipt of a H_3 query on $x \in \mathscr{U}$, if there is a tuple $(x, z_x, \delta_{2,x}) \in L_{H_3}$, $\mathscr{B}$ forwards the predefined value $\delta_{2,x}$ to $\mathscr{A}$, where $z_x \in \mathbb{Z}_p$ and $\delta_{2,x} \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ constructs $\delta_{2,x}$ as follows. $\mathscr{B}$ sets $\delta_{2,x} = g^{z_x}$, responds $\delta_{2,x}$ to $\mathscr{A}$ and adds the tuple $(x, z_x, \delta_{2,x})$ to L_{H_3}.
- H_4: on receipt of a H_4 query on $(A_1, A_3, (B_1, C_1), \cdots, (B_l, C_l), (M, \rho))$, if there is a tuple $(A_1, A_3, (B_1, C_1), \cdots, (B_l, C_l), (M, \rho), \xi_1, \delta_3) \in L_{H_4}$, $\mathscr{B}$ forwards the predefined value δ_3 to $\mathscr{A}$, where $\xi_1 \in \mathbb{Z}_p$ and $\delta_3 \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ sets $\delta_3 = g^{\xi_1}$, responds δ_3 to $\mathscr{A}$ and adds the tuple $(A_1, A_3, (B_1, C_1), \cdots, (B_l, C_l), (M, \rho), \xi_1, \delta_3)$ in L_{H_4}, where $\xi_1 \in_R \mathbb{Z}_p$.
- H_5: on receipt of a H_5 query on $\delta \in \mathbb{G}_T$, if there is a tuple $(\delta, \xi_2) \in L_{H_5}$, $\mathscr{B}$ forwards the predefined value ξ_2 to $\mathscr{A}$, where $\xi_2 \in \mathbb{G}_1$. Otherwise, $\mathscr{B}$ sets $H_5(\delta) = \xi_2$ to $\mathscr{A}$ and adds the tuple (δ, ξ_2) to L_{H_5}, where $\xi_2 \in_R \mathbb{G}_1$.

In addition, $\mathscr{B}$ maintains the list L_{sk} which is initially empty as follows: L_{sk} records the tuples (S, sk_S) which are the results of the queries to the private key oracle.

Query Phase. The challenger answers to $\mathscr{A}$'s queries as follows.

1. *Private Key Query* $< S >$. Let S be an attribute set such that $S \neq S^*$. We suppose there is an attribute y such that either $(y \in S \wedge y \notin S^*)$ or $(y \notin S \wedge y \in S^*)$. Without loss of generality, we assume that $(y \notin S \wedge y \in S^*)$. Then, $\mathscr{B}$ picks at random $\{t'_x\}_{x \in S}, \tilde{t}$ in $\mathbb{Z}_p$ and implicitly sets the elements $\{t_x\}_{x \in S}, t$ as

follows:

$$\begin{aligned} t_x &= t'_x, \text{ for } x \in S, \quad x \neq y \\ t_y &= -b + t'_y \\ t &= \sum_{x\in S} t_x = -b + \sum_{x \in S} t'_x \end{aligned}$$

Therefore, the components of the private key sk_S can be computed as follows:

$$\begin{aligned} K &= g^{\alpha} h^{t} = g^{ab} g^{a(-b+\sum_{x\in S} t'_x)} = g^{\sum_{x\in S} t'_x} \\ L &= g^{t} = g^{-b+\sum_{x\in S} t'_x} \\ K_x &= L^{z_x} = g^{z_x(-b+\sum_{x\in S} t'_x)} \\ \tilde{Q} &= g^{\tilde{t}} \\ \tilde{Q}_j &= g^{-q_j} h^{\tilde{t}} = g^{-q_j} g^{a\tilde{t}}, \text{ for } j \in [1,U] \end{aligned}$$

such that $K_x = L^{z_x} = \delta_{2,x}^{t} = H_3(x)^t$.

2. *Set Intersection Query* $< S, S', C_{(M,\rho)} >$. The challenger parses the ciphertext $C_{(M,\rho)}$ as $((M,\rho),A_1,A_2,A_3,(B_1,C_1),\cdots,(B_l,C_l),D,(g_{1,0},X_{1,0}),\cdots,(g_{U,4},X_{U,4}))$.
   - If S does not satisfy (M,ρ) then the $\mathscr{B}$ outputs $\perp$. Note that S' is the attribute set used to define (M,ρ) in the algorithm Encrypt.
   - Otherwise, $\mathscr{B}$ constructs $\mathscr{S}$ as in the real scheme. If $\mathscr{S} = S \cap S'$ does not satisfy (M,ρ), then the challenger outputs $\perp$. If $|\mathscr{S}| \geq U$, then the challenger outputs $\mathbb{S}$.

Output. The challenge private key $sk^*_{S^*} = (K^*, L^*, \{K^*_x\}_{x\in S^*}, \tilde{Q}^*, \tilde{Q}^*_1, \cdots, \tilde{Q}^*_U)$ for the attribute set S^* is constructed as follows. If $sk^*_{S^*}$ is a valid key, then it should satisfy the following equation:

$$\frac{e(g,K^*)\cdot \prod_{x\in S^*} e(g,K_x)}{e(h,L)\cdot \prod_{x\in S^*} e(H_3(x),L)} = e(g,g)^{\alpha} = e(g,g)^{ab}$$

and so

$$\frac{e(g^c,K^*)\cdot \prod_{x\in S^*} e(g^c,K_x)}{e(h^c,L)\prod_{x\in S^*} e(H_3(x)^c,L)} = e(g,g)^{c\cdot\alpha} = e(g,g)^{abc}$$

and solves the BDH problem.

Analysis. In the simulation of the private key oracle, the responses to $\mathscr{A}$ are perfect. The simulations of the random oracles are perfect except for H_3. Let H_3^* be the event that $\mathscr{A}$ has queried $\delta_{2,x}$ to H_3 before the output phase (this event happens with probability q_{H_3}/p). Therefore, $Adv^{BDH}_{\mathscr{B}} \geq \varepsilon - \frac{q_{H_3}}{p}$.

6.2.6 Performance

In the above construction, we observe that the size of the parameters is linear in the value U. We recall that U is made public and assumed to be large enough to support DNA properties that our scheme requires. Indeed, the minisatellites are found in more than $1,000$ locations in the human genome. In the following table, we give details about the parameter sizes:

Parameters	Components in $\mathbb{G}_1$	Maximum size ($U = 1{,}000$)
$params$	$U+2$	$1{,}000$
sk_S	$3U+2$	$3{,}000$
$C_{(M,\rho)}$	$8U+l+3$	$9{,}000$

where l is the number of rows in the matrix M. We suppose that $l = O(U)$. We only count the number of elements in $\mathbb{G}_1$ since considering the other ones is not relevant. Indeed, we just find one element in $\mathbb{G}_T$ in *params* and one element in $\{0,1\}^{2\lambda}$ in $C_{(M,\rho)}$. For instance, for an elliptic curve of 220-bit length, our scheme is of almost $3{,}000{,}00$-bit length. Thus, the size of our construction is less than 400 kilobytes, meaning that its implementation seems realistic and practical.

In Table 6.2, we evaluate the efficiency of our CP-DBE symmetric pairing-based scheme. We use results of cryptographic operation implementations (group exponentiation of a random group element with a random exponent and pairing operations on random group elements) using the MIRACL framework [203, 196] for a 128-bit security level. All the following experiments are based on a dual core IntelR XeonR CPU W3503@2.40GHz with 2.0GB RAM running Ubuntu R10.04. The elliptic curve utilised for all the benchmarks was the super-singular symmetric curve $y^2 = x^3 + 1 \mod p$ with embedding degree 2 for suitable primes p.

We do not take into account multiplication/division in $\mathbb{G}_1$ and $\mathbb{G}_T$ as well as exponentiation in $\mathbb{G}_T$ as these operations have timings negligible compared to the ones for exponentiation in $\mathbb{G}_1$ and pairing operation.

	Group Exponentiation in $\mathbb{G}_1$	Pairing
Time/operation	34.4	176.6
Setup	$34{,}434{,}4$	176.6
KeyGen	$68{,}937.6$	
Encrypt	$240{,}937.6$	$706{,}576.6$
Decrypt	$378{,}434.4$	$1{,}237{,}612.8$

Table 6.2: Timings for our CP-DBE symmetric pairing-based system. Times are in milliseconds. We assume $U = 1000$.

We note that the total time in the algorithms Setup and KeyGen is substantial, but we recall that these algorithms should be run only once to generate the public parameters and the static private keys for all the users. Finally, in the algorithms Encrypt and Decrypt, it takes $947{,}514.2$ milliseconds and $1{,}616{,}047.2$ milliseconds respectively, mainly due to the cost of pairing computations and to the number l of rows in the matrix M that can be up to U. In particular, the time for the execution of the algorithm Decrypt derives from the verification process.

Since we gave a consequent value for U, it results that the protocol takes a significant time to be executed. For the performance section, we decided to give the approximate value of $U = 1000$ from the fact that the minisatellites are found in more than $1{,}000$ locations in the human genome. Nevertheless, since our scheme is proven secure for any value U, one can decide to choose a smaller integer and thus obtain a faster protocol execution.

6.2.7 Conclusion

We introduced a solution to encrypt documents using the DNA sequences of an individual and to decrypt them using the DNA sequences of another individual. Focusing on the principle of the DNA parentage test, we provided the primitive ciphertext-policy DNA-based encryption (CP-DBE). Based on Waters ABE scheme [237], we extended it by adding one extra feature to allow the decryption of the ciphertext, that the set of intersection operation. We proved that the CP-DBE construction is selectively IND-CCA secure and selectively collusion resistant in the random oracle model.

Chapter 7

Managing Electronic Health Records in Cloud Computing

Nowadays, Electronic health records (EHRs) are mostly stored on cloud servers. Given the EHR of a patient, the documents may be often changed by adding, deleting and/or modifying the contents. Thus, we must find a solution that allows each user accessing the EHR to easily and efficiently change the EHR. To make the process easier for both the patient and the cloud server, we propose to let the EHR be stored in plain instead of under their encrypted form. However, this brings critical security and privacy issues. Therefore, we have to ensure that the cloud server honestly acts by verifying the stored data integrity, as well as that the party who audits the cloud server learns nothing on the contents of the stored EHR.

A simple solution lets a clinician following a patient download each necessary time the entire contents that have to be added, deleted or modified, and then upload the updated contents on the server. Observe that such solution is cumbersome since we require the clinician to have enough resources on his/her side to be able to proceed (regarding the storage and computational resources for instance). The primitive that we suggest avoids the aforementioned situation by permitting the clinician to change some medical documents without downloading them. More precisely, the medical documents will be uploaded only once, and then each authorised user will add, delete and/or modify documents without retrieving them by actually requesting the cloud server to do so. As mentioned above, the cloud server will be audited periodically to ensure that it correctly stores and updates the stored EHR.

7.1 Dynamic Provable Data Possession with Public Verifiability and Data Privacy

The ability to check the integrity of the data is one of the most essential difficulty in storing data on an untrusted server. A data integrity verification ensures that the server will always keep the data intact. There exist various systems to prevent this issue, such as peer-to-peer storage systems, network file systems, long-term archives, web-service object stores and database systems. More precisley, these storage auditing systems enable a user (seen as a client from the server's point of view) to check that his/her stored data on an untrusted server are available and ready to collect, and so reassure the client.

The client, that is the data owner, requires to be sure that the server really possesses the claimed stored data. Numerous proof-of-storage solutions have been proposed, such as proofs of retrievability (POR) [127, 205] and provable data possessions (PDP) [12, 13]. In particular, a POR system allows a client to verify the integrity of his/her data stored on an untrusted server (using spot checking), to correct the possible errors (using error-correcting code) and to retrieve the entire document; while a PDP system enables the client to check that a server has stored his/her data without retrieving them from the server and without letting the server to access the entire data document. Both systems should satisfy the main property of *efficiency* in terms of computational and communication complexities and the storage overhead on the server's side should be as small as possible. The properties of unbounded uses on the number of proof-of-storage interactions and statelessness of the client are required to obtain systems with public verifiability, in which anyone can verify the integrity of the stored data [230, 112].

More recently, an idea emerged as delegating the data integrity check to a third party auditor (TPA) [225, 224]. More precisely, the client retains his/her data on an untrusted server and asks a TPA to verify the authenticity of the stored data. This concept is named *public verifiability*.

At the same time, another idea arised as *dynamically* updating the stored data [77, 232, 252]. In other words, the client is able to insert, delete and modify his/her stored data blocks and the server should then update these blocks on its side.

Observe that a storage service is susceptible to attacks or failures and leads to possible non-retrievable losses of the client's stored data. These attacks and failures are due to the fact that the storage system is vulnerable to internal and external attacks that harm the data integrity even being more powerful and reliable than the client's personal computing devices. A solution is to construct a system that offers an efficient, frequent and secure data integrity check process to the client. Nevertheless, the frequency of data integrity verification and the percentage of checked data are often limited because of the computational and communication costs on both server's and client's sides, although these two properties are really essential for storage service.

We consider the following scenario. A medical institute provides cloud servers to all the citizens, as a storage for their EHRs, as well as a TPA to check that the servers correctly stores these EHRs.

A patient, Bob, has his EHR stored on a cloud server. His EHR were first uploaded by a medical institute (for instance, belonging to the government). Bob is able to manage his EHR by adding, removing and deleting medical documents. For instance, Bob can add some medical test results when they are performed overseas and no automatic medical document transfer exists between Bob's country and the visited country.

Moreover, the practionners following Bob can also manage his medical documents. For instance, we consider that case that Bob comes to the medical office for a checkup, and meets his GP Greg. The latter has the ability to read and write Bob's records. In more details, we let Greg be able to create a new chekup report and add i to Bob's stored EHR, to delete or to modify an existing stored record.

7.1.1 Contributions

In this section, we provide a dynamic provable data possession (DPDP) system with public verifiability and data privacy. There are three entities in the system: a client who is the owner of the data to be stored, an untrusted server that stores the data (e.g. a cloud), and

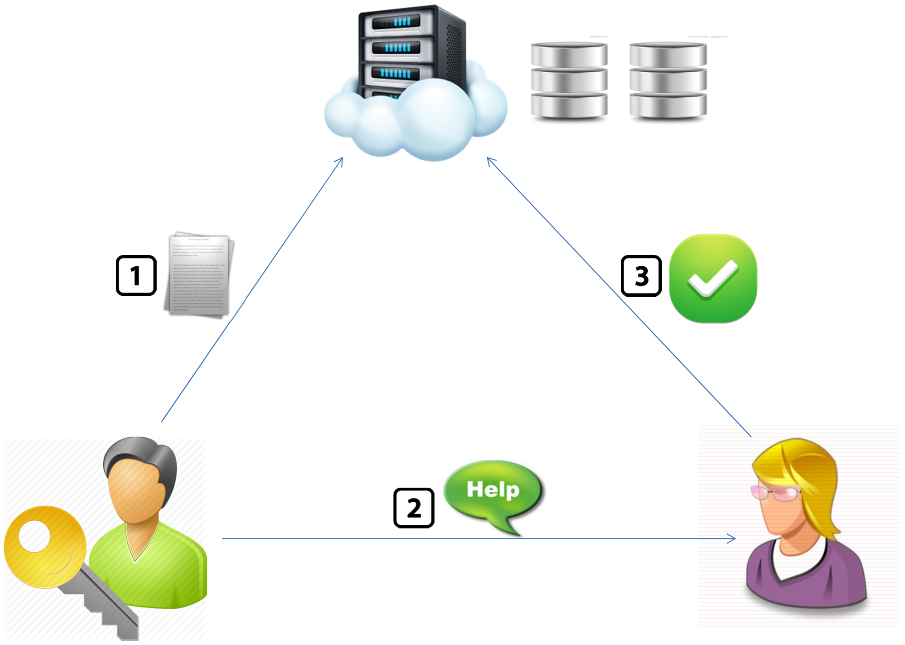


Figure 7.1: Dynamic Provable Data Possession with Public Verifiability and Data Privacy. A patient, Bob, has uploaded his EHR in plain on a cloud server (this task can be done by a medical institute chosen by the government and applied to all the citizens). A TPA is selected by the medical institute to regularly check that the server correctly stores Bob's medical data at all time.

a semi-honest TPA who can be required when the client wants to check the integrity of his/her data stored on the server.

The client gets a large amount of data that s/he wants to store on the server without retaining a local copy. The server gets an important storage space and computation resources, and supplies services for the client. The system is public verifiable, meaning that anyone is enabled to check the integrity of the data, not only the TPA on behalf of the client or the client itself. Indeed, we stress that the client may be able to perform integrity checking by itself. However, the TPA can be requested to judge whether the data integrity is maintained by checking the proof of data possession provided by the server. For instance, in case of conflict between the client and the server or of the client's limited resources. Since this often happens in practice, we only consider the case of the TPA acting on behalf of the client.

The system is also data dynamic at the data block level supporting three operations: insertion, deletion and modification. Finally, the system is secure at the untrusted server, meaning that a server cannot successfully generate a correct proof of data possession without storing all the data blocks, and data private, meaning that the TPA learns nothing about the data of the client from all available information.

The main contribution is the practicality of our scheme. The first refinement is a better efficiency due the use of asymmetric pairings. The second amelioration is a decrease of

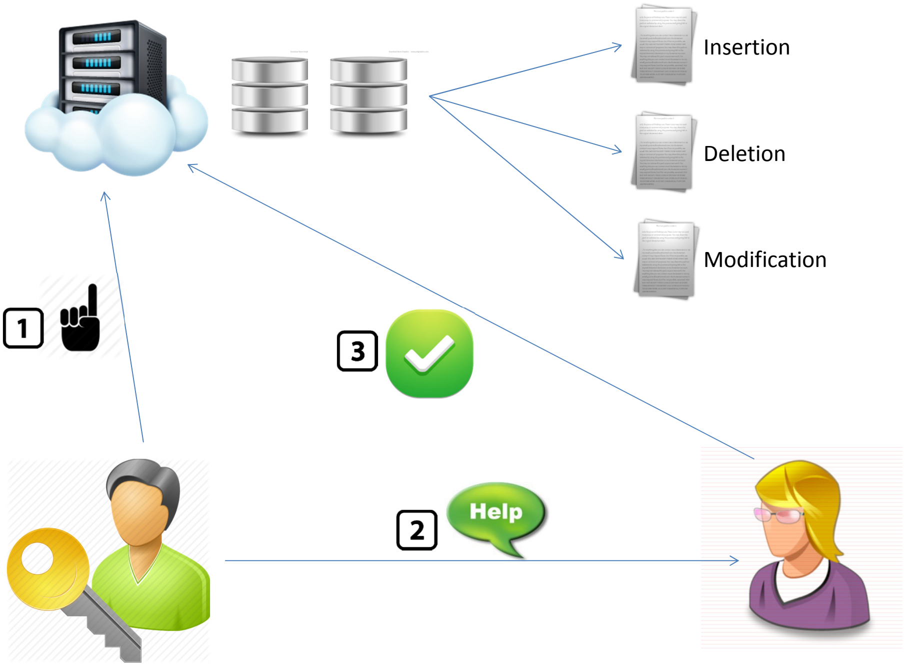


Figure 7.2: Dynamic Provable Data Possession with Public Verifiability and Data Privacy. The patient Bob has the possibility to insert new documents, and delete and modify existing documents. To do so, he sends a request to the cloud server with the necessary information to let the latter proceed. Bob asks for a help to the TPA to check that the server has correctly updated his data.

the number of group exponentiation and pairing computations. In particular, the TPA needs to compute no exponentiation and only three pairings in order to verify the proof of data possession generated by the server. This implies that the latter can be requested by the client through the TPA to create the proof on any percentage of the stored data, without any computational constraints. The result of these improvements is clear in terms of performance evaluation.

7.1.2 Protocol Definition

We recall the definition of the primitive given in Section 3.1.7.
Let m be a data document to be stored. First, m is divided into n blocks m_i and then each block m_i is divided into s sectors $m_{i,j}$. We assume that all the blocks and sectors are elements in $\mathbb{Z}_p$, where p is a large prime. For instance, p should be λ-bit long, where λ is the security parameter such that $n >> \lambda$.

To illustrate, we refer to an example given in [205]. We suppose that m is a b-bit document. Thus, m is split into n blocks such that $n = \lceil b/s \cdot \log(p) \rceil$. Morever, each block is composed of s sectors, where $1 \leq s$. The tradeoff between the storage overhead and the communtication overhead is a follows: the communication complexity increases by $s+1$ elements of $\mathbb{Z}_p$. Thus, a larger value of s yields less storage overhead at cost of a high communication.

A dynamic provable data possession (DPDP) scheme with public verifiability and data privacy is composed of the following six algorithms:

- $\mathsf{KeyGen}(\lambda) \to (pk, sk)$. This algorithm is run by the client to setup the scheme. On input the security parameter λ, output a pair of public and private keys (pk, sk).
- $\mathsf{TagGen}(pk, sk, m_i) \to T_{m_i}$. This algorithm is run by the client to generate the verification metadata. On inputs the public key pk, the private key sk and a data block m_i (for $i \in [1,n]$ before data operations, and for $i \in (0, n+1) \cap \mathbb{Q}$ after data operations, where m_i are the data blocks that form the document m), output a verification metadata T_{m_i}. Then, the client sets all the data blocks m_i in an ordered collection $\mathbb{F}$ and all the corresponding verification metadata T_{m_i} in an ordered collection $\mathbb{E}$. S/he sends the collections $\mathbb{F}$ and $\mathbb{E}$ to the server and removes them from his/her local storage.
- $\mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info = (\text{operation}, l, m_l, T_{m_l})) \to (\mathbb{F}', \mathbb{E}', \nu')$. This algorithm is run by the server in response to a data operation requested by the client. On inputs the public key pk, the previous collection $\mathbb{F}$ of all the data blocks, the previous collection $\mathbb{E}$ of all the corresponding verification metadata, the type of the data operation to be performed (insertion, deletion or modification at block level), the index l denoting the rank where the data operation is performed (in the ordered collections $\mathbb{F}$ and $\mathbb{E}$ such that $l = \frac{i_1+i_2}{2}$ for insertion and $l = i$ for deletion and modification), the data block m_l to be inserted, deleted or modified ($m_l = m_{\frac{i_1+i_2}{2}}$ for insertion, m_l is not required for deletion and $m_l = m'_i$ for modification), and the corresponding verification metadata T_l to be inserted, deleted or modified ($T_{m_l} = T_{m_{\frac{i_1+i_2}{2}}}$ for insertion, T_{m_l} is not required for deletion and $T_{m_l} = T_{m'_i}$ for modification), for $l \in (0, n+1) \cap \mathbb{Q}$. We give more precisions for several operation options:
  - *insertion:* $m_{\frac{i_1+i_2}{2}}$ is inserted between the two consecutive blocks m_{i_1} and m_{i_2} and $T_{m_{\frac{i_1+i_2}{2}}}$ is inserted between the two consecutive verification metadata $T_{m_{i_1}}$ and $T_{m_{i_2}}$. For $i_1 = 0$ (i.e. we append a block at the beginning of the document), $m_{\frac{i_2}{2}}$ is appended before m_{i_2} and $T_{m_{\frac{i_2}{2}}}$ is appended before $T_{m_{i_2}}$. For $i_2 = n+1$ (i.e. we append a block at the end of the document), $m_{\frac{i_1+n+1}{2}}$ is appended after m_{i_1} and $T_{m_{\frac{i_1+n+1}{2}}}$ is appended after $T_{m_{i_1}}$.

    We assume that the data block $m_{\frac{i_1+i_2}{2}}$ and the corresponding verification metadata $T_{m_{\frac{i_1+i_2}{2}}}$ were provided by the client to the server, such that $T_{m_{\frac{i_1+i_2}{2}}}$ was correctly computed by running the algorithm TagGen.
  - *deletion:* m_i is deleted, meaning that m_{i_1} is followed by m_{i_2} and T_{m_i} is deleted, meaning that $T_{m_{i_1}}$ is followed by $T_{m_{i_2}}$, for three consecutive blocks m_{i_1}, m_i and m_{i_2}. For $i_1 = 0$ (i.e. we remove a block at the beginning of the document), m_i is removed, and so the document now begins with m_{i_2}, and T_{m_i} is removed, and so the collection of verification metadata now begins with $T_{m_{i_2}}$. For $i-2 = n+1$ (i.e. we remove a block at the end of the document), m_i is removed, and so the document now ends with m_{i_1}, and T_{m_i} is removed, and so the collection of verification metadata now ends with $T_{m_{i_1}}$.

  - *modification:* m'_i replaces m_i and $T_{m'_i}$ replaces T_{m_i}. We assume that the data block m'_i and the corresponding verification metadata $T_{m'_i}$ were provided by the client to the server, such that $T_{m'_i}$ was correctly computed by running the algorithm TagGen.

  Note that the set of block indices is $[1,n]$ when the document is uploaded for the first time, and the is included in $(0,n+1) \cap \mathbb{Q}$ after insertions.

  Finally, output the updated data block collection $\mathbb{F}'$ containing m_l for insertion and modification and not containing it for deletion, the updated verification metadata collection $\mathbb{E}'$ containing T_{m_l} for insertion and modification and not containing it for deletion and an updating proof ν'.

- $\mathsf{CheckOp}(pk, \nu') \rightarrow true/false$. This algorithm is run by the TPA on behalf of the client to validate the updating proof. On inputs the public key *pk* and the updating proof ν' sent by the server, output *true* if ν' is a correct updating proof; output *false* otherwise. We suppose that the resulting answer may be then sent to the client.

- $\mathsf{GenProof}(pk, F, chal, \Sigma) \rightarrow \nu$. This algorithm is run by the server in order to generate a proof of data possession. On input the public key *pk*, an ordered collection $F \subset \mathbb{F}$ of blocks, a challenge *chal* and an ordered collection $\Sigma \subset \mathbb{E}$ which are the verification metadata corresponding to the blocks in F, output a proof of data possession ν for the data blocks in F that are determined by the challenge *chal*.

  We assume that a first challenge $chal_C$ may be generated by the client and forwarded to the TPA. Then, the TPA will generate a challenge *chal* from $chal_C$ and will send it to the server. In particular, if the client wants to check the integrity of his/her data without the help of the TPA, then $chal_C = chal$. We omit the process done by the client at this point.

- $\mathsf{CheckProof}(pk, chal, \nu) \rightarrow true/false$. This algorithm is run by the TPA in order to validate the proof of data possession. On inputs the public key *pk*, the challenge *chal* and the proof of data possession ν, output *true* if ν is a correct proof of data possession for the blocks determined by *chal*; output *false* otherwise. We suppose that the resulting answer may be then sent to the client.

Correctness. We require that a dynamic provable data possession scheme with public verifiability and data privacy is correct if for any $\lambda \in \mathbb{N}$, $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$, $T_{m_i} \leftarrow \mathsf{TagGen}(pk, sk, m_i)$ (for $i \in [1,n]$ before data operations, and for $i \in (0,n+1) \cap \mathbb{Q}$ after data operations, where m_i are the data blocks that form the document m), and

- if $(\mathbb{F}', \mathbb{E}', \nu') \leftarrow \mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info)$, we have that $\mathsf{CheckOp}(pk, sk, \nu') = true$;
- if $\nu \leftarrow \mathsf{GenProof}(pk, F, chal, \Sigma)$, we have that $\mathsf{CheckProof}(pk, sk, chal, \nu) = true$.

Remarks.

- *PDP versus POR:* When examining data storage practicality concerns, POR systems can be seen as the "right" choice. Indeed, a POR system supplies successful audit guarantees that all the data can be extracted from the server. However, it requires erasure codes that are a cost for both storage and communication.

One solution is to consider PDP systems that can be seen as "weaker" than the aforementioned one. Indeed, we only want to guarantee that a certain percentage (e.g. 90%) of data blocks are available, not all of them. Using this model, the overhead from erasure code applications is avoided.

Suppose that 10% of a client's data has just been lost. The data integrity verification can still pass as 90% of the data are still available. But the probability that the available 90% are chosen to run the audit process is really low. Thus, a client has more chances to learn about the 10% loss and that the data integrity verification fails. Moreover, to ensure a positive result, the audit process can be performed an unlimited number of times.

- *About the proof of data possession:* The set of stored data blocks (following a certain percentage of blocks, e.g. 90%) that are checked are chosen by the TPA on behalf of the client. The server has to generate a proof of data possession based on this set. We notice that sometimes in the literature [112, 245], the TPA just sends a challenge *chal* without specifying which blocks have to be checked, which leads to the fact that the server must generate a proof of data possession based on all the stored data blocks, at the cost of the communication overhead.

- *About the data operations:* We assume that the frequency of checking the integrity of the stored data is much higher than the frequency of performing data operations. To generate an updating proof, no challenge is required, meaning that the updating proof is only based on the recently updated data block and the corresponding verification metadata. Therefore, one can think that this proof is not strong enough; however we suppose that the TPA on behalf of the client regularly asks to the server to check the integrity of the data by generating a challenge that can include the recently updated data blocks.

  Moreover, when the server is generating the updating proof ν', it can include an element $info' \in \{\text{insertion}, \text{deletion}, \text{modification}\}$ in this proof to enable the TPA to know which operation was performed. A solution to check that the server has correctly updated the collection $\mathbb{F}'$ of the data blocks and the collection $\mathbb{E}'$ of the verification metadata after a given operation is to order the data into a Merkle hash tree (MHT) [230] or rank-based authenticated skip lists [77].

  Similarly to [77], we have to include the entity TPA in the process to let it act on behalf of the client. Therefore, in the algorithm $\mathsf{CheckOp}$, the private key *sk* cannot be taken as input in order to allow the TPA proceed. In addition, since we assume that the client does not store neither the data blocks nor the corresponding verification metadata on his/her local storage and so does not the TPA, the ordered collections $\mathbb{F}$ and $\mathbb{E}$ cannot be outputs of the algorithm PerfOp. Laslty, the algorithm TagGen can be seen as a preparation of the data operation to be performed, meaning that extra output could be the element $info$ to be sent to the server.

7.1.3 Security Models

Model for the Security against the Server

The definition of the security game given below follows the ones found in [12] and [77]. We consider a DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen},$

$\mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$. Let a data possession game between a challenger $\mathscr{B}$ and an adversary $\mathscr{A}$ be as follows.

Setup. $\mathscr{B}$ runs the algorithm KeyGen on input the security parameter λ to obtain (pk, sk). The element pk is given to $\mathscr{A}$ while the element sk is kept secret.

Adaptive Queries. $\mathscr{A}$ makes adaptive queries by calling two oracles, namely a verification metadata generation oracle $\mathscr{O}_{TG}$ and a data operation performance oracle $\mathscr{O}_{DOP}$.

First, $\mathscr{A}$ is given access to a verification metadata generation oracle $\mathscr{O}_{TG}$ as follows. $\mathscr{A}$ chooses several blocks m_i and gives them to $\mathscr{B}$, for $i \in [1, n]$. $\mathscr{B}$ generates the corresponding verification metadata $T_{m_i} \leftarrow \mathsf{TagGen}(pk, sk, m_i)$ and forwards them to $\mathscr{A}$. Then, $\mathscr{A}$ creates an ordered collection $\mathbb{F} = \{m_1, \cdots, m_n\}$ of data blocks along with an ordered collection $\mathbb{E} = \{T_{m_1}, \cdots, T_{m_n}\}$ of the corresponding verification metadata.

Thereafter, $\mathscr{A}$ is given access to a data operation performance oracle $\mathscr{O}_{DOP}$ as follows. $\mathscr{A}$ gives to $\mathscr{B}$ a block m_i, for $i \in [1, n]$, and the corresponding value $info_i$ about the data operation that $\mathscr{A}$ wants to perform. $\mathscr{A}$ also submits a new ordered collection $\mathbb{F}'$ of data blocks, a new ordered collection $\mathbb{E}'$ of verification metadata, and the corresponding updating proof ν'. $\mathscr{B}$ verifies the value ν' by running the algorithm $\mathsf{CheckOp}(pk, \nu')$ and replies by giving back to $\mathscr{A}$ the resulting answer that is either $true$ or $false$. If the answer is $false$, then $\mathscr{B}$ aborts; otherwise, it proceeds. The above interaction between $\mathscr{A}$ and $\mathscr{B}$ can be repeated.

Challenge. $\mathscr{A}$ chooses some data blocks m_i^* along with the corresponding values $info_i^*$, for $i \in \mathscr{I} \subseteq (0, n+1) \cap \mathbb{Q}$. Adaptive queries can be again made by $\mathscr{A}$, such that the first value $info_i^*$ specifies a full re-write update (this corresponds to the first time that the client sends a document to the server). $\mathscr{B}$ still checks the data operations.

The final version of the blocks m_i^* for $i \in \mathscr{I}$ is considered such that these blocks were created according to the data operations requested by $\mathscr{A}$, and verified and accepted by $\mathscr{B}$ beforehand. $\mathscr{B}$ sets $\mathbb{F} = \{m_i^*\}_{i \in \mathscr{I}}$ of these data blocks and $\mathbb{E} = \{T_{m_i^*}\}_{i \in \mathscr{I}}$ of the corresponding verification metadata. It then chooses an integer $k \leq |\mathscr{I}|$ and sets an ordered collection $F = \{m_{i_1^*}, \cdots, m_{i_k^*}\} \subset \mathbb{F}$ and the ordered collection $\Sigma = \{T_{m_{i_1^*}}, \cdots, T_{m_{i_k^*}}\} \subset \mathbb{E}$ of the corresponding verification metadata, for $i_j \in \mathscr{I}$ and $j \in [1, k]$. It computes a resulting challenge $chal$ for F and Σ and sends it to $\mathscr{A}$.

Forgery. $\mathscr{A}$ computes a proof of data possession ν on $chal$. Then, $\mathscr{B}$ runs $\mathsf{CheckProof}(pk, chal, \nu)$ and replies by giving back the answer to $\mathscr{A}$ that is either $true$ or $false$. If the answer is $true$ then $\mathscr{A}$ wins.

The DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$ is *secure* against the server if for all probabilistic polynomial-time (PPT) adversary $\mathscr{A}$ that wins the above game, there exists another polynomial-time algorithm $\mathscr{E}$, denoted as a knowledge extractor, that is capable of extracting the data blocks. The rationale of the model is that for any adversary that can win the data possession game, it can employ $\mathscr{E}$ to compute the data blocks in polynomial time. In other words, any algorithm that can answer the challenge must be in possession of the underlying data blocks stored in one form or another.

Note that the above security model follows the definition of the extractor in proofs of knowledge. The goal is to extract at least the challenged parts of a document m from the adversary's approving responses.

Model for the Data Privacy against the TPA

We present two security models to prove data privacy against the TPA in our protocol. The first model is the weakest one; however, this is still clever enough to ensure security. We follow the weak security model to prove that the basic DPDP construction DPDP and the index hash table (IHT)-based DPDP construction are data private.

The second model is the strongest one, based on indistinguishability. We use this strong security model to prove that the Merkle hash tree (MHT)-based DPDP construction is data private.

Note that the IHT-based and the MHT-based DPDP systems are presented in Sections 7.3 and 7.4. Both systems follow the DPDP protocol definition as well as the security models here.

Weak Security Model. Despite the fact that this model is not based on the indistinguishability property, it closely follows the adversarial reality. Indeed, we can wish that even if the TPA is able to distinguish two documents, it still does not learn anything about the contents of these documents. Moreover, it may have to check the same blocks several times during different challenge-response audits. For instance, even if the TPA notices that it has verified the same block during two consecutive challenge-response audits, it only knows that this block appeared twice, however it does not know more information about it. We recall that the task of the TPA is to check that the server correctly performs the data operations at block level and stores the whole data. Therefore, for a given operation, the TPA is aware of which block is considered and so, it may be able to differentiate it with another one given another operation. Yet again, it does not have access to more details about contents of these two blocks.

In addition, note that such security model is found in the literature [230, 227] to show that public auditing systems preserve data privacy. The requirement is to introduce a TPA that checks that the server correclty stores the data of a user, without bringing new vulnerabilities toward the data privacy. To prove that the schemes are data private, the authors in [230, 227] revealed the existence of a challenger that can produce a valid proof of data possession without the knowledge of the challenged blocks, by using a backpatching technique in the random oracle model as in [205].

Thus, we argue that a security model based on one-wayness is sufficient for most of the cases In the case of one-wayness, an adversary can not obtain the whole data contained in a given verification metadata by assuming that the public parameters are at its disposal.

We consider a DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$. Let a weak data privacy game between a challenger $\mathscr{B}$ and an adversary $\mathscr{A}$ be as follows.

Setup. The challenger runs the algorithm KeyGen to generate the pair of public and private keys (pk, sk) and gives pk to the adversary. The element sk is kept secret.

Queries. $\mathscr{A}$ is allowed to make verification metadata generation queries as follows. The adversary sends a file $m = (m_1, \cdots, m_n)$ to $\mathscr{B}$. The latter computes the corresponding verification metadata $T_m = (T_{m_1}, \cdots, T_{m_n})$ and gives it back to $\mathscr{A}$. Then, two ordered collectections $\mathbb{F} = \{m_1, \cdots, m_n\}$ of file blocks and $\mathbb{E} = \{T_{m_1}, \cdots, T_{m_n}\}$ of the corresponding verification metadata are created.

Challenge. $\mathscr{A}$ sumits a challenge *chal* containing $k \leq n$ indices, along with the k corresponding file blocks in F and their k verification metadata in Σ.

Generation of the Proof. The challenger computes a proof of data possession $\nu^* \leftarrow \mathsf{GenProof}(pk, F, chal, \Sigma)$ such that the blocks in F are determined by the challenge *chal* and Σ contains the corresponding verification metadata.

$\mathscr{A}$ succeeds in the weak data privacy game if $F \nsubseteq \mathbb{F}$ and $\Sigma \nsubseteq \mathbb{E}$, and $\mathsf{CheckProof}(pk, chal, \nu^*) \rightarrow true$. The advantage of the adversary $\mathscr{A}$ in winning the weak data privacy game is defined as $Adv^{WP}_{DPDP,\mathscr{A}}(\lambda) = Pr[\mathscr{A} \text{ succeeds}]$.

The DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$ is *weakly data private* if there is no PPT adversary $\mathscr{A}$ who can win the above weak data privacy game with non-negligible advantage $Adv^{WP}_{DPDP,\mathscr{A}}(\lambda)$. Informally, this means that there is no adversary $\mathscr{A}$ who can recover the document from a given verification metadata tuple with non-negligible probability.

Strong Security Model. The definition of the security game described below follows the one proposed by Fan et al. [80]. They defined data privacy using an indistinguishability game between a challenger $\mathscr{B}$ (palying the role of the server) and an adversary $\mathscr{A}$ (playing the role of the TPA). A similar definition for data privacy has been presented in [245] as an enhancement of the model given in [112].

We consider a DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$. Let a strong data privacy game between a challenger $\mathscr{B}$ and an adversary $\mathscr{A}$ be as follows.

Setup. $\mathscr{B}$ runs the algorithm KeyGen on input the security parameter λ to obtain (pk, sk). The element pk is given to $\mathscr{A}$ while the element sk is kept secret.

Queries. $\mathscr{A}$ is allowed to make verification metadata generation queries as follows. The adversary sends a file $m = (m_1, \cdots, m_n)$ to $\mathscr{B}$. The latter computes the corresponding verification metadata $T_m = (T_{m_1}, \cdots, T_{m_n})$ and gives it back to $\mathscr{A}$. Then, two ordered collectections $\mathbb{F} = \{m_1, \cdots, m_n\}$ of file blocks and $\mathbb{E} = \{T_{m_1}, \cdots, T_{m_n}\}$ of the corresponding verification metadata are created.

Challenge. $\mathscr{A}$ submits two different documents m_0 and m_1 of equal length, such that they have not be chosen in the query phase, and sends them to $\mathscr{B}$. The latter generates T_{m_0} and T_{m_1} by running the algorithm TagGen. Then, $\mathscr{B}$ randomly chooses a bit $\mu_R \in \{0, 1\}$ and forwards T_{m_μ} to $\mathscr{A}$. Then, $\mathscr{A}$ sets a challenge *chal* and sends it to $\mathscr{B}$. The latter generates a proof of data possession ν^* based on m_μ, T_{m_μ} and *chal*, and replies to $\mathscr{A}$ by giving ν^*.

Guess. $\mathscr{A}$ chooses a bit $\mu' \in \{0, 1\}$ and wins the game if $\mu' = \mu$.

The advantage of the adversary $\mathscr{A}$ in winning the above strong data privacy game is defined as $Adv^{SP}_{DPDP,\mathscr{A}}(\lambda) = |Pr[\mu' = \mu] - \frac{1}{2}|$. A proof of data possession ν^* has indistinguishability if for any PPT adversary $\mathscr{A}$, $Adv^{SP}_{DPDP,\mathscr{A}}(\lambda)$ is a negligible function in the security parameter λ.

The DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$ is *strongly data private* if there is no PPT adversary $\mathscr{A}$ who can win the above data privacy game with non-negligible advantage $Adv^{SP}_{DPDP,\mathscr{A}}(\lambda)$.

7.1.4 Construction

We now give the constrution of the basic DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$.

The document to be stored is split into n blocks, and each block is split into s sectors. We let each block and sector be elements of $\mathbb{Z}_p$ for some large prime p. For instance, let the document be b bits long. Then, the document is split into $n = \lceil b/s \cdot \log(p) \rceil$ blocks. The aforementioned intuition comes from [205]. Therefore, a tradeoff exists between the storage overhead and the communtication overhead. More precisely, the communication complexity rises as $s+1$ elements of $\mathbb{Z}_p$. Finally, a larger value of s yields less storage overhead at cost of a high communication. Observe that the size of the verification metadata should be much smaller than the size of the data documents: in our case, $|T_m| = |m| = O(p)$. Moreover, p should be λ bits long, where λ is the security parameter such that $n >> \lambda$. The curve to be chosen should be such that the discrete log is 2^λ-secure.

- $\mathsf{KeyGen}(\lambda) \to (pk, sk)$. Let $\mathscr{G}(\lambda)$ be an algorithm that, on input the security parameter λ, generates the cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let g_1 and g_2 be two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Then, the client randomly chooses s elements $h_1, \cdots, h_s \in_R \mathbb{G}_1$. Moreover, s/he selects at random $\alpha \in_R \mathbb{Z}_p$ and sets his/her public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^\alpha)$ and his/her private key $sk = \alpha$.

- $\mathsf{TagGen}(pk, sk, m) \to T_m$. A document m is split into n blocks m_i, for $i \in [1, n]$. Each block m_i is then split into s sectors $m_{i,j} \in \mathbb{Z}_p$, for $j \in [1, s]$. We suppose that $|m| = b$ and $n = \lceil b/s \cdot \log(p) \rceil$. Therefore, the document m can be seen a $n \times s$ matrix with elements denoted as $m_{i,j}$. The client computes the verification metadata $T_{m_i} = (\prod_{j=1}^s h_j^{m_{i,j}})^{-sk} = (\prod_{j=1}^s h_j^{m_{i,j}})^{-\alpha} = (\prod_{j=1}^s h_j^{-\alpha \cdot m_{i,j}})$ for $i \in [1, n]$. Then, it sets $T_m = (T_{m_1}, \cdots, T_{m_n}) \in \mathbb{G}_1^n$.

  Moreover, the client stores all the data blocks m_i in an ordered collection $\mathbb{F}$ and the corresponding verification metadata T_{m_i} in an ordered collection $\mathbb{E}$. It forwards these two collections to the server and deletes them from his/her local storage.

- $\mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info = (\text{insertion}, \frac{i_1+i_2}{2}, m_{\frac{i_1+i_2}{2}}, T_{m_{\frac{i_1+i_2}{2}}})) \to (\mathbb{F}', \mathbb{E}', \nu')$. After receiving the elements $\frac{i_1+i_2}{2}$, $m_{\frac{i_1+i_2}{2}}$ and $T_{m_{\frac{i_1+i_2}{2}}}$ from the client, such that m_{i_1} and m_{i_2} are two consecutive data blocks, the server prepares the updating proof as follows. It first selects at random $u_1, \cdots, u_s \in_R \mathbb{Z}_p$ and computes $U_1 = h_1^{u_1}, \cdots, U_s = h_s^{u_s}$. It also chooses at random $w_{\frac{i_1+i_2}{2}} \in_R \mathbb{Z}_p$ and sets $c_j = m_{\frac{i_1+i_2}{2},j} \cdot w_{\frac{i_1+i_2}{2}} + u_j \in \mathbb{Z}_p$ and $C_j = h_j^{c_j}$, for $j \in [1,s]$, and $d = T_{m_{\frac{i_1+i_2}{2}}}^{w_{\frac{i_1+i_2}{2}}}$. Finally, it returns $\nu' = (U_1, \cdots, U_s, C_1, \cdots, C_s, d) \in \mathbb{G}_1^{2s+1}$ to the TPA.

To append data blocks at the beginning or at the end of the document, the client sends the tuple $(\text{insertion}, i, m_i, T_{m_i})$ to the server, for $i \in (\mathbb{Q} \cap (0,1]) \cup (\mathbb{Q} \cap [n, n+1))$. The latter then selects at random $u_1, \cdots, u_s \in_R \mathbb{Z}_p$ and computes $U_1 = h_1^{u_1}, \cdots$, $U_s = h_s^{u_s}$. It also chooses at random $w_i \in_R \mathbb{Z}_p$ and sets $c_j = m_{i,j} \cdot w_i + u_j$ and $C_j = h_j^{c_j}$, for $j \in [1,s]$, and $d = T_{m_i}^{w_i}$. Finally, it returns $\nu' = (U_1, \cdots, U_s, C_1, \cdots, C_s, d)$ to the TPA.

- $\mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info = (\text{deletion}, i)) \rightarrow (\mathbb{F}', \mathbb{E}', \nu')$. After receiving an index i from the client, the server prepares the updating proof as follows. It first selects at random $u_1, \cdots, u_s \in_R \mathbb{Z}_p$ and computes $U_1 = h_1^{u_1}, \cdots, U_s = h_s^{u_s}$. It also chooses at random $w_i \in_R \mathbb{Z}_p$ and sets $c_j = m_{i,j} \cdot w_i + u_j \in \mathbb{Z}_p$ and $C_j = h_j^{c_j}$, for $j \in [1,s]$, and $d = T_{m_i}^{w_i}$, where m_i and T_{m_i} are the existing data block and verification metadata to be deleted respectively. Finally, it returns $\nu' = (U_1, \cdots, U_s, C_1, \cdots, C_s, d) \in \mathbb{G}_1^{2s+1}$ to the TPA.

- $\mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info = (\text{modification}, i, m'_i, T_{m'_i})) \rightarrow (\mathbb{F}', \mathbb{E}', \nu')$. After receiving the elements i, m'_i and $T_{m'_i}$ from the client, the server prepares the updating proof as follows. It first selects at random $u_1, \cdots, u_s \in_R \mathbb{Z}_p$ and computes $U_1 = h_1^{u_1}, \cdots, U_s = h_s^{u_s}$. It also chooses at random $w_i \in_R \mathbb{Z}_p$ and sets $c_j = m'_{i,j} \cdot w_i + u_j \in \mathbb{Z}_p$ and $C_j = h_j^{c_j}$, for $j \in [1,s]$, and $d = T_{m'_i}^{w_i}$. Finally, it returns $\nu' = (U_1, \cdots, U_s, C_1, \cdots, C_s, d) \in \mathbb{G}_1^{2s+1}$ to the TPA.

- $\mathsf{CheckOp}(pk, \nu') \rightarrow true/false$. The TPA has to check whether the following equation holds:

$$e(d, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} U_j, g_2) \stackrel{?}{=} e(\prod_{j=1}^{s} C_j, g_2) \tag{7.1}$$

  If the equation 7.1 holds, then the TPA returns $true$ to the client; otherwise, it returns $false$ to the client.

- $\mathsf{GenProof}(pk, F, chal, \Sigma) \rightarrow \nu$. After possibly receiving a challenge $chal_C$ from the client, the TPA prepares a challenge $chal$ to send to the server as follows. First, it chooses a subset $I \subseteq (0, n+1) \cap \mathbb{Q}$, randomly chooses $|I|$ elements $v_i \in_R \mathbb{Z}_p$ and sets $chal = \{(i, v_i)\}_{i \in I}$. Second, after receiving the challenge $chal$ which indicates the specific blocks for which the client, through the TPA, wants a proof of data possession, the server sets the ordered collection $F = \{m_i\}_{i \in I} \subset \mathbb{F}$ of blocks and an ordered collection $\Sigma = \{T_{m_i}\}_{i \in I} \subset \mathbb{E}$ which are the verification metadata corresponding to the blocks in F. It then selects at random $r_1, \cdots, r_s \in_R \mathbb{Z}_p$ and computes $R_1 = h_1^{r_1}, \cdots, R_s = h_s^{r_s}$. It also sets $b_j = \sum_{(i,v_i) \in chal} m_{i,j} \cdot v_i + r_j \in \mathbb{Z}_p$ and $B_j = h_j^{b_j}$, for $j \in [1,s]$, and $c = \prod_{(i,v_i) \in chal} T_{m_i}^{v_i}$. Finally, it returns $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c) \in \mathbb{G}_1^{2s+1}$ to the TPA.

- $\mathsf{CheckProof}(pk, chal, \nu) \rightarrow true/false$. The TPA has to check whether the following equation holds:

$$e(c, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} R_j, g_2) \stackrel{?}{=} e(\prod_{j=1}^{s} B_j, g_2) \tag{7.2}$$

  If the equation 7.2 holds, then the TPA returns $true$ to the client; otherwise, it returns $false$ to the client.

Correctness. If all the algorithms are correctly generated, then the above scheme is correct. For the updating proof, we have:

$$\begin{aligned}
e(d, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} U_j, g_2) &= e(T_{m_i}^{w_i}, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{m_{i,j}\cdot(-\alpha)\cdot w_i}, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{m_{i,j}\cdot w_i}, g_2)^{\alpha-\alpha} \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{m_{i,j}\cdot w_i + u_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{c_j}, g_2) = e(\prod_{j=1}^{s} C_j, g_2)
\end{aligned}$$

For the proof of data possession, we have:

$$\begin{aligned}
e(c, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} R_j, g_2) &= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} T_{m_i}^{v_i}, g_2^{\alpha}\right) \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} \prod_{j=1}^{s} h_j^{m_{i,j}\cdot(-\alpha)\cdot v_i}, g_2^{\alpha}\right) \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e\left(\prod_{j=1}^{s} h_j^{\sum_{\substack{(i,v_i)\\ \in chal}} m_{i,j}\cdot v_i}, g_2\right)^{\alpha-\alpha} \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e\left(\prod_{j=1}^{s} h_j^{\sum_{\substack{(i,v_i)\\ \in chal}} m_{i,j}\cdot v_i + r_j}, g_2\right) \\
&= e(\prod_{j=1}^{s} h_j^{b_j}, g_2) = e(\prod_{j=1}^{s} B_j, g_2)
\end{aligned}$$

7.1.5 Security Proofs

Proof for the Security against the Server

Theorem. Let $\mathscr{A}$ be a PPT adversary that has advantage ε against the security of the basic DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen},$

$\mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$. Then, there is a challenger $\mathscr{B}$ that solves the DHI and DL problems in $\mathbb{G}$ and $\mathbb{G}_1$ respectively with advantage $\varepsilon' = O(\varepsilon)$.

For any PPT adversary $\mathscr{A}$ who wins the game, there is a challenger $\mathscr{B}$ that interacts with the adversary $\mathscr{A}$ to break the DHI and DL problems in $\mathbb{G}$ and $\mathbb{G}_1$ as follows.

Setup. $\mathscr{B}$ runs $\mathsf{GroupGen}(\lambda) \rightarrow (p, \mathbb{G}, \mathbb{G}_T, e)$. Then, it is given the DHI instance tuple (g, g^a) such that g is a generator of $\mathbb{G}$ and $a \in \mathbb{Z}_p^*$, chooses two exponents $x, y \in \mathbb{Z}_p$ and computes $g_1 = g^x$ and $g_2 = g^y$. It also sets $\mathbb{G}_1 =< g_1 >$ and $\mathbb{G}_2 =< g_2 >$. Note that $(g^a)^x = g_1^a$ and $(g^a)^y = g_2^a$. $\mathscr{B}$ then chooses $\beta_j \in_R \mathbb{Z}_p$ and sets $h_j = g_1^{\beta_j}$ for $j \in [1, s]$. $\mathscr{B}$ sets the public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^a, H)$ and forwards it to $\mathscr{A}$. It keeps g_1^a secret. The private key sk is implicitly set as equal to a.

Adaptive Queries. $\mathscr{A}$ has access to the tag generation oracle $\mathscr{O}_{TG}$ as follows. It first adaptively selects several blocks m_i, for $i \in [1, n]$. $\mathscr{B}$ splits each block m_i into s sectors $m_{i,j}$. Then, it computes

$$T_{m_i} = (\prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (\prod_{j=1}^{s} h_j^{m_{i,j}})^{-a} = \prod_{j=1}^{s} (g_1^{\beta_j})^{m_{i,j} \cdot (-a)} = \prod_{j=1}^{s} (g_1^a)^{-\beta_j \cdot m_{i,j}}$$

for $i \in [1, n]$, and gives them to $\mathscr{A}$. The adversary sets an ordered collection $\mathbb{F} = \{m_1, \cdots, m_n\}$ of blocks and an ordered collection $\mathbb{E} = \{T_{m_1}, \cdots, T_{m_n}\}$ which are the verification metadata corresponding to the blocks in $\mathbb{F}$.

$\mathscr{A}$ has access to the data operation performance oracle $\mathscr{O}_{DOP}$ as follows. Repeatedly, the adversary selects a block m_l and the corresponding element $info_l$ and forwards them to the challenger. The index l denotes the rank where $\mathscr{A}$ wants the data operation to be performed. More precisely, l is equal to $\frac{i_1+i_2}{2}$ for an insertion and to i for a deletion or a modification. Moreover, $m_l = \perp$ in the case of a deletion, since only the rank is needed to perform this kind of operation. Then, $\mathscr{A}$ outputs a new ordered collection $\mathbb{F}'$ (containing the updated version of the block m_l), a new verification metadata ordered collection $\mathbb{E}'$ (containing the updated version of the verification metadata T_{m_l}) and a corresponding updating proof $\nu' = (U_1, \cdots, U_s, C_1, \cdots, C_s, d, w_l)$, such that w_l is randomly chosen from $\mathbb{Z}_p$, $d = T_{m_l}^{w_l}$, and u_j is randomly chosen from $\mathbb{Z}_p$, $U_j = h_j^{u_j}$, $c_j = m_{l,j} \cdot w_l + u_j$ and $C_j = h_j^{c_j}$, for $j \in [1, s]$. $\mathscr{B}$ runs the algorithm $\mathsf{CheckOp}$ on the value ν' and sends the answer to $\mathscr{A}$. If the answer is $false$, then the challenger aborts; otherwise, it proceeds.

Challenge. The adversary selects some blocks m_i^* and the corresponding elements $info_i^*$, for $i \in \mathscr{I} \subseteq (0, n+1) \cap \mathbb{Q}$, and forwards them to the challenger who checks the data operations. In particular, the first $info_i^*$ indicates a full re-write.

Then, the challenger chooses a subset $I \subseteq \mathscr{I}$, randomly chooses $|I|$ elements $v_i \in_R \mathbb{Z}_p$ and sets $chal = \{(i, v_i)\}_{i \in I}$. It forwards $chal$ as a challenge to $\mathscr{A}$.

Forgery. Upon receiving the challenge $chal$, the resulting proof of data possession on the correct stored document m should be $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c)$ and pass the equation 7.2, where $R_j = h_j^{r_j} = g_1^{\beta_j r_j}$ for random exponents $r_j \in_R \mathbb{Z}_p$, $B_j =$

$h_j^{b_j} = g_1^{\beta_j b_j} = g_1^{\beta_j(\sum_{(i,v_i)\in chal} v_i \cdot m_{i,j} + r_j)}$ and $c = \prod_{(i,v_i)\in chal} T_{m_i}^{v_i}$. However, $\mathscr{A}$ generates a proof of data possession on an incorrect stored document $\tilde{m}$ set as $\tilde{v} = (\tilde{R}_1, \cdots, \tilde{R}_s, \tilde{B}_1, \cdots, \tilde{B}_s, \tilde{c})$, where $\tilde{R}_j = h_j^{\tilde{r}_j} = g_1^{\beta_j \tilde{r}_j}$ for random exponents $\tilde{r}_j \in_R \mathbb{Z}_p$, and $\tilde{B}_j = h_j^{\tilde{b}_j} = g_1^{\beta_j \tilde{b}_j} = g_1^{\beta_j(\sum_{(i,v_i)\in chal} v_i \cdot \tilde{m}_{i,j} + \tilde{r}_j)}$, as well as $\tilde{c} = \prod_{(i,v_i)\in chal} T_{\tilde{m}_i}^{v_i}$. Finally, the adversary returns $\tilde{v}$ to the challenger. If the proof of data possession still pass the verification, then $\mathscr{A}$ wins. Otherwise, it fails.

Analysis. We define $\Delta r_j = \tilde{r}_j - r_j$, $\Delta b_j = \tilde{b}_j - b_j$ and $\Delta\tau_j = \tilde{b}_j - b_j - (\tilde{r}_j - r_j) = \sum_{(i,v_i)\in chal} v_i(\tilde{m}_{i,j} - m_{i,j})$, for $j \in [1,s]$. At least one element of $\{\Delta\tau_j\}_{j\in[1,s]}$ is non-zero.

We prove that if the adversary can win the game, then solutions to the DHI and the DL problems are found, which contradicts the assumption that the DHI and the DL problems are hard in $\mathbb{G}$ and $\mathbb{G}_1$ respectively. Let assume that the adversary (playing the role of the server) wins the game. We recall that if $\mathscr{A}$ wins then $\mathscr{B}$ can extract the actual blocks $\{m_i\}_{(i,v_i)\in chal}$ in polynomially-many interactions with $\mathscr{A}$. Without loss of generality, suppose that $chal = \{(i,v_i)\}$ (i.e. the challenge contains only one block).

First case ($\tilde{c} \neq c$)**:** According to the equation 7.2, we have

$$\begin{aligned} e(\frac{\tilde{c}}{c}, g_2^a) &= \frac{e(\prod_{j=1}^s \tilde{B}_j, g_2)}{e(\prod_{j=1}^s \tilde{R}_j, g_2)} \cdot \left(\frac{e(\prod_{j=1}^s B_j, g_2)}{e(\prod_{j=1}^s R_j, g_2)}\right)^{-1} \\ &= e(\prod_{j=1}^s h_j^{\Delta b_j}, g_2) \cdot e(\prod_{j=1}^s h_j^{-\Delta r_j}, g_2) \\ &= e(\prod_{j=1}^s h_j^{\Delta\tau_j}, g_2) = e(g_1^{\sum_{j=1}^s \beta_j \cdot \Delta\tau_j}, g_2) \end{aligned}$$

and so, we gets

$$\begin{aligned} \left(e(\frac{\tilde{c}}{c}, g_2^a)\right)^{1/a} &= \left(e(\prod_{j=1}^s h_j^{\Delta\tau_j}, g_2)\right)^{1/a} \\ e(\frac{\tilde{c}}{c}, g_2)^{a\cdot\frac{1}{a}} &= e(g_1^{\sum_{j=1}^s \beta_j \cdot \Delta\tau_j}, g_2)^{1/a} \\ e(\frac{\tilde{c}}{c}, g_2) &= e(g_1^{\frac{\sum_{j=1}^s \beta_j \cdot \Delta\tau_j}{a}}, g_2) \end{aligned}$$

meaning that we have found the solution to the DHI problem, that is

$$g_1^{1/a} = (\frac{\tilde{c}}{c})^{\frac{1}{\sum_{j=1}^s \beta_j \cdot \Delta\tau_j}}$$

unless evaluating the exponent causes a divide-by-zero. Nevertheless, we notice that not all of the $\Delta\tau_j$ can be zero (indeed, if $\tau_j = m_{i,j} \cdot v_i = \tilde{\tau}_j = \tilde{m}_{i,j} \cdot v_i$ for each $j \in [1,s]$, then $c = \tilde{c}$ which contradicts the hypothesis, and the values β_j are information theoretically hidden from $\mathscr{A}$ (Pedersen commitments), so the denominator is zero only with probability $1/p$, which is negligible. Finally, since $\mathscr{B}$ knows the exponent x such that $g_1 = g^x$, it can directly compute

$$g^{1/a} = ((g^x)^{1/a})^{1/x} = (g_1^{1/a})^{1/x} = \left((\frac{\tilde{c}}{c})^{\frac{1}{\sum_{j=1}^s \beta_j \cdot \Delta\tau_j}}\right)^{1/x}$$

and obtains $g^{1/a}$. Thus, if $\mathscr{A}$ wins the game, then a solution to the DHI problem can be found with probability equal to $1 - 1/p$.

Second case ($\tilde{c} = c$): According to the equation 7.2, we have $e(\tilde{c}, g_2^a) \cdot e(\prod_{j=1}^s \tilde{R}_j, g_2) = e(\prod_{j=1}^s \tilde{B}_j, g_2)$. Since the proof $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c)$ is a correct one, we also have $e(c, g_2^a) \cdot e(\prod_{j=1}^s R_j, g_2) = e(\prod_{j=1}^s B_j, g_2)$. We recall that $chal = \{(i, v_i)\}$. From the previous analysis step, we know that $\tilde{c} = c$. Therefore, we get that $\prod_{j=1}^s \tilde{B}_j \cdot (\prod_{j=1}^s \tilde{R}_j)^{-1} = \prod_{j=1}^s B_j \cdot (\prod_{j=1}^s R_j)^{-1}$. We can write as $\prod_{j=1}^s h_j^{\tilde{b}_j - \tilde{r}_j} = \prod_{j=1}^s h_j^{b_j - r_j}$ or even as $\prod_{j=1}^s h_j^{\Delta b_j - \Delta r_j} = \prod_{j=1}^s h_j^{\Delta \tau_j} = 1$. For two elements $g_1, h \in \mathbb{G}_1$, there exists $\xi \in \mathbb{Z}_p$ such that $h = g_1^{\xi}$ since $\mathbb{G}_1$ is a cyclic group. Without loss of generality, given $g_1, h \in \mathbb{G}_1$, each h_j should randomly and correctly be generated by computing $h_j = g_1^{y_j} \cdot h^{z_j} \in \mathbb{G}_1$ such that y_j and z_j are random values in $\mathbb{Z}_p$. Then, we have $1 = \prod_{j=1}^s h_j^{\Delta \tau_j} = \prod_{j=1}^s (g_1^{y_j} \cdot h^{z_j})^{\Delta \tau_j} = g_1^{\sum_{j=1}^s y_j \cdot \Delta \tau_j} \cdot h^{\sum_{j=1}^s z_j \cdot \Delta \tau_j}$. Clearly, we can find a solution to the DL problem. More specifically, given $g_1, h = g_1^{\xi} \in \mathbb{G}_1$, we can compute $h = g_1^{\frac{\sum_{j=1}^s y_j \cdot \Delta b_j}{\sum_{j=1}^s z_j \cdot \Delta \tau_j}} = g_1^{\xi}$ unless the denominator is zero. However, as we suggested earlier, at least one element of $\{\Delta \tau_j\}_{j \in [1,s]}$ is non-zero. Since z_j is a random element of $\mathbb{Z}_p$, the denominator is zero with probability equal to $1/p$, which is negligible. Thus, if the adversary wins the game, then a solution of the DL problem can be found with probability equal to $1 - \frac{1}{p}$, which contradicts the fact that the DL problem is assumed to be hard in $\mathbb{G}_1$.

Therefore, for the adversary, it is computationally infeasible to win the game and generate an incorrect proof of data possession which can pass the verification.

The simulation of the verification metadata generation oracle $\mathscr{O}_{TG}$ is perfect. The simulation of the data operation performance oracle $\mathscr{O}_{DOP}$ is almost perfect unless $\mathscr{B}$ aborts. This happens when the data operation was not correclty performed. As previously, we can prove that if $\mathscr{A}$ can pass the updating proof, then solutions to the DHI and DL problems are found in $\mathbb{G}$ and $\mathbb{G}_1$ respectively. Following the above analysis and according to Equation 7.1, if $\mathscr{A}$ generates an incorrect updating proof which can pass the verification, then solutions to the DHI and DL problems can be found with probabilities both equal to $1 - \frac{1}{p}$. Therefore, for $\mathscr{A}$, it is computationally infeasible to generate an incorrect updating proof which can pass the verification. The proof is completed.

Proof for the Data Privacy against the TPA

We start by giving the privacy proof following the strong security model. Note that this proof is actually wrong; indeed, we will explain later that the basic scheme DPDP is not strongly data private since it is threatened by an attack against the data privacy given in Section 7.2. We will then present a correct data privacy proof following the weak security model.

Based on the Strong Model.

Theorem. Let $\mathscr{A}$ be a PPT adversary that interacts with a challenger $\mathscr{B}$ in breaking the strong data privacy property of the basic DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$ with advantage ε.

For any probabilistic PPT adversary $\mathscr{A}$ who wins the game, there is a challenger $\mathscr{B}$ that interacts with the adversary $\mathscr{A}$ as follows.

Setup. $\mathscr{B}$ runs $\mathscr{G}$ on input λ to obtain the tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, and selects two generators g_1 and g_2 belonging to $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Then, it randomly chooses s elements $h_1, \cdots, h_s \in_R \mathbb{G}_1$ and an element $\alpha \in_R \mathbb{Z}_p$. It sets the public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^{\alpha})$ and forwards it to $\mathscr{A}$. It sets the private key $sk = \alpha$ and keeps it.

Queries. $\mathscr{A}$ gives to the challenger two documents $m_0 = (m_{0,1}, \cdots, m_{0,n})$ and $m_1 = (m_{1,1}, \cdots, m_{1,n})$ of equal length. $\mathscr{B}$ randomly selects a bit $\mu \in_R \{0,1\}$ and for $i \in [1,n]$, splits each block $m_{\mu,i}$ into s sectors $m_{\mu,i,j}$. Then, it computes $T_{m_{\mu,i}} = (\prod_{j=1}^{s} h_j^{m_{\mu,i,j}})^{-sk} = (\prod_{j=1}^{s} h_j^{m_{\mu,i,j}})^{-\alpha}$, for $i \in [1,n]$, and gives them to $\mathscr{A}$.

Challenge. The adversary chooses a subset $I \subseteq [1,n]$, randomly chooses $|I|$ elements $v_i \in_R \mathbb{Z}_p$ and sets $chal = \{(i, v_i)\}_{i \in I}$. It forwards $chal$ as a challenge to $\mathscr{B}$.

Upon receiving the challenge $chal$, the challenger selects an ordered collection $F = \{m_i\}_{i \in I}$ of blocks and an ordered collection $\Sigma = \{T_{m_i}\}_{i \in I}$ which are the verification metadata corresponding to the blocks in F such that $T_{m_i} = (\prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (\prod_{j=1}^{s} h_j^{m_{i,j}})^{-\alpha}$, for $i \in I$. It then randomly chooses $r_1, \cdots, r_s \in_R \mathbb{Z}_p$ and computes $R_1^* = h_1^{r_1}, \cdots, R_s^* = h_s^{r_s}$. It also randomly selects $b_1, \cdots, b_s \in \mathbb{Z}_p$ and computes $B_1^* = h_1^{b_1}, \cdots, B_s^* = h_s^{b_s}$. It sets $c^* = \prod_{(i,v_i) \in chal} T_{m_i}^{v_i}$ as well. Finally, the challenger returns $\mathcal{V}^* = (R_1^*, \cdots, R_s^*, B_1^*, \cdots, B_s^*, c^*)$.

Guess The adversary returns a bit $\mu' \in \{0,1\}$ as a guess for μ. The adversary wins if $\mu' = \mu$.

Analysis. The probability $Pr[\mu' = \mu]$ must be equal to $\frac{1}{2}$ since the verification metadata $T_{m_{\mu,i}}$, for $i \in [1,n]$, and the proof $\mathcal{V}^*$ are independent regarding the bit μ. We now prove that the verification metadata and the proof of data possession given to the adversary are correctly distributed. The value $T_{m_{\mu,i}}$ is equal to $(\prod_{j=1}^{s} h_j^{m_{\mu,i,j}})^{-sk} = (\prod_{j=1}^{s} h_j^{m_{\mu,i,j}})^{-\alpha}$. Since $sk = \alpha$ is kept secret from $\mathscr{A}$, the above simulation is perfect. For a document m_μ, there exists $v_{\mu,i}$, for $(i, v_{\mu,i}) \in chal_\mu$, such that $b_{\mu,j} = \sum_{(i,v_{\mu,i}) \in chal_\mu} m_{\mu,i,j} \cdot v_{\mu,i} + r_{\mu,j}$. In addition, $R_{\mu,1}, \cdots, R_{\mu,s}, B_{\mu,1}, \cdots, B_{\mu,s}$ are statically indistinguishable with the actual outputs corresponding to m_0 or m_1. Thus, the answers given to the adversary are correctly distributed. The proof is completed.

N.B. The above analysis is wrong: the affirmation "The probability $Pr[\mu' = \mu]$ must be equal to $\frac{1}{2}$ since the verification metadata $T_{m_{\mu,i}}$, for $i \in [1,n]$, and the proof $\mathcal{V}^*$ are independent regarding the bit μ." is incorrect: $T_{m_{\mu,i}}$ and $\mathcal{V}^*$ actually depend on μ.

A solution to correct the above security proof is to follow the weak security model instead of the strong one that is based on indistinguishability. Since unfortunately, the basic DPDP system with public verifiability and data privacy DPDP does not satisfy such property.

Based on the Weak Model.

Theorem. Let $\mathscr{A}$ be a PPT adversary that has advantage ε against the weak data privacy property of the DPDP scheme with public verifiability and data privacy $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$. Then, there is a challenger $\mathscr{B}$ that solves the DHI problem in $\mathbb{G}$ with advantage $\varepsilon' = O(\varepsilon)$.

For any PPT adversary $\mathscr{A}$ who wins the game, there is a challenger $\mathscr{B}$ that wants to break the DHI assumption in $\mathbb{G}$ by interacting with $\mathscr{A}$ as follows.

Setup. $\mathscr{B}$ runs $\mathsf{GroupGen}(\lambda) \to (p, \mathbb{G}, \mathbb{G}_T, e)$. Then, it is given the DHI instance tuple (g, g^a) such that g is generator of $\mathbb{G}$ and $a \in \mathbb{Z}_p^*$, chooses two exponents $x, y \in \mathbb{Z}_p$ and computes $g_1 = g^x$ and $g_2 = g^y$. It also sets $\mathbb{G}_1 =< g_1 >$ and $\mathbb{G}_2 =< g_2 >$. Note that $(g^a)^x = g_1^a$ and $(g^a)^y = g_2^a$. $\mathscr{B}$ chooses $\beta_j \in_R \mathbb{Z}_p$ and sets $h_j = g_1^{\beta_j}$ for $j \in [1,s]$. $\mathscr{B}$ sets the public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^a, H)$ and forwards it to $\mathscr{A}$. It keeps g_1^a secret. The private key sk is implicitly set as equal to a.

Queries. $\mathscr{A}$ makes queries to the verification metadata generation oracle as follows. It first adaptively selects blocks m_i, for $i \in [1,n]$. $\mathscr{B}$ splits each block m_i into s sectors $m_{i,j}$. Then, it computes $T_{m_i} = (\prod_{j=1}^s h_j^{m_{i,j}})^{-sk} = (\prod_{j=1}^s h_j^{m_{i,j}})^{-a} = \prod_{j=1}^s (g_1^{\beta_j})^{m_{i,j}\cdot(-a)} = \prod_{j=1}^s (g_1^a)^{-\beta_j \cdot m_{i,j}}$.

$\mathscr{B}$ gives the blocks and the verification metadata to $\mathscr{A}$. The latter sets an ordered collection $\mathbb{F} = \{m_1, \cdots, m_n\}$ of blocks and an ordered collection $\mathbb{E} = \{T_{m_1}, \cdots, T_{m_n}\}$ which are the verification metadata corresponding to the blocks in $\mathbb{F}$.

Challenge. The adversary submits a challenge $chal = \{(i, v_i)\}_{i \in I}$. Without loss of generality, we suppose there is only element i in I. As a shorthand, we write $chal = \{(i, v_i)\}$. Moreover, $\mathscr{A}$ gives an ordered collection $F = \{\tilde{m}_i\} \cap \mathbb{F} = \emptyset$ of the data block determined by $chal$, and an ordered collection $\Sigma = \{T_{\tilde{m}_i}\} \cap \mathbb{E} = \emptyset$ of the corresponding verification metadata. Note that there are only one element $\tilde{m}_i$ in F and one element $T_{\tilde{m}_i}$ in Σ.

Generation of the Proof. Upon receiving the challenge $chal = \{(i, v_i)\}$, the collection $F = \{\tilde{m}_i\}$ and the collection $\Sigma = \{T_{\tilde{m}_i}\}$, the challenger generates the proof of data possession $\tilde{\nu} = (\tilde{R}_1, \cdots, \tilde{R}_s, \tilde{B}_1, \cdots, \tilde{B}_s, \tilde{c})$, such that $\tilde{r}_j$ is randomly chosen from $\mathbb{Z}_p$, $\tilde{R}_j = h_j^{\tilde{r}_j}$, $\tilde{b}_j = \sum_{(i,v_i) \in chal} \tilde{m}_{i,j} \cdot v_i + \tilde{r}_j$ and $\tilde{B}_j = h_j^{\tilde{b}_j}$, for $j \in [1,s]$. It also sets $\tilde{c} = \prod_{(i,v_i) \in chal} T_{\tilde{m}_i}^{v_i}$.

Finally, the challenger runs $\mathsf{CheckProof}$ on $\tilde{\nu}$. If the proof of data possession still pass the verification, then $\mathscr{A}$ wins. Otherwise, it fails.

Analysis. Given an actual data block m_i and the corresponding verification metadata T_{m_i}, let $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c)$ be a honest proof of data possession that pass the verification. Both ν and $\tilde{\nu}$ pass the verification, and so verify the equation 7.2.

We define $\Delta r_j = \tilde{r}_j - r_j$, $\Delta b_j = \tilde{b}_j - b_j = \sum_{(i,v_i) \in chal} (\tilde{m}_{i,j} - m_{i,j}) v_i + \Delta r_j$ and $\Delta \tau_j = \sum_{(i,v_i) \in chal} (\tilde{m}_{i,j} - m_{i,j}) v_i = (\tilde{m}_{i,j} - m_{i,j}) v_i$, for $j \in [1,s]$. Note that r_j and b_j are the elements of a honest proof of data possession ν such that $r_j \in_R \mathbb{Z}_p$ and $b_j = \sum_{(i,v_i) \in chal} m_{i,j} \cdot v_i + r_j = m_{i,j} \cdot v_i + r_j$ where $m_{i,j}$ are the sectors blocks (not the ones that the adversary claims to have).

We prove that if the adversary can win the game, then a solution to the DHI problem is found, which contradicts the assumption that the DHI problem is hard in $\mathbb{G}$. Let assume that the adversary (playing the role of the TPA) wins the game.

According to the equation 7.2 and since $T_{\tilde{m}_i} \neq T_{m_i}$ giving that $\tilde{c} \neq c$, we have

$$\begin{aligned} e(\frac{\tilde{c}}{c}, g_2^a) &= \frac{e(\prod_{j=1}^{s} \tilde{B}_j, g_2)}{e(\prod_{j=1}^{s} \tilde{R}_j, g_2)} \cdot \left(\frac{e(\prod_{j=1}^{s} B_j, g_2)}{e(\prod_{j=1}^{s} R_j, g_2)} \right)^{-1} \\ &= e(\prod_{j=1}^{s} h_j^{\Delta b_j}, g_2) \cdot e(\prod_{j=1}^{s} h_j^{-\Delta r_j}, g_2) \\ &= e(\prod_{j=1}^{s} h_j^{\Delta \tau_j}, g_2) = e(g_1^{\sum_{j=1}^{s} \beta_j \cdot \Delta \tau_j}, g_2) \end{aligned}$$

and so, we gets

$$\begin{aligned} \left(e(\frac{\tilde{c}}{c}, g_2^a) \right)^{1/a} &= \left(e(\prod_{j=1}^{s} h_j^{\Delta \tau_j}, g_2) \right)^{1/a} \\ e(\frac{\tilde{c}}{c}, g_2)^{a \cdot \frac{1}{a}} &= e(g_1^{\sum_{j=1}^{s} \beta_j \cdot \Delta \tau_j}, g_2)^{1/a} \\ e(\frac{\tilde{c}}{c}, g_2) &= e(g_1^{\frac{\sum_{j=1}^{s} \beta_j \cdot \Delta \tau_j}{a}}, g_2) \end{aligned}$$

meaning that we have found the solution to the DHI problem, that is

$$g_1^{1/a} = (\frac{\tilde{c}}{c})^{\frac{1}{\sum_{j=1}^{s} \beta_j \cdot \Delta \tau_j}}$$

unless evaluating the exponent causes a divide-by-zero. Nevertheless, we notice that not all of the $\Delta \tau_j$ can be zero (indeed, if $\tau_j = m_{i,j} \cdot v_i = \tilde{\tau}_j = \tilde{m}_{i,j} \cdot v_i$ for each $j \in [1, s]$, then $c = \tilde{c}$ which contradicts the hypothesis, and the values β_j are information theoretically hidden from $\mathscr{A}$ (Pedersen commitments), so the denominator is zero only with probability $1/p$, which is negligible. Finally, since $\mathscr{B}$ knows the exponent x such that $g_1 = g^x$, it can directly compute

$$g^{1/a} = ((g^x)^{1/a})^{1/x} = (g_1^{1/a})^{1/x} = \left((\frac{\tilde{c}}{c})^{\frac{1}{\sum_{j=1}^{s} \beta_j \cdot \Delta \tau_j}} \right)^{1/x}$$

and obtains $g^{1/a}$. Thus, if $\mathscr{A}$ wins the game, then a solution to the DHI problem can be found with probability equal to $1 - 1/p$.

7.1.6 Performance

We evaluate the efficiency of our scheme CBEKS in Table 7.1. We consider cryptographic operations such as group multiplication on random group elements, group exponentiation of a random group element with a random exponent, as well as pairing operation on random group elements. We use implementation results of these operations in the CHARM framework [4, 152] based on Miyaji-Nakabayashi-Takano (MNT) [162] curves. We do not take into account multiplication in $\mathbb{Z}_p$ as this operation has timings negligible compared to the ones for exponentiation and pairing. The benchmarks were executed on a quad core processor AMD A10-5800K with 16GB of RAM running Fedora 18.

We evaluated the protocol algorithms on the asymmetric bilinear group based on the MNT elliptic curve provided by CHARM. Complex multiplication is required in order to find curves of a certain embedding degree with a specified order. In particular, MNT curves have an embedding degree $k = 6$.

We assume that 2GB data are stored. The document is split into one million blocks of size 2KB, such that the size of the index is $|n| = 20$ bits. We assume that the number of sectors in each data block is $s = 100$ and the number of blocks determined in the challenge *chal* is $|I| = 460$. Note that the challenge requires 460 blocks to allow the detection of 1% fraction of incorrect data with 99% probability of detecting misbehavior [12]. More precisely, the probability of detecting corrupted block is $1-(1-|X|)^{|I|}$, where $|X|$ is the number of corrupted blocks.

	Mult. in $\mathbb{G}_1$	Mult. in $\mathbb{G}_T$	Exp. in $\mathbb{G}_1$	Exp. in $\mathbb{G}_2$	Exp. in $\mathbb{G}_T$	Pairing
Time/op.	0.002558	0.006992	0.5895	5.314	1.1	3.798
KeyGen				5.314		
TagGen	0.2558		58.95			
PerfOp			118.4895			
CheckOp	0.5116	0.006992				11.394
GenProof	1.17668		118.4895			
CheckProof	0.5116	0.006992				11.394

Table 7.1: Timings for our DPDP asymmetric pairing-based system. Times are in milliseconds. Time/op. refers to Time/operation, Mult. refers to the multiplication operation and Exp. refers to the group exponentiation operation.

The algorithm KeyGen is executed only once and does not require much computation. Similarly, the algorithm TagGen is run only once per file. The timing is mainly due to the file being split into $s = 100$ blocks. The algorithms PerfOp and GenProof differentiate since only one block is considered in PerfOp while $chal = 460$ blocks are counted in GenProof. Running times are equal for the algorithms CheckOp and CheckProof since these algorithms execute the same process.

7.1.7 Computational and Communication Costs: Comparison with Existing Schemes

In Table 7.2, we compare the computational costs of our scheme with the ones in [224, 251]. We choose the schemes presented in [224, 251] since they are recent and offer similar features to our scheme.

In all the schemes, during the execution of the algorithm KeyGen, the number of exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_2$ is constant. The algorithm TagGen in [251] and in ours requires $O(s)$ exponentiations in $\mathbb{G}_1$, where s is the number of sectors in each data block. In [224], TagGen needs only a constant number of exponentiations in $\mathbb{G}_1$, however there is an extra computational cost with the generation of the verification metadata, which is the cost of computing three exponentiations in $\mathbb{G}_T$ and three pairings. Moreover, the number of multiplications in $\mathbb{G}_1$ is constant in [224, 251], whereas it is linear in s in our case. In [251] and in our scheme, the generation of the proof of prossession in GenProof needs the computation of $O(s+|I|)$ exponentiations in $\mathbb{G}_1$; whereas in [224], the generation of the proof of prossession only involves the computation of $O(|I|)$ exponentiations in $\mathbb{G}_1$. In addition, the number of multiplications in $\mathbb{G}_1$ is linear in $|I|$ in our scheme and in [224], while it is linear in both $|I|$ and s in [251]. The computational cost due to the check of the proof in CheckProof differs in the three schemes. In [224], the algorithm CheckProof requires $O(|I|)$ exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$ and four pairings. In [251], the algorithm CheckProof needs $O(s+|I|)$ exponentiations in $\mathbb{G}_1$ and a constant number of pairings equal to three. Moreover, the number of multiplications in both $\mathbb{Z}_p$ and $\mathbb{G}_1$ varies between the three systems. In [224], $O(s+|I|)$ and $O(|I|)$ multiplications are required in $\mathbb{Z}_p$ and $\mathbb{G}_1$ respectively. In [251], $O(s+|I|)$ multiplications in $\mathbb{G}_1$ are computed, while $O(s)$ multiplications in $\mathbb{G}_1$ are needed in our case. Finally, in our scheme, the computation cost is lighter; more precisely, it is only the cost of computing three pairings (no exponentiation is required).

The communication cost of our protocol is mostly due to two factors: the challenge and the proof of data possession. The communication cost of a challenge $chal = \{(i, v_i)\}_{i \in I}$ is $|I|(|n|+|p|)$ bits, where $|I|$ is the number of selected data blocks, $|n|$ is the length of an index and $|p|$ is the length of an element in $\mathbb{Z}_p$. The communication cost of a proof of data possession $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c)$ is $(2 \cdot s + 1)|p|$ bits, where s is the number of sectors in each block. An additional cost can be the communication cost of an updating proof $\nu' = (U_1, \cdots, U_s, C_1, \cdots, C_s, d)$, which is $(2 \cdot s + 1)|p|$ bits. However, this happens only when the client wants to update his/her data. We assume that the frequency of checking the integrity of the data is much higher than the frequency of performing data operations. Therefore, we leave out this additional cost.

In Table 7.3, we evaluate the computational cost of the verification of the proof and communication costs of the verification request and of the proof of data possession in our scheme and compare them with the ones in the DPDP scheme proposed in [251]. We note that the communication costs are equivalent in both schemes. These incurred costs are negligible compared to the 2GB total size of the stored data. The time required to create the proof of data possession on the server side is very efficient, even if the client has to perform some work for the verification. The latter observation is likeable; since the server should be the entity that has more workload. Moreover, the more noticeable difference between the two DPDP protocols appears in the verification of the proof of data possession. Actually, the computational cost to generate the proof in our case is *constant* whereas this cost is linear in parameters related to the number of blocks (and of sectors) in [251]. The overall result demonstrates the practicality of our scheme in comparison to the existing scheme from [251].

7.2 Replace Attack, Replay Attack and Attack against Data Privacy

In this section, we carefully describe how the three attacks threaten the DPDP scheme with public verifiability and data privacy in Section 7.1.

7.2.1 Replace Attack

We explain the first attack on the basic scheme DPDP given in Section 7.1. Informally, the attack works as follows. Let us assume that the server stores only one block, say m_1

DPDP scheme	Algorithm	Operation total cost						Pairing type	Public verifiability	Dynamicity
from [224]	KeyGen	-	-	$3\mathrm{E}_{\mathbb{G}_1}$	$2\mathrm{E}_{\mathbb{G}_2}$	-	-	asymmetric	×	×
	TagGen	$1\mathrm{M}_{\mathbb{Z}_p}$	$1\mathrm{M}_{\mathbb{G}_1}$	$11\mathrm{E}_{\mathbb{G}_1}$	-	$3\mathrm{E}_{\mathbb{G}_T}$	3P			
	GenProof	$\|I\|\mathrm{M}_{\mathbb{Z}_p}$	$(\|I\|-1)\mathrm{M}_{\mathbb{G}_1}$	$\|I\|\mathrm{E}_{\mathbb{G}_1}$	-	-	-			
	CheckProof	$(s+\|I\|)\mathrm{M}_{\mathbb{Z}_p}$	$(9\|I\|-6)\mathrm{M}_{\mathbb{G}_1}$	$(8+8\|I\|)\mathrm{E}_{\mathbb{G}_1}$	-	$\|I\|\mathrm{E}_{\mathbb{G}_T}$	4P			
from [251]	KeyGen	-	-	$2\mathrm{E}_{\mathbb{G}_1}$	/	-	-	symmetric	✓	✓
	TagGen	-	$1\mathrm{M}_{\mathbb{G}_1}$	$(s+2)\mathrm{E}_{\mathbb{G}_1}$	/	-	-			
	GenProof	$\|I\|\mathrm{M}_{\mathbb{Z}_p}$	$(s+\|I\|-2)\mathrm{M}_{\mathbb{G}_1}$	$(s+\|I\|+1)\mathrm{E}_{\mathbb{G}_1}$	/	-	1P			
	CheckProof	-	$(s+\|I\|-2)\mathrm{M}_{\mathbb{G}_1}$	$(s+\|I\|)\mathrm{E}_{\mathbb{G}_1}$	/	-	3P			
In this section	KeyGen	-	-	-	$1\mathrm{E}_{\mathbb{G}_2}$	-	-	asymmetric	✓	✓
	TagGen	-	$(s-1)\mathrm{M}_{\mathbb{G}_1}$	$s\mathrm{E}_{\mathbb{G}_1}$	-	-	-			
	GenProof	$\|I\|\mathrm{M}_{\mathbb{Z}_p}$	$(\|I\|-1)\mathrm{M}_{\mathbb{G}_1}$	$(2s+\|I\|)\mathrm{E}_{\mathbb{G}_1}$	-	-	-			
	CheckProof	-	$(2s-2)\mathrm{M}_{\mathbb{G}_1}$	-	-	-	3P			

Table 7.2: Group multiplication, group exponentiation and pairing benchmarks. $\mathrm{M}_{\mathbb{Z}_p}$ and $\mathrm{M}_{\mathbb{G}_1}$ denote multiplications in $\mathbb{Z}_p$ and $\mathbb{G}_1$ respectively. $\mathrm{E}_{\mathbb{G}_1}$, $\mathrm{E}_{\mathbb{G}_2}$ and $\mathrm{E}_{\mathbb{G}_T}$ denote exponentiations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ respectively. P denotes pairing in $\mathbb{G}_T$. Let s be the number of sector in one data block and $|I|$ be the cardinality of the set $I \subseteq (0,n+1) \cap \mathbb{Q}$ when setting the challenge *chal*. The sign “-” means that there is no group operation performed. Let “/” point out symmetric pairings, i.e. $\mathbb{G}_1 = \mathbb{G}_2$ as in [251].

	DPDP in [251]	Our DPDP
Communication cost of the verification request	30.59	30.59
Communication cost of the proof of data possession	6.464	12.864
Computational cost of the verification of the proof	36.334	0.43

Table 7.3: Costs in kilobytes (KB) of the verification request sent by the TPA (on behalf of the client) to the server, the proof of data possession generated by the server for the TPA, and the verification of the proof of data possession done by the TPA.

without loss of generality, instead of n blocks as the client believes. The TPA on behalf of the client regularly audits the server by sending it a challenge *chal* for blocks with indices in a set $I \subseteq [1,n]$, such that $|sub| = k \leq n$. The server generates a proof of data possession on the k blocks m_1 (instead of the blocks defined by *chal*). In other words, the server uses k times the block m_1 to obtain the proof of data possession according to the size of the index set used for the challenge. The attack is successful if the server manages to pass the verification process and has its proof of data possession being accepted by the TPA.

We recall how the client generates the verification metadata for a document m that s/he wants to upload on the server.

- $\mathsf{TagGen}(pk, sk, m) \rightarrow T_m$. A document m is split into n blocks m_i, for $i \in [1,n]$. Each block m_i is then split into s sectors $m_{i,j} \in \mathbb{Z}_p$, for $j \in [1,s]$. Therefore, the document m can be seen as a $n \times s$ matrix with elements denoted as $m_{i,j}$. The client computes the verification metadata $T_{m_i} = (\prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (\prod_{j=1}^{s} h_j^{m_{i,j}})^{-\alpha} = \prod_{j=1}^{s} h_j^{-\alpha \cdot m_{i,j}}$ for $i \in [1,n]$. Then, s/he sets $T_m = (T_{m_1}, \cdots, T_{m_n}) \in \mathbb{G}_1^n$.

  Moreover, the client stores all the data blocks m_i in an ordered collection $\mathbb{F}$ and the corresponding verification metadata T_{m_i} in an ordered collection $\mathbb{E}$. S/he forwards these two collections to the server and deletes them from his/her local storage.

  Yet, the server is asked to generate a proof of data possession. However, let us assume that it only stores the first block m_1 of the document m (it has deleted all other blocks) and we show that it can still pass the verification process.

- $\mathsf{GenProof}(pk, F, chal, \Sigma) \rightarrow \nu$. The TPA prepares a challenge *chal* to send to the server as follows. First, it chooses a subset $I \subseteq (0, n+1) \cap \mathbb{Q}$, randomly chooses $|I|$ elements $v_i \in_R \mathbb{Z}_p$ and sets $chal = \{(i, v_i)\}_{i \in I}$. Second, after receiving the challenge *chal* which indicates the specific blocks for which the client through the TPA wants a proof of data possession, the server sets the ordered collection $F = \{m_1\}_{i \in I} \subset \mathbb{F}$ of blocks (instead of $F = \{m_i\}_{i \in I}$) and an ordered collection $\Sigma = \{T_{m_1}\}_{i \in I} \subset \mathbb{E}$ which are the verification metadata corresponding to the blocks in F (instead of $\Sigma = \{T_{m_i}\}_{i \in I}$). It then selects at random $r_1, \cdots, r_s \in_R \mathbb{Z}_p$ and computes $R_1 = h_1^{r_1}, \cdots, R_s = h_s^{r_s}$. It also sets $b_j = \sum_{(i,v_i) \in chal} m_{1,j} \cdot v_i + r_j \in \mathbb{Z}_p$ for $j \in [1,s]$ (instead of $b_j = \sum_{(i,v_i) \in chal} m_{i,j} \cdot v_i + r_j$), then $B_j = h_j^{b_j}$, for $j \in [1,s]$, along with $c = \prod_{(i,v_i)\in chal} {T_{m_1}}^{v_i}$ (instead of $c = \prod_{(i,v_i)\in chal} T_{m_i}^{v_i}$). Finally, it returns $\nu = (R_1,\cdots,R_s,B_1,\cdots,B_s,c) \in \mathbb{G}_1^{2s+1}$ to the TPA.

- $\mathsf{CheckProof}(pk, chal, \nu) \rightarrow true/false$. The TPA has to check whether the following equation holds:

$$e(c, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} R_j, g_2) \overset{?}{=} e(\prod_{j=1}^{s} B_j, g_2) \tag{7.3}$$

If the equation 7.3 holds, then the TPA returns $true$ to the client; otherwise, it returns $false$ to the client.

Correctness. For the proof of data possession:

$$\begin{aligned}
e(c, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} R_j, g_2) &= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} {T_{m_1}}^{v_i}, g_2^{\alpha}\right) \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} \prod_{j=1}^{s} h_j^{m_{1,j}\cdot(-\alpha)\cdot v_i}, g_2^{\alpha}\right) \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{\sum_{\substack{(i,v_i)\\ \in chal}} m_{1,j}\cdot v_i}, g_2)^{\alpha-\alpha} \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e\left(\prod_{j=1}^{s} h_j^{\sum_{\substack{(i,v_i)\\ \in chal}} m_{1,j}\cdot v_i + r_j}, g_2\right) \\
&= e(\prod_{j=1}^{s} h_j^{b_j}, g_2) = e(\prod_{j=1}^{s} B_j, g_2)
\end{aligned}$$

Therefore, the equation 7.3 holds, although the server is actually storing one block only.

N.B. This attack is not due to the dynamicity property of the DPDP scheme with public verifiability and data privacy DPDP given in Section 7.1. Such attack could happen even on static data.

7.2.2 Replay Attack

We explain the second successful attack on the basic scheme DPDP given in Section 7.1. Informally, the attack works as follows. The server is storing some data blocks of a client, such that the latter regularly requests the server to updates his/her data according to the three available operations (insertion, deletion and modification). For instance, the client asks the server to modify the block m_i into the block m_i'. However, the server does not proceed and keeps the data block m_i on its storage. Then, the TPA has to check that the operation has been correctly done and asks the server for an updating proof on m_i'. The server generates the updating proof as requested, but using the data block m_i. The attack is successful if the server manages to pass the verification process and has its updating proof being accepted by the TPA.

In more details, a client asks the server to modify the data block m_i by sending the new version of the block m'_i and the corresponding verification metadata $T_{m'_i}$. However, the server does not follow the client's request and decides to keep the old version of the block m_i and the corresponding verification metadata T_{m_i}, and deletes m'_i and $T_{m'_i}$.

- $\mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info = (\text{modification}, i, m'_i, T_{m'_i})) \rightarrow (\mathbb{F}', \mathbb{E}', \nu')$. After receiving the elements i, m'_i and $T_{m'_i}$ from the client, the server prepares the updating proof as follows. It first retrieves the block m_i and the verification metadata T_{m_i} corresponding to the index i, and once it gets these elements, it deletes m'_i and $T_{m'_i}$. It then selects at random $u_1, \cdots, u_s \in_R \mathbb{Z}_p$ and computes $U_1 = h_1^{u_1}, \cdots, U_s = h_s^{u_s}$. It also chooses at random $w_i \in_R \mathbb{Z}_p$ and sets $c_j = m_{i,j} \cdot w_i + u_j \in \mathbb{Z}_p$ (instead of $c_j = m'_{i,j} \cdot w_i + u_j \in \mathbb{Z}_p$) and $C_j = h_j^{c_j}$, for $j \in [1, s]$, and $d = {T_{m_i}}^{w_i}$ (instead of $d = T_{m'_i}^{w_i}$). Finally, it returns $\nu' = (U_1, \cdots, U_s, C_1, \cdots, C_s, d) \in \mathbb{G}_1^{2s+1}$ to the TPA.
- $\mathsf{CheckOp}(pk, \nu') \rightarrow true/false$. The TPA has to check whether the following equation holds:

$$e(d, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} U_j, g_2) \overset{?}{=} e(\prod_{j=1}^{s} C_j, g_2) \tag{7.4}$$

If the equation 7.4 holds, then the TPA returns $true$ to the client; otherwise, it returns $false$ to the client.

Correctness. For the updating proof:

$$\begin{aligned}
e(d, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} U_j, g_2) &= e({T_{m_i}}^{w_i}, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{m_{i,j} \cdot (-\alpha) \cdot w_i}, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{m_{i,j} \cdot w_i}, g_2)^{\alpha - \alpha} \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{m_{i,j} \cdot w_i + u_j}, g_2) \\
&= e(\prod_{j=1}^{s} h_j^{c_j}, g_2) = e(\prod_{j=1}^{s} C_j, g_2)
\end{aligned}$$

Therefore, the equation 7.4 holds, although the server has not updated the block m'_i and the corresponding verification metadata $T_{m'_i}$.

N.B. This attack is due to the dynamicity property of the basic scheme given in Section 7.1.

7.2.3 Attack against Data Privacy

We explain the third successful attack on the DPDP scheme with public verifiability and data privacy DPDP given in Section 7.1. Informally, the TPA (as the adversary) and the server (as the challenger) play the strong data privacy game such that the TPA gives two equal-length blocks m_0 and m_1 to the server and the latter replies to the TPA by sending the verification metadata T_{m_μ} of the data block m_μ, after choosing a bit $\mu \in_R \{0,1\}$. Then, the TPA also selects a bit $\mu' \in \{0,1\}$ The attack is successful if using $m_{\mu'}$, the TPA can discover which block $m_\mu \in \{m_0, m_1\}$ was chosen by the server.

More formally, the data privacy attack on the basic scheme given in Section 7.1 works as follows. The TPA, who plays the role of the adversary, provides two equal-length blocks m_0 and m_1 to the challenger, such that $m_0 = (m_{0,1}, \cdots, m_{0,n})$ and $m_1 = (m_{1,1}, \cdots, m_{1,n})$. The challenger randomly chooses a bit $\mu \in_R \{0,1\}$, computes $T_{m_{\mu,i}} \leftarrow \mathsf{TagGen}(pk, sk, m_{\mu,i})$, for $i \in [1,n]$, and gives them to the TPA. We recall that $T_{m_{\mu,i}}$ is equal to $(\prod_{j=1}^{s} h_j^{m_{\mu,i,j}})^{-sk} = (\prod_{j=1}^{s} h_j^{m_{\mu,i,j}})^{-\alpha}$.

For $i \in [1,n]$, note that

$$e(T_{m_{\mu,i}}, g_2) = e((\prod_{j=1}^{s} h_j^{m_{\mu,i,j}})^{-sk}, g_2) = e((\prod_{j=1}^{s} h_j^{m_{\mu,i,j}})^{-\alpha}, g_2) = e(\prod_{j=1}^{s} h_j^{m_{\mu,i,j}}, (g_2^{\alpha})^{-1}).$$

The computation of the last pairing requires only public elements. Therefore, for $\mu' \in \{0,1\}$, the TPA is able to generate the pairing $e(\prod_{j=1}^{s} h_j^{m_{\mu',i,j}}, (g_2^{\alpha})^{-1})$, given the public key pk and the data block that it gave to the challenger, as well as the pairing $e(T_{m_{\mu,i}}, g_2)$, given the verification metadata sent by the challenger, and finally compares them. If these two pairings are equal, then $\mu' = \mu$; otherwise $\mu' \neq \mu$.

N.B. This attack is due to the public verifiability of the basic scheme DPDP given in Section 7.1, based on the definition of the strong data privacy game.

7.3 IHT-Based Dynamic Provable Data Possession with Public Verifiability and Data Privacy

A solution to avoid the replace attack mentioned in the previous section is to embed the index i of the data block m_i into the verification metadata T_{m_i}. When the TPA on behalf of the client checks the proof of data possession generated by the server, it requires to use all the indices of the challenged data blocks to process the verification. Such idea was proposed for the publicly verifiable scheme in [205].

A solution to avoid the replay attack mentioned in the previous section is to embed the version number vnb_i of the data block m_i into the verification metatdata T_{m_i}. The first time that the client sends the data block m_i to the server, the number vnb_i is set to be equal to 1 (meaning that the first version of the data block is uploaded) and is appended to the index i. When the client wants to modify the data block m_i with m_i', it specifies the number $vnb_i = 2$ (meaning that the second version of data block is uploaded) and generates the verification metadata $T_{m_i'}$ accordingly. When the TPA on behalf of the client checks that the block was correctly updated by the server, it has to use both the index i and the version number vnb_i of the data block.

We stress that the index i of the data block m_i is unique. More precisely, when a block is inserted, a new index is created that has not been used and when a block is modified, the index does not change. However, when a block is deleted, its index does not disappear in order to let the scheme remain secure. To explain why, we consider that the index of a deleted block is removed. Let $m = (m_1, \cdots, m_{10})$ be a document stored on the server. The client first requests to the server to delete the data block m_5. Thus, the index 5 disappears. Later, the client asks to insert a block $m'_{\frac{4+6}{2}} = m'_5$ between the data blocks m_4 and m_6. However, the server might have not properly deleted the previous data block m_5 when the client asked for, and so the server may not replace the not-yet-deleted block m_5 by the block m'_5 that the client wants to insert, and can still pass the data integrity verification using the not-yet-deleted block m_5. In order to elude this situation, the index i is kept as "used" even if the block m_i has been deleted, and when a data block should be added between m_{i_1} and m_{i_2}, then the client can choose either an index equal to $\frac{i_1+i}{2}$ for $m_{\frac{i_1+i}{2}}$ or $\frac{i+i_2}{2}$ for $m_{\frac{i+i_2}{2}}$. In addition, we should explicit the insertion process when one or both of the involved indices are not natural numbers. More precisely, let us consider the blocks m_i such that $i \in \mathbb{N}$ and $m_{\frac{2i+1}{2}}$ that has been inserted earlier (and $\frac{2i+1}{2} \notin \mathbb{N}$). We wish to add a block m_l between the two aforementioned blocks: its index l should be set as the mean of the indices of the two blocks, i.e. $l = \frac{1}{2}(i + \frac{2i+1}{2}) = \frac{4i+1}{4}$. Such mean method should be kept in mind when inserting a new data block.

In our construction, we specify that the client deletes the data blocks and the corresponding verification metadata from his/her local storage once these elements are sent to the server. We also implicitly let the TPA know the indices of the data blocks that are currently stored on the server in order to challenge the server on a certain data amount of the stored data. To be quite clear, we let the client conserve a table of indices of the data blocks m_i kept on the server along with their version numbers vnb_i. Such table is then forwarded to the TPA. Refering to the aforementioned example with $m = (m_1, \cdots, m_{10})$, we suppose that the block m_5 has been deleted, the blocks $m_{\frac{4+5}{2}} = m_{\frac{9}{2}}$ and $m_{\frac{1}{2}(4+\frac{9}{2})} = m_{\frac{17}{4}}$ have been added, and the block m_6 has been modified twice. Let the table stored on the client's local storage be as follows:

Index i	Version Number vnb_i	Comments
1	1	-
$\cdots$	$\cdots$	$\cdots$
4	1	-
$17/4$	1	-
$9/2$	1	-
5	-	**DELETED**
6	3	-
7	1	-
$\cdots$	$\cdots$	$\cdots$
10	1	-

The above index hash table (IHT) is composed of several columns: one for the block index, one for the version number and one for some auxiliary comments. A column for random values can be added, if the verification metadata should be randomized to enhance the data privacy against the TPA. We stress that each record in the IHT is different from another one to ensure that the data blocks and their corresponding verification metadata cannot be forged. An example of an IHT-based PDP scheme can be found in [251].

The IHT framework gives better security level against the untrusted server while the security level against the TPA is lowered. In addition, we have to consider the random oracle model instead of the standard model. The IHT framework also leads into an increase of the complexity of the system; indeed, the communication and the computational costs grow for all the entities involved in the protocol. We recall that the TPA is required to help the client that does not have the necessary resources to audit the server efficiently and regularly. However, the task of the TPA should remain simple in verifying that the proof of data possession is consistent with the given challenge and that the updating proof is consistent with the requested data operation. Nevertheless, adding IHTs obliges the TPA to provide more local storage and more communication burden. Such additional costs make that the intervention of the TPA is not necessarly advantageous One may think that the TPA becomes a bigger threat since it keeps more sensitive information elements; however, note that the TPA already has access to the indices of the data blocks in the basic DPDP scheme DPDP given in Section 7.1.

Finally, we show later that such changes only slightly affect the efficiency and the practicality of the IHT-based protocol compared to the basic one presented in Section 7.1.

7.3.1 IHT-based Construction

The IHT-based DPDP scheme with public verifiability and data privacy construction is as follows:

- $\mathsf{KeyGen}(\lambda) \to (pk, sk)$. Let $\mathcal{G}$ be an algorithm that, on input the security parameter λ, generates the cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let g_1 and g_2 be two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Let the hash function $H : \mathbb{Q} \times \mathbb{N} \to \mathbb{G}_1$ be seen as a random oracle. Then, the client randomly chooses s elements $h_1, \cdots, h_s \in_R \mathbb{G}_1$. Moreover, s/he selects at random $\alpha \in_R \mathbb{Z}_p$ and sets his/her public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^{\alpha}, H)$ and his/her private key $sk = \alpha$.

- $\mathsf{TagGen}(pk, sk, m) \to T_m$. A document m is split into n blocks m_i, for $i \in [1, n]$. Each block m_i is then split into s sectors $m_{i,j} \in \mathbb{Z}_p$, for $j \in [1, s]$. We suppose that $|m| = b$ and $n = \lceil b/s \cdot \log(p) \rceil$. Therefore, the document m can be seen a $n \times s$ matrix with elements denoted as $m_{i,j}$. The client computes the verification metadata

$$T_{m_i} \quad = \quad (H(i, vnb_i) \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = H(i, vnb_i)^{-\alpha} \cdot \prod_{j=1}^{s} h_j^{-\alpha \cdot m_{i,j}}$$

  Yet, it sets $T_m = (T_{m_1}, \cdots, T_{m_n}) \in \mathbb{G}_1^n$.

  Then, the client stores all the data blocks m_i in an ordered collection $\mathbb{F}$ and the corresponding verification metadata T_{m_i} in an ordered collection $\mathbb{E}$. S/he forwards these two collections to the server and deletes them from his/her local storage.

  Moreover, the client creates an IHT to record the information necessary for the TPA to proceed on behalf of the client.

- $\mathsf{PerfOp}(pk, \mathbb{F}, \mathbb{E}, info = (\text{operation}, l, m_l, T_{m_l})) \to (\mathbb{F}', \mathbb{E}', \nu')$. After receiving the elements l, m_l and T_{m_l} from the client, the server prepares the updating proof as follows. It first selects at random $u_1, \cdots, u_s \in_R \mathbb{Z}_p$ and computes $U_1 = h_1^{u_1}, \cdots, U_s = h_s^{u_s}$. It also chooses at random $w_l \in_R \mathbb{Z}_p$ and sets $c_j = m_{l,j} \cdot w_l + u_j \in \mathbb{Z}_p$ and $C_j =$

$h_j^{c_j}$, for $j \in [1,s]$, and $d = T_{m_l}^{w_l}$. Finally, it returns $\nu' = (U_1, \cdots, U_s, C_1, \cdots, C_s, d, w_l) \in \mathbb{G}_1^{2s+1}$ to the TPA.

More precisely, for the operation:

- *insertion:* $(l, m_l, T_{m_l}) = (\frac{i_1+i_2}{2}, m_{\frac{i_1+i_2}{2}}, T_{m_{\frac{i_1+i_2}{2}}})$ and $vnb_l = vnb_{\frac{i_1+i_2}{2}} = 1$.
- *deletion:* $(l, m_l, T_{m_l}) = (i, \_, \_)$ and $vnb_l = vnb_i = \_$, meaning that the elements m_l, T_{m_l} and vnb_l are not required. Indeed, the server uses the block $m_l = m_i$ and the corresponding verification metadata $T_{m_l} = T_{m_i}$ that are kept on its storage to generate ν'.
- *modification:* $(l, m_l, T_{m_l}) = (i, m_i', T_{m_i'})$ and $vnb_l = vnb_i' = vnb_i + 1$.

- $\mathsf{CheckOp}(pk, \nu') \to true/false$. The TPA has to check whether the following equation holds:

$$e(d, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} U_j, g_2) \overset{?}{=} e(H(l, vnb_l)^{w_l}, g_2) \cdot e(\prod_{j=1}^{s} C_j, g_2) \qquad (7.5)$$

If the equation 7.5 holds, then the TPA returns $true$ to the client; otherwise, it returns $false$ to the client.

- $\mathsf{GenProof}(pk, F, chal, \Sigma) \to \nu$. The TPA on behalf of the client prepares a challenge $chal$ that is then sent to the server. First, it chooses a subset $I \subseteq (0, n+1) \cap \mathbb{Q}$, randomly chooses $|I|$ elements $v_i \in_R \mathbb{Z}_p$ and sets $chal = \{(i, v_i)\}_{i \in I}$.

  Second, after receiving the challenge $chal$ provided by the TPA, the server sets the ordered collection $F = \{m_i\}_{i \in I} \subset \mathbb{F}$ of blocks and an ordered collection $\Sigma = \{T_{m_i}\}_{i \in I} \subset \mathbb{E}$ which are the verification metadata corresponding to the blocks in F. It then selects at random $r_1, \cdots, r_s \in_R \mathbb{Z}_p$ and computes $R_1 = h_1^{r_1}, \cdots, R_s = h_s^{r_s}$. It also sets $b_j = \sum_{(i,v_i) \in chal} m_{i,j} \cdot v_i + r_j \in \mathbb{Z}_p$ and $B_j = h_j^{b_j}$, for $j \in [1,s]$, and $c = \prod_{(i,v_i) \in chal} T_{m_i}^{v_i}$. Finally, it returns $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c) \in \mathbb{G}_1^{2s+1}$ to the TPA.

- $\mathsf{CheckProof}(pk, chal, \nu) \to true/false$. The TPA has to check whether the following equation holds:

$$e(c, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} R_j, g_2) \overset{?}{=} e(\prod_{\substack{(i,v_i) \\ \in chal}} H(i, vnb_i)^{v_i}, g_2) \cdot e(\prod_{j=1}^{s} B_j, g_2) \qquad (7.6)$$

If the equation 7.6 holds, then the TPA returns $true$ to the client; otherwise, it returns $false$ to the client.

Correctness. We prove the correctness of the proof of data possession as follows:

$$\begin{aligned}
e(c, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} R_j, g_2) &= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} T_{m_i}^{v_i}, g_2^{\alpha}\right) \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} (H(i, vnb_i)^{-\alpha \cdot v_i} \cdot \prod_{j=1}^{s} h_j^{m_{i,j} \cdot (-\alpha) \cdot v_i}), g_2^{\alpha}\right) \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} H(i, vnb_i)^{v_i}, g_2\right)^{\alpha - \alpha} \cdot e(\prod_{j=1}^{s} h_j^{\sum_{\substack{(i,v_i)\\ \in chal}} m_{i,j} \cdot v_i}, g_2)^{\alpha - \alpha} \\
&\quad \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2) \\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} H(i, vnb_i)^{v_i}, g_2\right) \cdot e(\prod_{j=1}^{s} h_j^{\sum_{\substack{(i,v_i)\\ \in chal}} m_{i,j} \cdot v_i + r_j}, g_2) \\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} H(i, vnb_i)^{v_i}, g_2\right) \cdot e(\prod_{j=1}^{s} h_j^{b_j}, g_2) \\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} H(i, vnb_i)^{v_i}, g_2\right) \cdot e(\prod_{j=1}^{s} B_j, g_2)
\end{aligned}$$

We prove the correctness of the updating proof as follows:

$$\begin{aligned}
e(d, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} U_j, g_2) &= e(T_{m_l}^{w_l}, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(H(i, vnb_l)^{-\alpha \cdot w_l} \cdot \prod_{j=1}^{s} h_j^{m_{l,j} \cdot (-\alpha) \cdot w_l}, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(H(l, vnb_l)^{w_l}, g_2)^{\alpha - \alpha} \cdot e(\prod_{j=1}^{s} h_j^{m_{l,j} \cdot w_l}, g_2)^{\alpha - \alpha} \cdot e(\prod_{j=1}^{s} h_j^{u_j}, g_2) \\
&= e(H(l, vnb_l)^{w_l}, g_2) \cdot e(\prod_{j=1}^{s} h_j^{m_{l,j} \cdot w_l + u_j}, g_2) \\
&= e(H(l, vnb_l)^{w_l}, g_2) \cdot e(\prod_{j=1}^{s} h_j^{c_j}, g_2) \\
&= e(H(l, vnb_l)^{w_l}, g_2) \cdot e(\prod_{j=1}^{s} C_j, g_2)
\end{aligned}$$

7.3.2 Security Proofs

Proof of the Security against the Server

Theorem. Let $\mathscr{A}$ be a PPT adversary that has advantage ε against the security of the IHT-based DPDP scheme with public verifiability and data privacy. Suppose that $\mathscr{A}$ makes a total of $q_H > 0$ queries to H. Then, there is a challenger $\mathscr{B}$ that solves the CDH and DL problems $\mathbb{G}$ and $\mathbb{G}_1$ respectively, with advantage $\varepsilon' = O(\varepsilon)$.

For any PPT adversary $\mathscr{A}$ who wins the game, there is a challenger $\mathscr{B}$ that wants to break the CDH and DL problems $\mathbb{G}$ and $\mathbb{G}_1$ respectively, by interacting with $\mathscr{A}$ as follows.

Setup. $\mathscr{B}$ runs $\mathscr{G}$ on input λ to obtain the tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$. Then, it is given the CDH instance tuple (g, g^a, g^b) such that g is a generator of $\mathbb{G}$ and a, b are unknown exponents in $\mathbb{Z}_p$, chooses two exponents $x, y \in \mathbb{Z}_p$ and computes $g_1 = g^x$ and $g_2 = g^y$. It also sets $\mathbb{G}_1 =< g_1 >$ and $\mathbb{G}_2 =< g_2 >$. Note that $(g^a)^x = g_1^a$, $(g^b)^x = g_1^b$, $(g^a)^y = g_2^a$ and $(g^b)^y = g_2^b$. $\mathscr{B}$ chooses $\beta_j, \gamma_j \in_R \mathbb{Z}_p$ and sets $h_j = g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j}$ for $j \in [1, s]$. Let a hash function $H : \mathbb{Q} \times \mathbb{N} \to \mathbb{G}_1$ be controlled by $\mathscr{B}$ as follows. Upon receiving a query $(i_{l'}, vnb_{i_{l'}})$ to the random oracle H for some $l' \in [1, q_H]$:

- If $((i_{l'}, vnb_{i_{l'}}), \theta_{l'}, W_{l'})$ exists in L_H, return $W_{l'}$.
- Otherwise, choose $\beta_j, \gamma_j \in_R \mathbb{Z}_p$ and set $h_j = g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j}$ for $j \in [1, s]$. For each $i_{l'}$, choose $\theta_{l'} \in_R \mathbb{Z}_p$ at random and set

$$W_{l'} = \frac{g_1^{\theta_{l'}}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i_{l'},j}} (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i_{l'},j}}}$$

  for a given block $m_{i_{l'}} = (m_{i_{l'},1}, \cdots, m_{i_{l'},s})$. Put $((i_{l'}, vnb_{i_{l'}}), \theta_{l'}, W_{l'})$ in L_H and return $W_{l'}$ as answer.

$\mathscr{B}$ sets the public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^a, H)$ and forwards it to $\mathscr{A}$. It keeps g_1^a, g_1^b, g_2^b secret. The private key sk is implicitly set as equal to a.

Adaptive Queries. $\mathscr{A}$ has first access to the verification metadata generation oracle $\mathscr{O}_{TG}$ as follows. It first adaptively selects blocks m_i, for $i \in [1, n]$. $\mathscr{B}$ splits each block m_i into s sectors $m_{i,j}$. Then, it computes $T_{m_i} = (W \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (W \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-a}$, for $i \in [1, n]$, such that if $((i, vnb_i), \theta, W)$ exists in L_H, then the value W is used to compute T_{m_i}. Otherwise, an element $\theta \in_R \mathbb{Z}_p$ is chosen at random, the value $W = \frac{g_1^{\theta}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i,j}} (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}}}$ is computed for an element $h_j = g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j}$, $((i, vnb_i), \theta, W)$ is put in L_H and W is used for the generation of T_{m_i}.

Note that we have

$$\begin{aligned}
\prod_{j=1}^{s} h_j^{m_{i,j}} \cdot H(i, vnb_i) &= (\prod_{j=1}^{s} h_j^{m_{i,j}}) \cdot \frac{g_1^{\theta}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i,j}} \cdot (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}}} \\
&= \frac{g_1^{\sum_{j=1}^{s} \beta_j m_{i,j}} (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}} \cdot g_1^{\theta}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i,j}} \cdot (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}}} \\
&= g_1^{\theta}
\end{aligned}$$

and so, $T_{m_i} = (H(i,vnb_i)\cdot\prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (H(i,vnb_i)\cdot\prod_{j=1}^{s} h_j^{m_{i,j}})^{-a} = (g_1^a)^{-\theta}$.

$\mathscr{B}$ gives the blocks and the verification metadata to $\mathscr{A}$. The latter sets an ordered collection $\mathbb{F} = \{m_1,\cdots,m_n\}$ of blocks and an ordered collection $\mathbb{E} = \{T_{m_1},\cdots,T_{m_n}\}$ which are the verification metadata corresponding to the blocks in $\mathbb{F}$.

$\mathscr{A}$ has also access to the data operation performance oracle $\mathscr{O}_{DOP}$ as follows. Repeatedly, $\mathscr{A}$ selects a block m_l and the corresponding element $info_l$ and forwards them to $\mathscr{B}$. Here, l denotes the rank where $\mathscr{A}$ wants the data operation to be performed. More precisely, l is equal to $\frac{i_1+i_2}{2}$ for an insertion and to i for a deletion or a modification. Moreover, $m_l = \_$ in the case of a deletion, since only the rank is needed to perform this kind of operation. The version number vnb_l increases by one in the case of a modification. Then, $\mathscr{A}$ outputs a new ordered collection $\mathbb{F}'$ (containing the updated version of the block m_l), a new ordered collection $\mathbb{E}'$ (containing the updated version of the verification metadata T_{m_l}) and a corresponding updating proof $\nu' = (U_1,\cdots,U_s,C_1,\cdots,C_s,d,w_l)$, such that w_l is randomly chosen from $\mathbb{Z}_p$, $d = T_{m_l}^{w_l}$, and for $j \in [1,s]$, u_j is randomly chosen from $\mathbb{Z}_p$, $U_j = h_j^{u_j}$, $c_j = m_{l,j}\cdot w_l + u_j$ and $C_j = h_j^{c_j}$. $\mathscr{B}$ runs the algorithm CheckOp on the value ν' and sends the answer to $\mathscr{A}$. If the answer is $false$, then the challenger aborts; otherwise, it proceeds.

Challenge. $\mathscr{A}$ selects some blocks m_i^* and the corresponding elements $info_i^*$, for $i \in \mathscr{I} \subseteq (0,n+1)\cap\mathbb{Q}$, and forwards them to the challenger who checks the data operations. In particular, the first $info_i^*$ indicates a full re-write.

$\mathscr{B}$ chooses a subset $I \subseteq \mathscr{I}$, randomly chooses $|I|$ elements $v_i \in_R \mathbb{Z}_p$ and sets $chal = \{(i,v_i)\}_{i\in I}$. It forwards $chal$ as a challenge to $\mathscr{A}$.

Forgery. Upon receiving the challenge $chal$, the resulting proof of data possession on the correct stored document m should be $\nu = (R_1,\cdots,R_s,B_1,\cdots,B_s,c)$ and pass the equation 7.6. However, $\mathscr{A}$ generates a proof of data possession on an incorrect stored document $\tilde{m}$ as $\tilde{\nu} = (\tilde{R}_1,\cdots,\tilde{R}_s,\tilde{B}_1,\cdots,\tilde{B}_s,\tilde{c})$, such that $\tilde{r}_j$ is randomly chosen from $\mathbb{Z}_p$, $\tilde{R}_j = h_j^{\tilde{r}_j}$, $\tilde{b}_j = \sum_{(i,v_i)\in chal}\tilde{m}_{i,j}\cdot v_i + \tilde{r}_j$ and $\tilde{B}_j = h_j^{\tilde{b}_j}$, for $j \in [1,s]$, along with $\tilde{c} = \prod_{(i,v_i)\in chal} T_{\tilde{m}_i}^{v_i}$. Finally, it returns $\tilde{\nu}$ to $\mathscr{B}$. If the proof of data possession still pass the verification, then $\mathscr{A}$ wins. Otherwise, it fails.

Analysis. We define $\Delta r_j = \tilde{r}_j - r_j$, $\Delta b_j = \tilde{b}_j - b_j = \sum_{(i,v_i)\in chal}(\tilde{m}_{i,j} - m_{i,j})v_i + \Delta r_j$ and $\Delta\tau_j = \sum_{(i,v_i)\in chal}(\tilde{m}_{i,j} - m_{i,j})v_i$, for $j \in [1,s]$. Note that r_j and b_j are the elements of a honest proof of data possession ν such that $r_j \in_R \mathbb{Z}_p$ and $b_j = \sum_{(i,v_i)\in chal} m_{i,j}\cdot v_i + r_j$ where $m_{i,j}$ are the actual sectors (not the ones that the adversary claims to have).

We prove that if the adversary can win the game, then solutions to the CDH and DL problems are found, which contradicts the assumption that the CDH and DL problems are hard in $\mathbb{G}$ and $\mathbb{G}_1$ respectively. Let assume that the adversary (playing the role of the server) wins the game. We recall that if $\mathscr{A}$ wins then $\mathscr{B}$ can extract the actual blocks $\{m_i\}_{(i,v_i)\in chal}$ in polynomially-many interactions with $\mathscr{A}$. Without loss of generality, suppose that $chal = \{(i,v_i)\}$, meaning that the challenge contains only one block.

First case ($\tilde{c} \neq c$): According to the equation 7.6, we have

$$\begin{aligned} e(\frac{\tilde{c}}{c}, g_2) &= e\left(\frac{T_{\tilde{m}_i}}{T_{m_i}}, g_2\right)^{v_i} = e\left(\frac{(\prod_{j=1}^{s} h_j^{\tilde{\tau}_j})^{-a}}{(\prod_{j=1}^{s} h_j^{\tau_j})^{-a}}, g_2\right) \\ &= e(\prod_{j=1}^{s} h_j^{\Delta\tau_j}, g_2^{-a}) \\ &= e(\prod_{j=1}^{s} (g_1^{\beta_j} \cdot (g_1^{b})^{\gamma_j})^{\Delta\tau_j}, g_2^{-a}) \\ &= e(g_1, g_2)^{-a\sum_{j=1}^{s} \beta_j \Delta\tau_j} \cdot e(g_1, g_2)^{-ab\sum_{j=1}^{s} \gamma_j \Delta\tau_j} \end{aligned}$$

and so, we get that

$$e(\frac{\tilde{c}}{c} \cdot (g_1^{a})^{\sum_{j=1}^{s} \beta_j \Delta\tau_j}, g_2) = e(g_1^{b}, g_2^{-a})^{\sum_{j=1}^{s} \gamma_j \Delta\tau_j}$$

meaning that we have found the solution to the CDH problem, that is

$$(g_1^{b})^{a} = (g^{x})^{ab} = (\frac{\tilde{c}}{c} \cdot (g_1^{a})^{\sum_{j=1}^{s} \beta_j \Delta\tau_j})^{\frac{-1}{\sum_{j=1}^{s} \gamma_j \Delta\tau_j}}$$

unless evaluating the exponent causes a divide-by-zero. Nevertheless, we notice that not all of the $\Delta\tau_j$ can be zero (indeed, if $\tau_j = m_{i,j} \cdot v_i = \tilde{\tau}_j = \tilde{m}_{i,j} \cdot v_i$ for each $j \in [1, s]$, then $c = \tilde{c}$ which contradicts the hypothesis), and the values γ_j are information theoretically hidden from $\mathscr{A}$ (Pedersen commitments), so the denominator is zero only with probability $1/p$, which is negligible. Finally, since $\mathscr{B}$ knows the exponent x such that $g_1 = g^{x}$, it can directly compute

$$((\frac{\tilde{c}}{c} \cdot (g_1^{a})^{\sum_{j=1}^{s} \beta_j \Delta\tau_j})^{\frac{-1}{\sum_{j=1}^{s} \gamma_j \Delta\tau_j}})^{\frac{1}{x}}$$

and obtains g^{ab}. Thus, if $\mathscr{A}$ wins the game, then a solution to the CDH problem can be found with probability equal to $1 - 1/p$.

Second Case ($\tilde{c} = c$): According to the equation 7.6, we have

$$e(\tilde{c}, g_2^{a}) = e(H(i, vnb_i)^{v_i}, g_2) \cdot e(\prod_{j=1}^{s} \tilde{B}_j, g_2) \cdot e(\prod_{j=1}^{s} \tilde{R}_j, g_2)^{-1}.$$

Since the proof $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c)$ is a correct one, we also have

$$e(c, g_2^{a}) = e(H(i, vnb_i)^{v_i}, g_2) \cdot e(\prod_{j=1}^{s} B_j, g_2) \cdot e(\prod_{j=1}^{s} R_j, g_2)^{-1}.$$

We recall that $chal = \{(i, v_i)\}$. From the previous analysis step, we know that $\tilde{c} = c$. Therefore, we get that $\prod_{j=1}^{s} \tilde{B}_j \cdot (\prod_{j=1}^{s} \tilde{R}_j)^{-1} = \prod_{j=1}^{s} B_j \cdot (\prod_{j=1}^{s} R_j)^{-1}$. We can write as $\prod_{j=1}^{s} h_j^{\tilde{b}_j - \tilde{r}_j} = \prod_{j=1}^{s} h_j^{b_j - r_j}$ or even as $\prod_{j=1}^{s} h_j^{\Delta b_j - \Delta r_j} = \prod_{j=1}^{s} h_j^{\Delta\tau_j} = 1$. For two elements $g_1, h \in \mathbb{G}_1$, there exists $\xi \in \mathbb{Z}_p$ such that $h = g_1^{\xi}$ since $\mathbb{G}_1$ is a cyclic group. Without loss of generality, given $g_1, h \in \mathbb{G}_1$, each h_j could randomly and correctly be generated by computing $h_j = g_1^{y_j} \cdot h^{z_j} \in \mathbb{G}_1$ such that y_j and z_j

are random values in $\mathbb{Z}_p$. Then, we have $1 = \prod_{j=1}^{s} h_j^{\Delta\tau_j} = \prod_{j=1}^{s}(g_1^{y_j} \cdot h^{z_j})^{\Delta\tau_j} = g_1^{\sum_{j=1}^{s} y_j \cdot \Delta\tau_j} \cdot h^{\sum_{j=1}^{s} z_j \cdot \Delta\tau_j}$. Clearly, we can find a solution to the DL problem. More specifically, given $g_1, h = g_1^{\xi} \in \mathbb{G}_1$, we can compute $h = g_1^{\frac{\sum_{j=1}^{s} y_j \cdot \Delta\tau_j}{\sum_{j=1}^{s} z_j \cdot \Delta\tau_j}} = g_1^{\xi}$ unless the denominator is zero. However, not all of the $\Delta\tau_j$ can be zero and the values z_j are information theoretically hidden from $\mathscr{A}$, so the denominator is only zero with probability $1/p$, which is negligible. Thus, if $\mathscr{A}$ wins the game, then a solution to the DL problem can be found with probability equal to $1 - \frac{1}{p}$.

Therefore, for $\mathscr{A}$, it is computationally infeasible to win the game and generate an incorrect proof of data possession which can pass the verification.

The simulation of the verification metadata generation oracle $\mathscr{O}_{TG}$ is perfect. The simulation of the data operation performance oracle $\mathscr{O}_{DOP}$ is almost perfect unless $\mathscr{B}$ aborts. This happens when the data operation was not correclty performed. As previously, we can prove that if $\mathscr{A}$ can pass the updating proof, then solutions to the CDH and DL problems are found. Following the above analysis and according to Equation 7.5, if $\mathscr{A}$ generates an incorrect updating proof which can pass the verification, then solutions to the CDH and DL problems can be found with probabilities both equal to $1 - \frac{1}{p}$. Therefore, for $\mathscr{A}$, it is computationally infeasible to generate an incorrect updating proof which can pass the verification. The proof is completed.

Proof of the Data Privacy against the TPA

Theorem. Let $\mathscr{A}$ be a PPT adversary that has advantage ε against the weak data privacy property of the IHT-based DPDP scheme with public verifiability and data privacy. Suppose that $\mathscr{A}$ makes a total of $q_H > 0$ queries to H. Then, there is a challenger $\mathscr{B}$ that solves the CDH problem in $\mathbb{G}$ with advantage $\varepsilon' = O(\varepsilon)$.

For any PPT adversary $\mathscr{A}$ who wins the game, there is a challenger $\mathscr{B}$ that wants to break the CDH assumption in $\mathbb{G}$ by interacting with $\mathscr{A}$ as follows.

Setup. $\mathscr{B}$ runs $\mathsf{GroupGen}(\lambda) \rightarrow (p, \mathbb{G}, \mathbb{G}_T, e)$. Then, it is given the CDH instance tuple (g, g^a, g^b) such that g is a generator of $\mathbb{G}$ and a, b are unknown exponents in $\mathbb{Z}_p$, chooses two exponents $x, y \in \mathbb{Z}_p$ and computes $g_1 = g^x$ and $g_2 = g^y$. It also sets $\mathbb{G}_1 =< g_1 >$ and $\mathbb{G}_2 =< g_2 >$. Note that $(g^a)^x = g_1^a$, $(g^b)^x = g_1^b$, $(g^a)^y = g_2^a$ and $(g^b)^y = g_2^b$. $\mathscr{B}$ chooses $\beta_j, \gamma_j \in_R \mathbb{Z}_p$ and sets $h_j = g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j}$ for $j \in [1, s]$. Let a hash function $H : \mathbb{Q} \times \mathbb{N} \rightarrow \mathbb{G}_1$ be controlled by $\mathscr{B}$ as follows. Upon receiving a query (i_l, vnb_{i_l}) to the random oracle H for some $l \in [1, q_H]$:

- If $((i_l, vnb_{i_l}), \theta_l, W_l)$ exists in L_H, return W_l.
- Otherwise, choose $\beta_j, \gamma_j \in_R \mathbb{Z}_p$ and set $h_j = g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j}$ for $j \in [1, s]$. For earch i_l, choose $\theta_l \in_R \mathbb{Z}_p$ at random and set

$$W_l = \frac{g_1^{\theta_l}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i_l,j}} (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i_l,j}}}$$

for a given block $m_{i_l} = (m_{i_l,1}, \cdots, m_{i_l,s})$. Put $((i_l, vnb_{i_l}), \theta_l, W_l)$ in L_H and return W_l as answer.

$\mathscr{B}$ sets the public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^a, H)$ and forwards it to $\mathscr{A}$. It keeps g_1^a, g_1^b, g_2^b secret. The private key sk is implicitly set as equal to a.

Queries. $\mathscr{A}$ makes queries to the verification metadata generation oracle as follows. It first adaptively selects blocks m_i, for $i \in [1,n]$. $\mathscr{B}$ splits each block m_i into s sectors $m_{i,j}$. Then, it computes $T_{m_i} = (W \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (W \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-a}$, for $i \in [1,n]$, such that if $((i, vnb_i), \theta, W)$ exists in L_H, then the value W is used to compute T_{m_i}. Otherwise, an element $\theta \in_R \mathbb{Z}_p$ is chosen at random, $W = \frac{g_1^{\theta}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i,j}} (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}}}$ is computed for an element $h_j = g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j}$, $((i, vnb_i), \theta, W)$ is put in L_H and W is used for the generation of T_{m_i}.

Note that we have

$$\begin{aligned}
\prod_{j=1}^{s} h_j^{m_{i,j}} \cdot H(i, vnb_i) &= (\prod_{j=1}^{s} h_j^{m_{i,j}}) \cdot \frac{g_1^{\theta}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i,j}} \cdot (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}}} \\
&= \frac{g_1^{\sum_{j=1}^{s} \beta_j m_{i,j}} (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}} \cdot g_1^{\theta}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i,j}} \cdot (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}}} \\
&= g_1^{\theta}
\end{aligned}$$

and so, $T_{m_i} = (H(i, vnb_i) \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (H(i, vnb_i) \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-a} = (g_1^a)^{-\theta}$.

$\mathscr{B}$ gives the blocks and the verification metadata to $\mathscr{A}$. The latter sets an ordered collection $\mathbb{F} = \{m_1, \cdots, m_n\}$ of blocks and an ordered collection $\mathbb{E} = \{T_{m_1}, \cdots, T_{m_n}\}$ which are the verification metadata corresponding to the blocks in $\mathbb{F}$.

Challenge. The adversary submits a challenge $chal = \{(i, v_i)\}_{i \in I}$. Without loss of generality, we suppose there is only one element i in I. As a shorthand, we write $chal = \{(i, v_i)\}$. Moreover, $\mathscr{A}$ gives an ordered collection $F = \{\tilde{m}_i\} \cap \mathbb{F} = \emptyset$ of the data blocks determined by $chal$, and an ordered collection $\Sigma = \{T_{\tilde{m}_i}\} \cap \mathbb{E} = \emptyset$ of the corresponding verification metadata. Note that there are only one element $\tilde{m}_i$ in F and one element $T_{\tilde{m}_i}$ in Σ.

Generation of the Proof. Upon receiving the challenge $chal = \{(i, v_i)\}$, the collection $F = \{\tilde{m}_i\}$ and the collection $\Sigma = \{T_{\tilde{m}_i}\}$, the challenger generates the proof of data possession $\tilde{v} = (\tilde{R}_1, \cdots, \tilde{R}_s, \tilde{B}_1, \cdots, \tilde{B}_s, \tilde{c})$, such that $\tilde{r}_j$ is randomly chosen from $\mathbb{Z}_p$, $\tilde{R}_j = h_j^{\tilde{r}_j}$, $\tilde{b}_j = \sum_{(i,v_i) \in chal} \tilde{m}_{i,j} \cdot v_i + \tilde{r}_j$ and $\tilde{B}_j = h_j^{\tilde{b}_j}$, for $j \in [1,s]$. It also sets $\tilde{c} = \prod_{(i,v_i) \in chal} T_{\tilde{m}_i}^{v_i}$.

Finally, the challenger runs CheckProof on $\tilde{v}$. If the proof of data possession still pass the verification, then $\mathscr{A}$ wins. Otherwise, it fails.

Analysis. Given an actual block m_i and the corresponding verification metadata T_{m_i}, let $v = (R_1, \cdots, R_s, B_1, \cdots, B_s, c)$ be a honest proof of data possession that pass the verification.

We define $\Delta r_j = \tilde{r}_j - r_j$, $\Delta b_j = \tilde{b}_j - b_j = \sum_{(i,v_i) \in chal} (\tilde{m}_{i,j} - m_{i,j}) v_i + \Delta r_j$ and $\Delta \tau_j = \sum_{(i,v_i) \in chal} (\tilde{m}_{i,j} - m_{i,j}) v_i = (\tilde{m}_{i,j} - m_{i,j}) v_i$, for $j \in [1,s]$. Note that r_j and b_j are the elements of a honest proof of data possession v such that $r_j \in_R \mathbb{Z}_p$ and $b_j = \sum_{(i,v_i) \in chal} m_{i,j} \cdot$

$v_i + r_j = m_{i,j} \cdot v_i + r_j$ where $m_{i,j}$ are the actual sectors (not the ones that the adversary claims to have).

We prove that if the adversary can win the game, then a solution to the CDH problem is found, which contradicts the assumption that the CDH problem is hard in $\mathbb{G}$. Let assume that the adversary (playing the role of the TPA) wins the game.

According to the equation 7.6 and since $T_{\tilde{m}} \neq T_m$, giving that $\tilde{c} \neq c$, We have

$$\begin{aligned} e(\frac{\tilde{c}}{c}, g_2) &= e\left(\frac{T_{\tilde{m}_i}}{T_{m_i}}, g_2\right)^{v_i} = e\left(\frac{(\prod_{j=1}^{s} h_j^{\tilde{\tau}_j})^{-a}}{(\prod_{j=1}^{s} h_j^{\tau_j})^{-a}}, g_2\right) \\ &= e(\prod_{j=1}^{s} h_j^{\Delta\tau_j}, g_2^{-a}) \\ &= e(\prod_{j=1}^{s} (g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j})^{\Delta\tau_j}, g_2^{-a}) \\ &= e(g_1, g_2)^{-a\sum_{j=1}^{s}\beta_j\Delta\tau_j} \cdot e(g_1, g_2)^{-ab\sum_{j=1}^{s}\gamma_j\Delta\tau_j} \end{aligned}$$

and so, we gets

$$e(\frac{\tilde{c}}{c} \cdot (g_1^a)^{\sum_{j=1}^{s}\beta_j\Delta\tau_j}, g_2) = e(g_1^b, g_2^{-a})^{\sum_{j=1}^{s}\gamma_j\Delta\tau_j}$$

meaning that we have found the solution to the CDH problem, that is

$$(g_1^b)^a = (g^x)^{ab} = (\frac{\tilde{c}}{c} \cdot (g_1^a)^{\sum_{j=1}^{s}\beta_j\Delta\tau_j})^{\frac{-1}{\sum_{j=1}^{s}\gamma_j\Delta\tau_j}}$$

unless evaluating the exponent causes a divide-by-zero. Nevertheless, we notice that not all of the $\Delta\tau_j$ can be zero (indeed, if $\tau_j = m_{i,j} \cdot v_i = \tilde{\tau}_j = \tilde{m}_{i,j} \cdot v_i$ for each $j \in [1, s]$, then $c = \tilde{c}$ which contradicts the hypothesis), and the values γ_j are information theoretically hidden from $\mathscr{A}$ (Pedersen commitments), so the denominator is zero only with probability $1/p$, which is negligible. Finally, since $\mathscr{B}$ knows the exponent x such that $g_1 = g^x$, it can directly compute

$$((\frac{\tilde{c}}{c} \cdot (g_1^a)^{\sum_{j=1}^{s}\beta_j\Delta\tau_j})^{\frac{-1}{\sum_{j=1}^{s}\gamma_j\Delta\tau_j}})^{\frac{1}{x}}$$

and obtains g^{ab}. Thus, if $\mathscr{A}$ wins the game, then a solution to the CDH problem can be found with probability equal to $1 - 1/p$.

7.3.3 Performance of the IHT-based DPDP Scheme

We compare the IHT-based DPDP scheme with the basic DPDP scheme DPDP proposed in Section 7.1. First, the client and the TPA obviously have to store more information by keeping the IHT. Nevertheless, we stress that in any case, the client and the TPA should maintain an index list. Indeed, they need some information about the stored data in order to select some data blocks to be challenged. We recall that the challenge consists of pairs (index, random element). By appending an integer and sometimes an auxiliary comment (only in case of deletions) to each index, the extra burden does not seem excessive. Therefore, such table does slightly affect the client's local storage as well as the TPA's local storage. The communication between the client and the TPA rather increases since the client should send more elements to the TPA in order to keep the table updated.

Second, the client has to perform extra computation when generating the verification metatdata: for each data block m_i, s/he has to compute $H(i, vnb_i)$. However, the overhead of the communication between the client and the server does not inscrease.

Third, the TPA needs to compute an extra pairing $e(H(i, vnb_i)^{w_i}, g_2)$ in order to check that the server correclty performed a data operation requested by the client. The TPA also has to compute $|I|$ multiplications in $\mathbb{G}_1$ and one extra pairing when checking the proof of data possession: for each challenge $chal = \{(i, v_i)\}_{i \in I}$, it calculates $\prod_{(i,vi) \in chal} H(i, vnb_i)$ as well as the pairing $e(\prod_{(i,vi) \in chal} H(i, vnb_i)^{v_i}, g_2)$. This gives a constant total of four pairings in order to verify the data integrity instead of three as in the basic scheme given in Section 7.1, that is not a big loss in term of efficiency and practicality.

Finally, apart the storage of a light table and the computation of an extra pairing by the TPA for the verification of both the updating proof and the proof of data possession, the new construction for the DPDP protocol with public verifiability and data privacy is still practical by adopting asymmetric pairings to gain efficiency and by still reducing the group exponentiation and pairing operations. In addition, this scheme still allows the TPA on behalf of the client to request the server for a proof of data possession on as many data blocks as possible at no extra cost, as in the basic cheme given in Section 7.1.

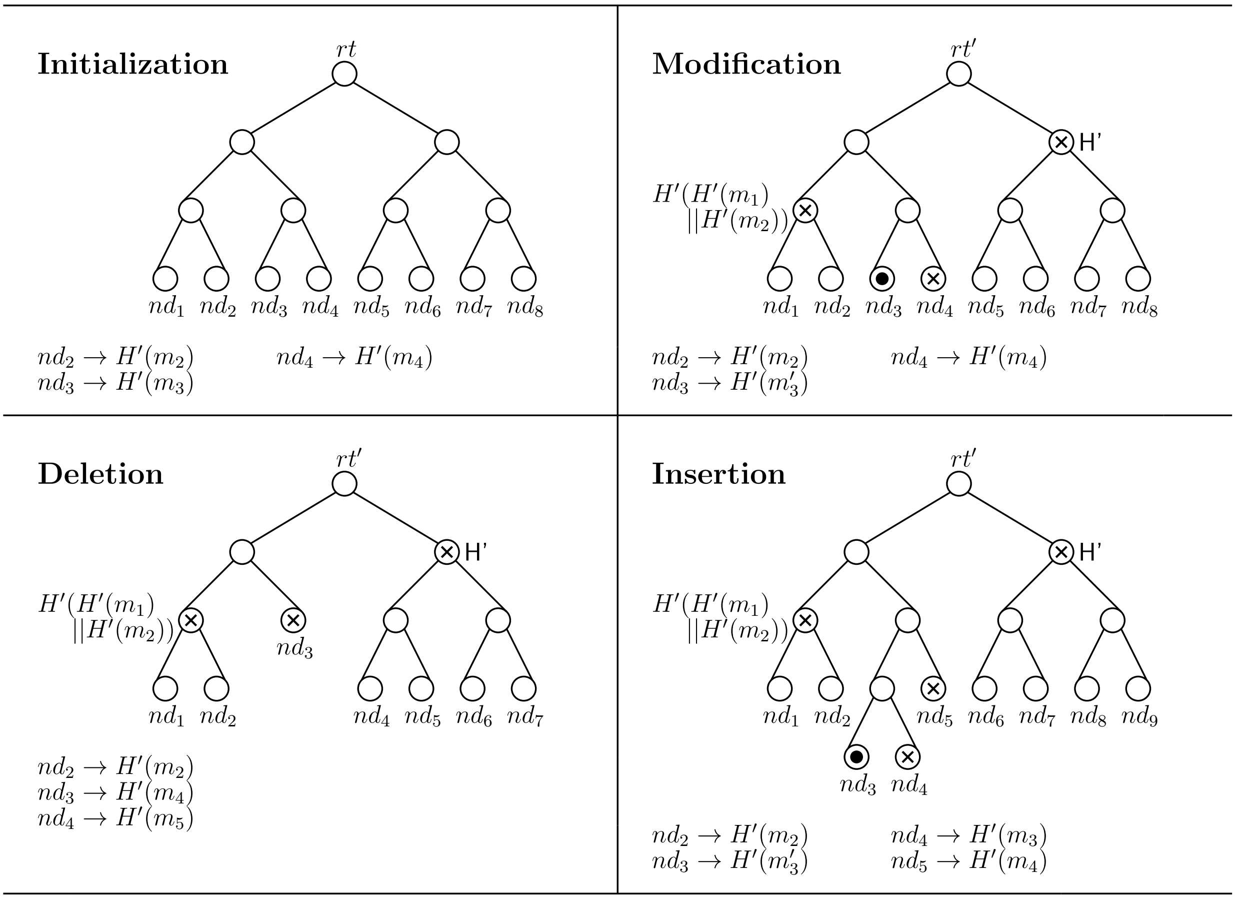
7.4 MHT-Based Dynamic Provable Data Possession with Public Verifiability and Data Privacy

The second solution to avoid the replace and replay attacks and the attacks against data privacy is to implement a Merkle hash tree (MHT) for each document such that the data blocks are ordered. The MHT [157] is similar to a binary tree in the way that each node nd has at most two children. Following the update algorithm, each internal (non-leaf) node has always two children. The construction of a MHT is as follows. For leaf node nd_i based on the data block m_i, the assigned value is equal to $H'(m_i)$, where the hash function $H' : \{0,1\}^* \to \mathbb{G}_1$ is seen as a random oracle. Note that the hash values are affected to leaf nodes in the increasing order of the blocks, i.e. nd_1 corresponds to the hash of the first block m_1, nd_2 corresponds to the hash of the first block m_2, and so on. A parent node of nd_i and nd_{i+1} has a value computed as $H'(H'(m_i)||H'(m_{i+1}))$, where $||$ is the concatenation sign (for an odd index i). The auxiliary authentication information (AAI) Ω_i of a leaf node nd_i for the data block m_i is a set of hash values chosen from its upper levels, so that the root value rt can be computed through (m_i, Ω_i).

When the client desires to add, remove or change a data block on his/her stored data, s/he has first to inform the server of such wish. The client's request R contains the type of operation that has to be performed (insertion, deletion or modification) as well as the position where the operation will be done. Using such information, the server is able to select the appopriate elements from the current version of the MHT and set the AAI Ω_i as the tuple of all these elements. More precisely, the elements are the hash values assigned on the nodes of the MHT: the elements can be found either on the leaves or the internal nodes, at different level, in function of what needs the client to create the updated version of the MHT according to the data operation.

Let $m = (m_1, \cdots, m_8)$ be a document stored on the server. In Figure 7.3, we highlight which leaves and internal nodes are required in order to insert a data block m'_3 after the data block m_2, to delete the block m_3 and to modify the block m_3 by replacing it with

m'_3. Each hash value $H'(m_i)$ is assigned to a leaf node nd_i, for $i \in [1,n]$, in the increasing order. The blocks that sustain the operations are alloted to leaf nodes with a disk inside. The hash values that should be included into Ω_3 are affected to leaves or internal nodes with a cross inside. Such values will allow the client to compute the new version of the MHT, including the new root rt' that will be then signed.



N.B.: $\mathsf{H}' = H'(H'(H'(m_5)||H'(m_6))||H'(H'(m_7)||H'(m_8)))$

Figure 7.3: MHTs for the data document $m = (m_1, \cdots, m_8)$ at the initialization phase, at the modification phase (changing the block m_3 into m'_3), at the deletion phase (removing the block m_3), and at the insertion phase (adding the block m'_3 after the block m_2).

The AAI Ω_i is also required by the TPA to check the proof of data possession. More precisely, in addition to the proof of data possession ν, the server also forwards the values rt_{server}, $H'(m_i)$ and Ω_i for $i \in I$, according to the blocks indicated by $chal$. With such elements, the TPA is able to construct the current MHT and verifies that the root that it obtained is equal to rt_{server}. If such verification is successful, then the TPA proceeds to validate the proof of data possession.

We recall that the client generates the verification metadata for each block of a document $m = (m_1, \cdots, m_n)$, and sends both the data blocks and the corresponding verification metadata to the server. Now, the client has to also construct a MHT for such a document: given a hash function $H' : \{0,1\}^* \to \mathbb{G}_1$, s/he computes $H'(m_i)$ for $i \in [1,n]$, and assigns the values $H'(m_i)$ to each leaf of the MHT in the increasing order. Then, s/he calculates the hash values of the internal nodes until computing the root rt of the MHT, following the construction definition of such a tree. Given a digital signature scheme $\mathsf{SS} = (\mathsf{SS.KeyGen}, \mathsf{SS.Sign}, \mathsf{SS.Verify})$, s/he signs the root rt using the signing private key $\mathsf{SS}.sk \leftarrow \mathsf{SS.KeyGen}(\lambda)$ and obtains the signature $\sigma_{rt} \leftarrow \mathsf{SS.Sign}(\mathsf{SS}.sk, rt)$. Finally, s/he sends H' and σ_{rt} to the server. The client also computes the verifying public key $\mathsf{SS}.pk \leftarrow \mathsf{SS.KeyGen}(\lambda)$ and shares it with the TPA.

Upon receiving the ordered collection $\mathbb{F} = \{m_i\}_{i \in [1,n]}$ of the data blocks, the ordered collection $\mathbb{E} = \{T_{m_i}\}_{i \in [1,n]}$ of the corresponding verification metadata, the hash function H' and the signature σ_{rt}, the server first constructs the MHT such that each hash value $H'(m_i)$ is assigned to each leaf of the MHT in the increasing order. It then obtains a root rt_{server} and sends it to the client. The latter then runs $\mathsf{SS.Verify}(\mathsf{SS}.pk, \sigma_{rt}, rt_{server})$ and gets an answer either equal to $true$ or $false$. If the answer is equal to $true$, then the client knows that the server correctly downloaded his/her documents (the roots of their respective MHTs are identical), and proceeds. Otherwise, then the client knows that the server has not obtained the same MHT thus does not correctly stores the data, and aborts.

Data Operations. When the client wants to add a block $m_{i'}$ after the block m_i, remove m_i or modify m_i by replacing it with m'_i, for $i \in [1,n]$, s/he first sends a request to the server containing information such as the type of operation to be performed and its location (index of the block).

Upon each request, the server needs to return the AAI Ω_i of the block to be updated (in order to reconstruct the current MHT and build the new one after the data update) and the last signed root value σ_{rt} provided by the client. The latter is so able to authenticate the AAI Ω_i by verifying the given signed root value σ_{rt} with the root rt of the reconstructed MHT. If the AAI are successfully authenticated, then the client proceeds; otherwise, s/he aborts.

Thereafter, in order to get the new MHT, the client computes the value $H'(m_{i'})$ for insertion or $H'(m'_i)$ for modification. Note that s/he does not need to compute $H(m_i)$ for deletion. In the new MHT, for the operation:

- *insertion:* the block $m_{i'}$ takes the position of and becomes the block m_{i+1}, the block m_{i+1} takes the position of and becomes the block m_{i+2} and so on, until the block m_n that has a new position and becomes the block m_{n+1}.
- *deletion:* the block m_{i+1} takes the position of and becomes the deleted block m_i, the block m_{i+2} takes the position of and becomes the block m_{i+1} and so on, until the block m_n that takes the position of and becomes the block m_{n-1}.
- *modification:* m'_i simply takes the position of m_i.

The client then signs the updated root rt' obtained in the new MHT by running $\mathsf{SS.Sign}$, gets the signature $\sigma_{rt'}$ and forwards it to the server, in addition to $(m_{i'}, T_{m_{i'}})$ for insertion and $(m'_i, T_{m'_i})$ for modification. The server builds the MHT following the client's operation request and gives the resulting root rt'_{server} to the client. Yet, the client runs $\mathsf{SS.Verify}(\mathsf{SS}.pk, \sigma_{rt'}, rt'_{server})$ to get an answer either equal to $true$ or $false$. If the answer is equal to $true$, then the client knows that the server has correctly updated the data (the roots of their respective MHTs are identical), and proceeds (in particular, s/he can delete from his/her local storage $(m_{i'}, T_{m_{i'}}, \Omega_i, \sigma_{rt'})$ for insertion, $(\Omega_i, \sigma_{rt'})$ for deletion or $(m'_i, T_{m'_i}, \Omega_i, \sigma_{rt'})$ for modification). Otherwise, then the client knows that the server has not obtained the same MHT thus has not correctly performed the operation on the data, and thus aborts.

Note that the client may ask the TPA to challenge the server after each update process in order to check the server's behavior.

7.4.1 MHT-based Construction

Let $\mathsf{DPDP} = (\mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{PerfOp}, \mathsf{CheckOp}, \mathsf{GenProof}, \mathsf{CheckProof})$ be a DPDP scheme with public verifiability and data privacy such as the basic scheme given in Section 7.1. Let $\mathsf{SS} = (\mathsf{SS.KeyGen}, \mathsf{SS.Sign}, \mathsf{SS.Verify})$ be a strongly unforgeable digital signature scheme. The MHT-based DPDP scheme with public verifiability and data privacy construction is as follows:

- $\mathsf{MHT.KeyGen}(\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$. The client first runs $\mathsf{KeyGen}(\lambda) \rightarrow (pk, sk)$ and $\mathsf{SS.KeyGen}(\lambda) \rightarrow (\mathsf{SS}.pk, \mathsf{SS}.sk)$. More precisely, let $\mathscr{G}$ be an algorithm that, on input the security parameter λ, generates the cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Let g_1 and g_2 be two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Then, s elements $h_1, \cdots, h_s \in_R \mathbb{G}_1$, as well as $\alpha \in_R \mathbb{Z}_p$ are choosen randomly.

  The client sets his/her public key $\mathsf{pk} = (pk, \mathsf{SS}.pk) = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^{\alpha}, \mathsf{SS}.pk)$ and his/her private key $\mathsf{sk} = (sk, \mathsf{SS}.sk) = (\alpha, \mathsf{SS}.sk)$.

- $\mathsf{MHT.TagGen}(\mathsf{pk}, \mathsf{sk}, m) \rightarrow \mathsf{T}_m$. The client runs $\mathsf{TagGen}(pk, sk, m) \rightarrow T'_m = (T'_{m_1}, \cdots, T'_{m_n}) \in \mathbb{G}_1^n$ such that $T'_{m_i} = (\prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (\prod_{j=1}^{s} h_j^{m_{i,j}})^{-\alpha} = \prod_{j=1}^{s} h_j^{-\alpha \cdot m_{i,j}}$ for $i \in [1, n]$. S/he also chooses a hash function $H' : \{0,1\}^* \rightarrow \mathbb{G}_1$ seen as a random oracle. Then, s/he creates the MHT according to the document m as follows. For $i \in [1, n]$, the client computes $H'(m_i)$ and assigns this value to the i-th leaf. Once the n leaves refer to the n hash values, the client starts to construct the resulting MHT, and obtains the root rt. Finally, the client signs the root by running $\mathsf{SS.Sign}(\mathsf{SS}.sk, rt) \rightarrow \sigma_{rt}$. Using the hash values, s/he computes the verification metadata as follows: $T_{m_i} = H'(m_i)^{-sk} \cdot T'_{m_i} = (H'(m_i) \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (H'(m_i) \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-\alpha} = H'(m_i)^{-\alpha} \cdot \prod_{j=1}^{s} h_j^{-\alpha \cdot m_{i,j}}$, for $i \in [1, n]$.

  Then, the client stores all the data blocks m_i in an ordered collection $\mathbb{F}$ and the corresponding verification metadata T_{m_i} in an ordered collection $\mathbb{E}$. S/he forwards these two collections as well as (H', σ_{rt}) to the server.

  Once the server received $(\mathbb{F}, \mathbb{E}, H')$, it generates the MHT corresponding to the data uploaded by the client. It sends the resulting root rt_{server} to the client.

  Upon getting the root rt_{server}, the client runs $\mathsf{SS.Verify}(\mathsf{SS}.pk, \sigma_{rt}, rt_{server}) \rightarrow answer$. If $answer = false$, then the client stops the process. Otherwise (i.e. $answer = true$) s/he proceeds and deletes $(\mathbb{F}, \mathbb{E}, \sigma_{rt})$ from his/her local storage and keeps H' for further data operations.

- $\mathsf{MHT.PerfOp}(\mathsf{pk}, \mathbb{F}, \mathbb{E}, R = (\text{operation}, i), info = (m_i, T_{m_i}, \sigma_{rt'})) \rightarrow (\mathbb{F}', \mathbb{E}', rt_{server})$. First, the client sends a request R to the server. Such request $R = (\text{operation}, i)$ should contain at least the type of operation and the position where such operation will be performed.

  Upon receiving the request R, the server selects the AAI Ω_i from the MHT that the client needs in order to generate the root rt' of the updated MHT. It sends Ω_i to the client.

Once the client received Ω_i, s/he first authenticates it: if Ω_i is not the current AAI, then s/he aborts; otherwise, s/he constructs the updated MHT following the update that s/he wants the server to perform on his/her stored data. S/he calculates the new root rt' and signs it by running $\mathsf{SS.Sign}(\mathsf{SS}.sk, rt') \rightarrow \sigma_{rt'}$. Then, the client sends $info = (m_i, T_{m_i}, \sigma_{rt'})$ (note that m_i and T_{m_i} are not necessary in case of deletion).

After receiving the element $info$ from the client, the server first updates the MHT, calculates the new root rt'_{server} and sends this value to the client.

Upon getting the root rt'_{server}, the client runs $\mathsf{SS.Verify}(\mathsf{SS}.pk, \sigma_{rt'}, rt'_{server}) \rightarrow answer$. If $answer = false$, then the client stops the process. Otherwise (i.e. $answer = true$), s/he proceeds and deletes $(m_i, T_{m_i}, \sigma_{rt'})$ from his/her local storage.

- $\mathsf{MHT.GenProof}(\mathsf{pk}, F, chal, \Sigma) \rightarrow (\nu, rt_{server}, \{H'(m_i), \Omega_i\}_{i\in I})$. After a time period to be agreed between the client and the TPA, the latter prepares a challenge $chal$ to be sent to the server as follows: it chooses a subset $I \subseteq [1, n_{max}]$ (n_{max} is the maximum number of blocks after operations), randomly chooses $|I|$ elements $v_i \in_R \mathbb{Z}_p$ and sets $chal = \{(i, v_i)\}_{i\in I}$.

  Then, after receiving the challenge $chal$ which indicates the specific blocks for which the TPA on behalf of the client wants a proof of data possession, the server runs $\mathsf{GenProof}(pk, F, chal, \Sigma) \rightarrow \nu$ such that $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c) \in \mathbb{G}_1^{2s+1}$. More precisely, it sets the ordered collection $F = \{m_i\}_{i\in I} \subset \mathbb{F}$ of blocks and an ordered collection $\Sigma = \{T_{m_i}\}_{i\in I} \subset \mathbb{E}$ which are the verification metadata corresponding to the blocks in F. It then selects at random $r_1, \cdots, r_s \in_R \mathbb{Z}_p$ and computes $R_1 = h_1^{r_1}, \cdots, R_s = h_s^{r_s}$. It also sets $b_j = \sum_{(i,v_i)\in chal} m_{i,j} \cdot v_i + r_j \in \mathbb{Z}_p$ and $B_j = h_j^{b_j}$, for $j \in [1, s]$, as well as $c = \prod_{(i,v_i)\in chal} T_{m_i}^{v_i}$.

  Moreover, the server prepares the latest version of the stored root's signature σ_{rt} provided by the client, the root rt_{server} of the current MHT as well as the hash values $H'(m_i)$ and the AAI Ω_i for the challenged blocks, such that the current MHT can be constructed using $\{H'(m_i), \Omega_i\}_{i\in I}$. Finally, it returns $(\nu, \sigma_{rt}, rt_{server}, \{H'(m_i), \Omega_i\}_{i\in I})$ to the TPA.

- $\mathsf{MHT.CheckProof}(\mathsf{pk}, chal, \nu, \sigma_{rt}, rt_{server}, \{H'(m_i), \Omega_i\}_{i\in I}) \rightarrow true/false$. After receiving the set $\{H'(m_i), \Omega_i\}_{i\in I}$ from the server, the TPA first constructs the MHT and calculates the root rt_{TPA}. It then checks that $rt_{server} = rt_{TPA}$: if not, then it aborts; otherwise, it runs $\mathsf{SS.Verify}(\mathsf{SS}.pk, \sigma_{rt}, rt_{server}) \rightarrow answer$. If $answer = false$, then the TPA stops the process. Otherwise (i.e. $answer = true$), it proceeds and checks whether the following equation holds:

$$e(c, g_2^{\alpha}) \cdot e(\prod_{j=1}^{s} R_j, g_2) \overset{?}{=} e(\prod_{\substack{(i,v_i)\\ \in chal}} H'(m_i)^{v_i}, g_2) \cdot e(\prod_{j=1}^{s} B_j, g_2) \tag{7.7}$$

  If the equation 7.7 holds, then the TPA returns $true$ to the client; otherwise, it returns $false$ to the client.

Correctness. Supposing that the correctness holds for the systems DPDP and SS, if all the algorithms of MHT.DPDP are correctly generated, then the above scheme is correct.

We check the correctness of the proof of data possession:

$$
\begin{aligned}
e(c,g_2^{\alpha})\cdot e(\prod_{j=1}^{s} R_j, g_2) &= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} T_{m_i}^{v_i}, g_2^{\alpha}\right)\cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2)\\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} (H'(m_i)^{-\alpha\cdot v_i}\cdot \prod_{j=1}^{s} h_j^{m_{i,j}\cdot(-\alpha)\cdot v_i}), g_2^{\alpha}\right)\cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2)\\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} H'(m_i)^{v_i}, g_2\right)^{\alpha-\alpha} \cdot e(\prod_{j=1}^{s} h_j^{\sum_{\substack{(i,v_i)\\ \in chal}} m_{i,j}\cdot v_i}, g_2)^{\alpha-\alpha}\\
&\quad \cdot e(\prod_{j=1}^{s} h_j^{r_j}, g_2)\\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} H'(m_i)^{v_i}, g_2\right)\cdot e(\prod_{j=1}^{s} h_j^{\sum_{\substack{(i,v_i)\\ \in chal}} m_{i,j}\cdot v_i + r_j}, g_2)\\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} H'(m_i)^{v_i}, g_2\right)\cdot e(\prod_{j=1}^{s} h_j^{b_j}, g_2)\\
&= e\left(\prod_{\substack{(i,v_i)\\ \in chal}} H'(m_i)^{v_i}, g_2\right)\cdot e(\prod_{j=1}^{s} B_j, g_2)
\end{aligned}
$$

Comments on the MHT-based Construction. Let $\mathsf{SS} = (\mathsf{SS.KeyGen}, \mathsf{SS.Sign}, \mathsf{SS.Verify})$ be a secure digital signature scheme. One concrete and simple example in the random oracle model is $\mathsf{SS.Sign}(\mathsf{SS}.sk, rt) = (H'(rt))^{\alpha} = \sigma_{rt}$, where $H' : \{0,1\}^* \rightarrow \mathbb{G}_1$ is the hash function used to construct the MHT, $\alpha = sk \leftarrow \mathsf{KeyGen}(\lambda)$ is the private key from the basic DPDP scheme with public verifiability and data privacy DPDP, and rt is the root of the current MHT..

A possible security concern is the non-authentication of the TPA. We treat our scheme as publicly verifiable, meaning that everyone has the possibility to check that the server correctly stores the data. Therefore, a malicious user (not necessarily a client of the server) is able to proceed such verification. To avoid such experience, the client can choose and authenticate a particular TPA, and the server is then able to verify that the person who wants to audit it is the TPA selected by the client [233]. More precisely, client, TPA and server proceed as follows. When the client is uploading the data on the server, s/he also chooses a TPA to check the integrity of his/her data, and asks the TPA its identity. The latter encrypts its identity ID_{TPA} under the client's public key pk and sends the resulting ciphertext to the client. Then, the client decrypts the latter to recover ID_{TPA} and includes $\mathsf{SS.Sign}(\mathsf{SS}.sk, ID_{TPA})$ into the other elements (namely the data, the verification metadata, the signature of the MHT root and the hash function H') that are sent to the server. Later, during each challenge-response process, the TPA encrypts its identity ID_{TPA} under the server's public key pk_{server} and sends the resulting ciphertext to the server. Then, the server uses its associated private key sk_{server} to recover the identity ID_{TPA} and runs $\mathsf{SS.Verify}(\mathsf{SS}.pk, \mathsf{SS.Sign}(\mathsf{SS}.sk, ID_{TPA}), ID_{TPA})$ to check the validity of the identity. We assume that the server and a malicious non-authenticated TPA cannot collude together, i.e. the server does not accept to communicate with a non-authenticated TPA.

Note that the updating proof is no longer needed since the client has to authenticate the AAI sent by the server anyway, and has to verify that the updated MHT root rt'_{server} is identical to the client's one rt'. We also argue that the TPA can be required to challenge the server after each update requested by the client.

7.4.2 Security Proofs

Proof of the Security against the Server

Theorem. Let $\mathscr{A}$ be a PPT adversary that has advantage ε against the security of the MHT-based DPDP scheme with public verifiability and data privacy, given a collision resistant hash function H' and a strongly unforgeable digital signature scheme SS. Suppose that $\mathscr{A}$ makes a total of $q_{H'} > 0$ queries to H'. Then, there is a challenger $\mathscr{B}$ that solves the CDH and DL problems $\mathbb{G}$ and $\mathbb{G}_1$ respectively, with advantage $\varepsilon' = O(\varepsilon)$.

For any PPT adversary $\mathscr{A}$ who wins the game, there is a challenger $\mathscr{B}$ that wants to break the CDH and DL problems in $\mathbb{G}$ and $\mathbb{G}_1$ respectively, by interacting with $\mathscr{A}$ as follows.

Setup. $\mathscr{B}$ runs $\mathscr{G}$ on input λ to obtain the tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$. Then, it is given the CDH instance tuple (g, g^a, g^b) such that g is a generator of $\mathbb{G}$ and a, b are unknown exponents in $\mathbb{Z}_p$, chooses two exponents $x, y \in \mathbb{Z}_p$ and computes $g_1 = g^x$ and $g_2 = g^y$. It also chooses two generators g_1 and g_2 of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Note that $(g^a)^x = g_1^a$, $(g^b)^x = g_1^b$, $(g^a)^y = g_2^a$ and $(g^b)^y = g_2^b$. $\mathscr{B}$ chooses $\beta_j, \gamma_j \in_R \mathbb{Z}_p$ and sets $h_j = g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j}$ for $j \in [1, s]$. Let a hash function $H' : \{0,1\}^* \to \mathbb{G}_1$ be controlled by $\mathscr{B}$ as follows. Upon receiving a query m_{i_l} to the random oracle H' for some $l \in [1, q_{H'}]$:

- If (m_{i_l}, θ_l, W_l) exists in $L_{H'}$, return W_l.
- Otherwise, choose $\beta_j, \gamma_j \in_R \mathbb{Z}_p$ and set $h_j = g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j}$, for $j \in [1, s]$. For each i_l, choose $\theta_l \in_R \mathbb{Z}_p$ at random and set

$$W_l = \frac{g_1^{\theta_l}}{g_1^{\sum_{j=1}^{s} \beta_j m_{i_l,j}} (g_1^b)^{\sum_{j=1}^{s} \gamma_j m_{i_l,j}}}$$

for a given block $m_{i_l} = (m_{i_l,1}, \cdots, m_{i_l,s})$. Put (m_{i_l}, θ_l, W_l) in $L_{H'}$ and return W_l as answer.

Note that H' is supposed to be collision resistant and the digital signature scheme SS is supposed to be strongly unforgeable. $\mathscr{B}$ gives $\mathscr{A}$ the public key pk that contains the public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^a)$ as well as the public key $\mathsf{SS}.pk \leftarrow \mathsf{SS.KeyGen}(\lambda)$. It keeps g_1^a, g_1^b, g_2^b and $\mathsf{SS}.sk \leftarrow \mathsf{SS.KeyGen}(\lambda)$ secret. The private key sk is implicitly set as equal to a.

Adaptive Queries. $\mathscr{A}$ has first access to the verification metadata generation oracle $\mathscr{O}_{TG}$ as follows. It adaptively selects several blocks m_i, for $i \in [1,n]$. $\mathscr{B}$ splits each block m_i into s sectors $m_{i,j}$. Then, it computes $T_{m_i} = (W \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-sk} = (W \cdot \prod_{j=1}^{s} h_j^{m_{i,j}})^{-a}$, for $i \in [1,n]$, such that if (m_i, θ, W) exists in $L_{H'}$, then the value W is returned. Otherwise, an element $\theta \in_R \mathbb{Z}_p$ is chosen at random, $W = g^{\theta}$ is computed, and (m_i, θ, W) is put in $L_{H'}$.

$\mathscr{B}$ also creates the MHT resulting from the blocks m_i and gets the corresponding root rt. It signs rt by running $\sigma_{rt} \leftarrow \mathsf{SS.Sign}(\mathsf{SS}.sk, rt)$. It gives the verification metadata T_{m_i} and their corresponding hash values W, along with σ_{rt} to $\mathscr{A}$. The latter sets an ordered collection $\mathbb{F} = \{m_1, \cdots, m_n\}$ of blocks and an ordered collection $\mathbb{E} = \{T_{m_1}, \cdots, T_{m_n}\}$ which are the verification metadata corresponding to the blocks in $\mathbb{F}$.

$\mathscr{A}$ has also access to the data operation performance oracle $\mathscr{O}_{DOP}$ as follows. Repeatedly, $\mathscr{A}$ selects a block m_i and the corresponding elements $(R_i, info_i)$, and forwards them to $\mathscr{B}$. The signature of the root $\sigma_{rt'}$ is included in $info_i$, as the output of the signature algorithm $\mathsf{SS.Sign}(\mathsf{SS}.sk, rt')$. More precisely, i denotes the rank where $\mathscr{A}$ wants the data operation to be performed. Note that only the rank is needed to perform a deletion. Then, $\mathscr{A}$ outputs a new ordered collection $\mathbb{F}'$ (containing the updated version of the block m_i), a new ordered collection $\mathbb{E}'$ (containing the updated version of the verification metadata T_{m_i}) and a new root $rt'_{\mathscr{A}}$ corresponding to the updated MHT. $\mathscr{B}$ runs the algorithm $\mathsf{SS.Verify}$ on the values $\sigma_{rt'}$ and $rt'_{\mathscr{A}}$ and aborts if the answer is equal to $false$; it proceeds otherwise.

Challenge. $\mathscr{A}$ selects blocks m_i^* and the corresponding elements $info_i^*$ and R_i^* for $i \in \mathscr{I} = [1, n']$ for $n' \geq n$, and forwards them to the challenger who checks the data operations. In particular, the first $info_i^*$ indicates a full re-write.

$\mathscr{B}$ chooses a subset $I \subseteq \mathscr{I}$, randomly chooses $|I|$ elements $v_i \in_R \mathbb{Z}_p$ and sets $chal = \{(i, v_i)\}_{i \in I}$. It forwards $chal$ as a challenge to $\mathscr{A}$.

Forgery. Upon receiving the challenge $chal$, the resulting proof of data possession on the correct stored document m should be $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c)$ and pass the equation 7.7. However, $\mathscr{A}$ generates a proof of data possession on an incorrect stored document $\tilde{m}$ as $\tilde{\nu} = (\tilde{R}_1, \cdots, \tilde{R}_s, \tilde{B}_1, \cdots, \tilde{B}_s, \tilde{c})$, such that $\tilde{r}_j$ is randomly chosen from $\mathbb{Z}_p$, $\tilde{R}_j = h_j^{\tilde{r}_j}$, $\tilde{b}_j = \sum_{(i,v_i) \in chal} \tilde{m}_{i,j} \cdot v_i + \tilde{r}_j$ and $\tilde{B}_j = h_j^{\tilde{b}_j}$, for $j \in [1,s]$, as well as $\tilde{c} = \prod_{(i,v_i) \in chal} T_{\tilde{m}_i}^{v_i}$. Finally, it returns $\tilde{\nu} = (\tilde{R}_1, \cdots, \tilde{R}_s, \tilde{B}_1, \cdots, \tilde{B}_s, \tilde{c})$ to $\mathscr{B}$. If the proof of data possession still pass the verification, then $\mathscr{A}$ wins. Otherwise, it fails.

Analysis. We define $\Delta r_j = \tilde{r}_j - r_j$, $\Delta b_j = \tilde{b}_j - b_j = \sum_{(i,v_i) \in chal} (\tilde{m}_{i,j} - m_{i,j}) v_i + \Delta r_j$ and $\Delta \tau_j = \sum_{(i,v_i) \in chal} (\tilde{m}_{i,j} - m_{i,j}) v_i$, for $j \in [1,s]$. Note that r_j and b_j are the elements of a honest proof of data possession ν such that $r_j \in_R \mathbb{Z}_p$ and $b_j = \sum_{(i,v_i) \in chal} m_{i,j} \cdot v_i + r_j$ where $m_{i,j}$ are the actual sectors (not the ones that the adversary claims to have).

We prove that if the adversary can win the game, then solutions to the CDH and DL problems are found, which contradicts the assumption that the CDH and DL problems are hard in $\mathbb{G}$ and $\mathbb{G}_1$ respectively. Let assume that the adversary (playing the role of the server) wins the game. We recall that if $\mathscr{A}$ wins then $\mathscr{B}$ can extract the actual blocks

$\{m_i\}_{(i,v_i)\in chal}$ in polynomially-many interactions with $\mathscr{A}$. Without loss of generality, suppose that $chal = \{(i,v_i)\}$, meaning that the challenge contains only one block.

First case ($\tilde{c} \neq c$): According to the equation 7.7, we have

$$\begin{aligned}
e(\frac{\tilde{c}}{c}, g_2) &= e\left(\frac{T_{\tilde{m}_i}}{T_{m_i}}, g_2\right)^{v_i} = e\left(\frac{(\prod_{j=1}^{s} h_j^{\tilde{\tau}_j})^{-a}}{(\prod_{j=1}^{s} h_j^{\tau_j})^{-a}}, g_2\right) \\
&= e(\prod_{j=1}^{s} h_j^{\Delta\tau_j}, g_2^{-a}) \\
&= e(\prod_{j=1}^{s} (g_1^{\beta_j} \cdot (g_1^b)^{\gamma_j})^{\Delta\tau_j}, g_2^{-a}) \\
&= e(g_1,g_2)^{-a\sum_{j=1}^{s}\beta_j\Delta\tau_j} \cdot e(g_1,g_2)^{-ab\sum_{j=1}^{s}\gamma_j\Delta\tau_j}
\end{aligned}$$

and so, we get that

$$e(\frac{\tilde{c}}{c} \cdot (g_1^a)^{\sum_{j=1}^{s}\beta_j\Delta\tau_j}, g_2) = e(g_1^b, g_2^{-a})^{\sum_{j=1}^{s}\gamma_j\Delta\tau_j}$$

meaning that we have found the solution to the CDH problem, that is

$$(g_1^b)^a = (g^x)^{ab} = (\frac{\tilde{c}}{c} \cdot (g_1^a)^{\sum_{j=1}^{s}\beta_j\Delta\tau_j})^{\frac{-1}{\sum_{j=1}^{s}\gamma_j\Delta\tau_j}}$$

unless evaluating the exponent causes a divide-by-zero. Nevertheless, we notice that not all of the $\Delta\tau_j$ can be zero (indeed, if $\tau_j = m_{i,j} \cdot v_i = \tilde{\tau}_j = \tilde{m}_{i,j} \cdot v_i$ for each $j \in [1,s]$, then $c = \tilde{c}$ which contradicts the hypothesis), and the values γ_j are information theoretically hidden from $\mathscr{A}$ (Pedersen commitments), so the denominator is zero only with probability $1/p$, which is negligible. Finally, since $\mathscr{B}$ knows the exponent x such that $g_1 = g^x$, it can directly compute

$$((\frac{\tilde{c}}{c} \cdot (g_1^a)^{\sum_{j=1}^{s}\beta_j\Delta\tau_j})^{\frac{-1}{\sum_{j=1}^{s}\gamma_j\Delta\tau_j}})^{\frac{1}{x}}$$

and obtains g^{ab}. Thus, if $\mathscr{A}$ wins the game, then a solution to the CDH problem can be found with probability equal to $1 - 1/p$.

Second Case ($\tilde{c} = c$): According to the equation 7.7, we have

$$e(\tilde{c}, g_2^a) = e((H'(m_i)^{v_i}, g_2) \cdot e(\prod_{j=1}^{s}\tilde{B}_j, g_2) \cdot e(\prod_{j=1}^{s}\tilde{R}_j, g_2)^{-1}.$$

Since the proof $\nu = (R_1, \cdots, R_s, B_1, \cdots, B_s, c)$ is a correct one, we also have

$$e(c, g_2^a) = e((H'(m_i)^{v_i}, g_2) \cdot e(\prod_{j=1}^{s}B_j, g_2) \cdot e(\prod_{j=1}^{s}R_j, g_2)^{-1}.$$

We recall that $chal = \{(i,v_i)\}$. From the previous analysis step, we know that $\tilde{c} = c$. Therefore, we get that $\prod_{j=1}^{s}\tilde{B}_j \cdot (\prod_{j=1}^{s}\tilde{R}_j)^{-1} = \prod_{j=1}^{s}B_j \cdot (\prod_{j=1}^{s}R_j)^{-1}$. We can re-write as $\prod_{j=1}^{s}h_j^{\tilde{b}_j - \tilde{r}_j} = \prod_{j=1}^{s}h_j^{b_j - r_j}$ or even as $\prod_{j=1}^{s}h_j^{\Delta b_j - \Delta r_j} = \prod_{j=1}^{s}h_j^{\Delta\tau_j} = 1$.

For two elements $g_1, h \in \mathbb{G}_1$, there exists $\xi \in \mathbb{Z}_p$ such that $h = g_1^{\xi}$ since $\mathbb{G}_1$ is a cyclic group. Without loss of generality, given $g_1, h \in \mathbb{G}_1$, each h_j could randomly and correctly be generated by computing $h_j = g_1^{y_j} \cdot h^{z_j} \in \mathbb{G}_1$ such that y_j and z_j are random values of $\mathbb{Z}_p$. Then, we have $1 = \prod_{j=1}^{s} h_j^{\Delta\tau_j} = \prod_{j=1}^{s} (g_1^{y_j} \cdot h^{z_j})^{\Delta\tau_j} = g_1^{\sum_{j=1}^{s} y_j \cdot \Delta\tau_j} \cdot h^{\sum_{j=1}^{s} z_j \cdot \Delta\tau_j}$. Clearly, we can find a solution to the DL problem. More specifically, given $g_1, h = g_1^{\xi} \in \mathbb{G}_1$, we can compute $h = g_1^{\frac{\sum_{j=1}^{s} y_j \cdot \Delta\tau_j}{\sum_{j=1}^{s} z_j \cdot \Delta\tau_j}} = g_1^{\xi}$ unless the denominator is zero. However, not all of the $\Delta\tau_j$ can be zero and the values z_j are information theoretically hidden from $\mathscr{A}$, so the denominator is only zero with probability $1/p$, which is negligible. Thus, if $\mathscr{A}$ wins the game, then a solution to the DL problem can be found with probability equal to $1 - \frac{1}{p}$.

Therefore, for $\mathscr{A}$, it is computationally infeasible to win the game and generate an incorrect proof of data possession which can pass the verification.

Finally, the simulations of the tag generation oracle $\mathscr{O}_{TG}$ and the data operation performance oracle $\mathscr{O}_{DOP}$ are perfect. The proof is completed.

Proof of the Data Privacy against the TPA

Theorem. Let $\mathscr{A}$ be a PPT adversary that has advantage ε against the strong data privacy property of the MHT-based DPDP scheme with public verifiability and data privacy, given a collision resistant hash function H' and a strongly unforgeable digital signature scheme SS. Suppose that $\mathscr{A}$ makes a total of $q_{H'} > 0$ queries to H. Then, there is a challenger $\mathscr{B}$ that solves the $(s+1)$-DDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$ with advantage $\varepsilon' = O(\varepsilon)$.

We presume that the digital signature scheme SS is strongly unforgeable and the hash function H' is collision resistant. For any PPT adversary $\mathscr{A}$ who wins the game, there is a challenger $\mathscr{B}$ that wants to break the $(s+1)$-DDHE assumption by interacting with the adversary $\mathscr{A}$ as follows.

Setup. $\mathscr{B}$ runs $\mathscr{G}$ on input λ to obtain the tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ and receives the $(s+1)$-DDHE instance $(g_1, g_1^{a}, \cdots, g_1^{a^{s+1}}, g_2, g_2^{a})$ where g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$. $\mathscr{B}$ sets $\tilde{\mu} = 0$ when the output Z of the $(s+1)$-DDHE assumption is equal to $g_1^{a^{s+2}}$; otherwise, it sets $\tilde{\mu} = 1$ when the output Z of the $(s+1)$-DDHE assumption is a random element in $\mathbb{G}_1$. Then, it randomly chooses $\xi_1, \cdots, \xi_s, \xi_{s+1} \in_R \mathbb{Z}_p$, sets the elements $h_1 = (g_1^{a})^{\xi_1}, \cdots, h_s = (g_1^{a^s})^{\xi_s}, h_{s+1} = (g_1^{a^{s+1}})^{\xi_{s+1}}$ and implicitly fixes the private key $sk = a$. $\mathscr{B}$ also controls the hash function $H' : \{0,1\}^* \to \mathbb{G}_1$ as follows. Upon receiving a query m_{i_l} to the random oracle H' for some $l \in [1, q_{H'}]$:

- If $(m_{i_l}, \theta_l, V_l, W_l)$ exists in $L_{H'}$, return V_l and W_l.
- Otherwise, choose $\theta_l \in_R \mathbb{Z}_p$ at random and compute $V_l = g_1^{-\theta_l}$ and $W_l = h_1^{-\theta_l/\xi_1} = g_1^{-a\xi_1\theta_l/\xi_1} = g_1^{-a\theta_l}$. Put $(m_{i_l}, \theta_l, V_l, W_l)$ in $L_{H'}$ and return W_l as answer.

It sets the public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \cdots, h_s, g_2^{a})$. The challenger has also access to a secure digital signature scheme SS and runs the algorithm $\mathsf{SS.KeyGen}(\lambda)$ to obtain the public and private key pair $(\mathsf{SS}.pk, \mathsf{SS}.sk)$. $\mathscr{B}$ sets the public key $\mathsf{pk} = (pk, \mathsf{SS}.pk)$ and forwards it to $\mathscr{A}$. It keeps the private keys sk and $\mathsf{SS}.sk$, as well as the hash function H'.

Query Phase. $\mathscr{A}$ makes verification metadata generation queries as follows: $\mathscr{A}$ first selects a document $m = (m_1, \cdots, m_n)$ and sends it to $\mathscr{B}$. Then, the challenger splits each block m_i into s sectors $m_{i,j}$. Then, it computes $T_{m_i} = W \cdot \prod_{j=1}^{s} g_1^{a^j \cdot (-a) \cdot \xi_j \cdot m_{i,j}} = W \cdot \prod_{j=1}^{s} g_1^{-a^{j+1} \cdot \xi_j \cdot m_{i,j}}$, such that if (m_i, θ, V, W) exists in $L_{H'}$, then the value W is used to compute T_{m_i}; otherwise, an element $\theta \in_R \mathbb{Z}_p$ is chosen at random, $V = g_1^{-\theta}$ and $W = h_1^{-\theta/\xi_1}$ are computed, (m_i, θ, V, W) is put in $L_{H'}$ and W is used for the generation of T_{m_i}. $\mathscr{B}$ also creates the MHT resulting from the document m using the values V such that if (m_i, θ, V, W) exists in $L_{H'}$, then the value V is returned; otherwise, an element $\theta \in_R \mathbb{Z}_p$ is chosen at random, $V = g_1^{-\theta}$ and $W = h_1^{-\theta/\xi_1}$ are computed, and (m_i, θ, V, W) is put in $L_{H'}$. It finally gets the corresponding root rt. It gives the verification metadata $T_m = (T_{m_1}, \cdots, T_{m_n})$ along with the root's signature $\sigma_{rt} \leftarrow \mathsf{SS.Sign}(\mathsf{SS}.sk, rt)$ to $\mathscr{A}$.

Challenge. $\mathscr{A}$ first gives to the challenger two documents $m_0 = (m_{0,1}, \cdots, m_{0,n})$ and $m_1 = (m_{1,1}, \cdots, m_{1,n})$ of equal length and that have not been queried beforehand. $\mathscr{B}$ randomly selects a bit $\mu \in_R \{0,1\}$ and for $i \in [1,n]$, splits each block $m_{\mu,i}$ into s sectors $m_{\mu,i,j}$. Then, it computes $T_{m_{\mu,i}} = W_\mu \cdot \prod_{j=1}^{s} g_1^{-a^{j+1} \cdot \xi_j \cdot m_{\mu,i,j}}$, such that if $(m_{\mu,i}, \theta_\mu, V_\mu, W_\mu)$ exists in $L_{H'}$, then the value W_μ is returned; otherwise, an element $\theta_\mu \in_R \mathbb{Z}_p$ is chosen at random, $V_\mu = g_1^{-\theta_\mu}$ and $W_\mu = h_1^{-\theta_\mu/\xi_1}$ are computed, and $(m_{\mu,i}, \theta_\mu, V_\mu, W_\mu)$ is put in $L_{H'}$. $\mathscr{B}$ also creates the MHT resulting from the document m_μ using the values V_μ such that if $(m_{\mu,i}, \theta_\mu, V_\mu, W_\mu)$ exists in $L_{H'}$, then the value V_μ is returned; otherwise, an element $\theta_\mu \in_R \mathbb{Z}_p$ is chosen at random, $V_\mu = g_1^{-\theta_\mu}$ and $W_\mu = h_1^{-\theta_\mu/\xi_1}$ are computed, and $(m_{\mu,i}, \theta_\mu, V_\mu, W_\mu)$ is put in $L_{H'}$. It finally gets the corresponding root rt_μ. It gives the verification metadata $T_{m_\mu} = (T_{m_{\mu,1}}, \cdots, T_{m_{\mu,n}})$ along with the root's signature $\sigma_{rt_\mu} \leftarrow \mathsf{SS.Sign}(\mathsf{SS}.sk, rt_\mu)$ to $\mathscr{A}$.

Without loss of generality, $\mathscr{A}$ generates a challenge on one block only. It chooses a subset $I = \{i^*\} \subseteq [1,n]$, randomly chooses an element $v_{i^*} \in_R \mathbb{Z}_p$ and sets $chal = \{(i^*, v_{i^*})\}$. It forwards $chal$ as a challenge to $\mathscr{B}$. Upon receiving the challenge $chal$, $\mathscr{B}$ selects an ordered collection $F_\mu = \{m_{\mu,i^*}\}$ of blocks and an ordered collection $\Sigma_\mu = \{T_{m_{\mu,i^*}}\}$ which are the verification metadata corresponding to the blocks in F_μ such that $T_{m_{\mu,i^*}} = W_\mu \cdot \prod_{j=1}^{s} g_1^{-a^{j+1} \cdot \xi_j \cdot m_{\mu,i^*,j}}$. It then randomly chooses $r_1, \cdots, r_s \in_R \mathbb{Z}_p$ and computes $R_1^* = h_1^{a^2 \cdot r_1} = h_3^{\frac{\xi_1}{\xi_3} \cdot r_1} = g_1^{a^3 \cdot \xi_1 \cdot r_1}, \cdots, R_{s-1}^* = h_{s-1}^{a^2 \cdot r_{s-1}} = h_{s+1}^{\frac{\xi_{s-1}}{\xi_{s+1}} \cdot r_{s-1}} = g_1^{a^{s+1} \cdot \xi_{s-1} \cdot r_{s-1}}$ and $R_s^* = Z^{\xi_s \cdot r_s}$. It implicitly fixes

$$b_1 = a^2 \cdot (m_{\mu,i^*,1} \cdot v_{i^*} + r_1), \cdots, b_{s-1} = a^2(\cdot m_{\mu,i^*,s-1} \cdot v_{i^*} + r_{s-1})$$

by computing

$$\begin{aligned} B_1^* &= h_1^{b_1} = h_1^{a^2 \cdot m_{\mu,i^*,1} \cdot v_{i^*}} \cdot h_1^{a^2 \cdot r_1} = h_3^{\frac{\xi_1}{\xi_3} \cdot m_{\mu,i^*,1} \cdot v_{i^*}} \cdot h_3^{\frac{\xi_1}{\xi_3} \cdot r_1} = g_1^{a^3 \cdot \xi_1 \cdot m_{\mu,i^*,1} \cdot v_{i^*}} \cdot g_1^{a^3 \cdot \xi_1 \cdot r_1} \\ & \dots \\ B_{s-1}^* &= h_{s-1}^{b_{s-1}} = h_{s-1}^{a^2 \cdot m_{\mu,i^*,s-1} \cdot v_{i^*}} \cdot h_{s-1}^{a^2 \cdot r_{s-1}} = h_{s+1}^{\frac{\xi_{s-1}}{\xi_{s+1}} \cdot m_{\mu,i^*,s-1} \cdot v_{i^*}} \cdot h_{s+1}^{\frac{\xi_{s-1}}{\xi_{s+1}} \cdot r_{s-1}} \\ &= g_1^{a^{s+1} \cdot \xi_{s-1} \cdot m_{\mu,i^*,s-1} \cdot v_{i^*}} \cdot g_1^{a^{s+1} \cdot \xi_{s-1} \cdot r_{s-1}} \end{aligned}$$

and $B_s^* = Z^{\xi_s \cdot m_{\mu,i^*,s} \cdot v_{i^*}} \cdot Z^{\xi_s \cdot r_s}$. It sets $c^* = T_{m_{\mu,i^*}}^{v_{i^*}} = W_\mu^{v_{i^*}} \cdot \prod_{j=1}^s g_1^{-a^{j+1} \cdot \xi_j \cdot m_{\mu,i^*,j} \cdot v_{i^*}}$ as well.

Finally, $\mathscr{B}$ returns $\nu^* = (R_1^*, \cdots, R_s^*, B_1^*, \cdots, B_s^*, c^*)$ along with $(V_\mu, \Omega_{\mu,i^*})$, where Ω_{μ,i^*} is the AAI needed to create the MHT based on V_μ. Note that Ω_{μ,i^*} is generated by calling successively the random oracle H'. This means that, in the list $L_{H'}$, we can find tuples of the form (z, θ, V, W) such that the query z can be either a data block m_i as we defined above (meaning that m_i is appended to a leaf node) or a value $H'(y)$ that is attached to an internal node, where y is calculated according to the MHT construction rules.

If $\tilde{\mu} = 0$ then $Z = g_1^{a^{s+2}}$. Therefore, the proof of data possession is a valid random proof for the document m_μ. Otherwise, if $\tilde{\mu} = 1$, then Z is random value in $\mathbb{G}_1$. Since Z is random, the values R_s^* and B_s^* will be random elements of $\mathbb{G}_1$ from the adversary's view and the proof of data possession contains no information about m_μ.

Guess. $\mathscr{A}$ returns a bit μ'. If $\mu = \mu'$, $\mathscr{B}$ will output $\tilde{\mu}' = 0$ to indicate that it was given a $(s+1)$-DDHE tuple; otherwise it will output $\tilde{\mu}' = 1$ to indicate that it was given a random tuple.

Analysis. We first recall that the digital signature scheme SS is assumed to be strongly unforgeable and the hash function H' is supposed to be collision resistant. We prove that the verification metadata and the proof of data possession given to $\mathscr{A}$ are correctly distributed. In the case where $\tilde{\mu} = 1$, $\mathscr{A}$ gains no information about μ. Therefore, we have $Pr[\mu \neq \mu' | \tilde{\mu} = 1] = 1/2$. Since $\mathscr{B}$ guesses $\tilde{\mu}' = 1$ when $\mu \neq \mu'$, we have $Pr[\tilde{\mu} = \tilde{\mu}' | \tilde{\mu} = 1] = 1/2$. If $\tilde{\mu} = 0$, then $\mathscr{A}$ sees an upload of m_μ. The adversary's advantage in this situation is negligible by definition, i.e. equal to a given ε. Therefore, we have $Pr[\mu \neq \mu' | \tilde{\mu} = 0] = 1/2 + \varepsilon$. Since $\mathscr{B}$ guesses $\tilde{\mu}' = 0$ when $\mu = \mu'$, we have $Pr[\tilde{\mu} = \tilde{\mu}' | \tilde{\mu} = 0] = 1/2 + \varepsilon$. The value $T_{m_{\mu,i}}$ is equal to $(H'(m_{\mu,i}) \cdot \prod_{j=1}^s h_j^{m_{\mu,i,j}})^{-sk}$ where sk is implicitly set as equal to a and the h_j's are correclty distributed as in the real game. We recall that the elements SS.sk and H' are kept secret from $\mathscr{A}$. Note that $\mathscr{A}$ does not have access to the random oracle H' and the AAI given to the adversary with the proof of data possession results from calls to H'. In addition, $R_1^*, \cdots, R_s^*, B_1^*, \cdots, B_s^*$ are statically indistinguishable with the actual outputs corresponding to m_0 or m_1. Thus, the answers given to $\mathscr{A}$ are correctly distributed. The proof is completed.

N.B. Such security level is reached for data privacy since the hash function H' is kept secret by the server and the client and so, the adversary as the TPA does not have access to it.

7.4.3 Performance of the MHT-based DPDP Scheme

We compare the MHT-based DPDP scheme with public verifiability and data privacy with the basic scheme DPDP presented in Section 7.1. In the MHT-based construction, the client is able to verify the first upload and the updates without the help of the TPA. S/he authenticates the AAI given by the server, uses his/her MHT root and compares it with the one provided by the server. The TPA is required for data integrity checks: it regularly challenges the server to prove that the latter correctly stores the data of the client. The client and the TPA share a list containing block indices (or at least the maximum number n_{max} of blocks given after the operations that the client wanted the server to perform on his/her data) and the server stores the data, the verification metadata, the signature of the client's root (also regularly updated after the data operations) and the hash function H'. The server might construct the MHT each time this is needed or stores the latest version of the entire tree.

Obviously, the communication and the computational overheads grow. First of all, when the client is uploading the document for the first time, s/he has to compute all the elements in order to construct the MHT resulting from the document and compare the root with the one from the server's MHT. However, such elements are hash values, that are elements easily computable.

Moreover, the server stores more elements: the MHT and the signature of the client's root for each document, along with the document itself and the corresponding verification metadata as in Section 7.1. Note that for each time the server is asked to perform an operation, it has to update the MHT accordingly. Nevertheless, as suggested in Section 7.1, the server has huge computational and storage resources, thus this should not be a constraint for it. Note that the server can only store m_i, T_{m_i}, σ_{rt} and H' and constructs the MHT whenever necessary; however, this option does not seem more attractive.

Then, the communication burden increases between the client and the server, especially for the data operation process. More precisely, the client has first to inform the server that s/he wants to make an operation on his/her data and asks the necessary information. From this request, the server sends some AAI to the client in order to start the data operation process. Upon receiving such information, the client has to authenticate it and then create the MHT according to the data update to be performed. Then, as for the first upload, the client sends the resulting information back to the server. Using such elements, the server can perform the operation on the stored data and on the MHT, and obtain a new version of the root that is forwarded back to the client. Such root is compared with the one given by the client (under the form of a signature) on the client's side. However, we no longer need the help of the TPA to check the updates. Thus, the server no longer has to generate an updating proof through the algorithm PerfOp and the TPA no longer has to run the algorithm CheckOp to verify the correctness of the updating proof, compared to the scheme in Section 7.1. Overall, implementing a MHT-based scheme seems to fairly affect the computation and communication overheads.

In the MHT-based system, we assume that the client deletes all the elements related to the document (blocks, verification metadata, root signatures, MHTs) from his/her local storage, except a list containing the block indices that are needed for requesting a data integrity verification. Such list may be shared with the TPA since the latter works on behalf of the client in the challenge-response process. In order to reduce the communication overhead between the client and the server when the former prepares an update operation, we can force the client to store on his/her local storage the current MHT or a list of the hash values corresponding to all the leaves of the current MHT. Doing this, the client does not need to ask for the AAI before updating the data and so, the server does not need to send it back to the client. Note that the latter no longer needs to authenticate it, since s/he keeps the current version of the MHT.

If the client keeps the entire MHT, the hash values at internal nodes and the root are already computed; however s/he has to store $2^{k+1}-1$ elements for $n = 2^k$. Moreover, when an operation is performed, the client does not need to calculate all the hash values at internal nodes, but only the ones on the path from the modified leaves until the root, as well as the root itself. If the client keeps only a list of hash values corresponding to all the leaves of the MHT, meaning that s/he stores only n elements, s/he has to compute all the intermediary hash values at upper levels and the root each time s/he inserts, deletes or modifies data blocks. This option is possible if the client possesses enough resources for storage and computation.

The MHT-based construction seems less practical and efficient. Communication and computational burdens appear in order to obtain the desired security standards against the server and the TPA. The communication overheads increase between the client and the server. The computational overheads for the client raises also, although the client is supposed to be limited in resources. The storage space of the server should be bigger, since the server has to create and possibly stores MHTs for each client it has. The TPA has to provide more computational resources for each client in order to ensure valid data integrity checks. Nevertheless, experiments might show that the time gap between the algorithms in the scheme proposed in Section 7.1 and the ones in the MHT-based scheme is acceptable.

Comparison with the Existing Schemes. The MHT is an authenticated data structure (ADS) that allows the client and the TPA to check that the server correctly stores and updates the data blocks.

Erway et al. [77] proposed the first DPDP scheme. The verification of the data updates is based on a modified ADS, called rank-based authentication skip list (RASL). This provides authentication of the data block indices, which ensures security in regards to data block dynamicity. However, public verifiability is not reached. Note that such ADS with bottom-up levelling limits the insertion operations. For instance, if the leaf nodes are at level 0, any data insertion that creates a new level below the level 0 will bring necessary updates of all the level hash values and the client might not be able to verify.

Wang et al. [232] first presented a DPDP with public verifiability using MHT. However, security proofs and technical details lacked. The authors revised the aforementioned paper [232] and proposed a more complete paper [233] that focuses on dynamic and publicly verifiable PDP systems based on BLS signatures. To achieve the dynamicity property, they employed MHT. Nevertheless, because the check of the block indices is not done, the server can delude the client by corrupting a challenged block as follows: it is able to compute a valid proof with other non-corrupted blocks. Thereafter, in a subsequent work [230], Wang et al. suggested to add randomization to the above system [233], in order to guarantee that the server cannot deduce the contents of the data documents from the proofs of data possession.

Liu et al. [148] constructed a PDP protocol based on MHT with top-down levelling. Such protocol satisfies dynamicity and public verifiability. They opted for such design to let leaf nodes be on different levels. Thus, the client and the TPA have both to remember the total number of data blocks and check the block indices from two directions (leftmost to rightmost and vice versa) to ensure that the server does not delude the client with another node on behalf of a data block during the data integrity checking process.

In this section, the dynamic and publicly verifiable PDP scheme is based on MHT with bottom-up levelling, such that data block indices are authenticated. Such tree-based construction guarantees security for both dynamicity and public verifiability properties. We have explained how we can ensure that the server cannot successfully generate a correct proof of data possession without storing all the data blocks and the TPA cannot get any information about the challenged data blocks.

7.5 Conclusion

We first proposed a basic DPDP system with public verifiability and data privacy in the standard model. Unfortunately, the construction appeared to be vulnerable to three attacks.

We thus provided two solutions to solve the adversarial issues found in the basic scheme. These solutions manage to overcome replay attacks, replace attacks and attacks against data privacy by embedding index hash tables (IHT) or Merkle hash trees (MHT) into the basic construction. We proved that our two new schemes are both secure against the server and data private in the random oracle and we also showed that the practicality of these two new constructions is just slightly affected compared to the one of the basic DPDP scheme with public verifiability and data privacy.

Chapter 8

Conclusion and Future Work

8.1 Summary of the Contributions

Numerous solutions about the storage and access management of EHRs have been proposed in the literature. For instance, the issues related to EHR access control have been well studied and partially solved. In addition, problems arisen with EHRs stored on cloud servers have been carefully described and almost overcome. Nevertheless, since e-Health is a wide topic, this is difficult to approach it as one all and give an satisfactory solution that embeds all the problems that EHR management can encounter.

In this thesis, we focused on information security and cryptography in the perspective of e-Health. We decided to approach the EHR-based context by dividing it into subcases in order to present successful and effective solutions. In particular, we separated access control and cloud computing environments since these two aspect of EHR management do not necessarily bring the same kinds of security and efficiency concerns. We studied various healthcare environment-based scenarios, while these scenarios remained applicable to situations involving other kind of personal information. In summary, we met the following problematic aspects:

- *Efficiency and practicality* in terms of computation and communication;
- *Security and privacy* for both users' identities and data.

These issues need to be overcome since EHRs have been chosen as the option to archive sensitive personal medical information and would be propagated worldwide.

In each chapter, we first described the problematic aspects met in an e-Health environment, and we then presented EHR-based cryptographic primitives to answer them. The following cryptographic features and tools have been used to design our primitives:

- The definitions of our schemes followed the public-key encryption setting. This means that users receive a public and private key pair, where the public key is used for message encryption and the corresponding private key is required for successful decryption.
- Pairings were used in our constructions. More precisely, we used maps with domain and range in cyclic groups of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of order the prime p. We recall that e is said to be a symmetric pairing map when $\mathbb{G}_1 = \mathbb{G}_2$; e is said to be an asymmetric pairing map otherwise.

- We sometimes required the random oracle model to prove our schemes secure at a satisfactory level. To do so, we used hash functions seen as random oracles.
- We tried as much as possible to rely the security proofs of our schemes on simple algorithmic problems, such that these problems are shown to be hard to solve.

We now summarise our work in each chapter. In Chapter 4, we presented the following cryptographic primitives to broadcast and share EHRs among the invovled users in a secure and efficient way.

Broadcast Encryption with Membership. The broadcast encryption with membership (BEM) primitive has been presented as a solution to allow the medical institute to forward encrypted medical information such that only hospital staff members selected by the hospital can retrieve the original information while the medical insitute does not know the identities of these authorised staff members.

We gave the definition of such BEM primitive along with the security models that should be satisfied. We then proposed a BEM construction and proved it semi-statically IND-CCA (indistinguishability against chosen ciphertext attack) secure, privacy preserving and maximum number accountability secure in the standard model. In addition, we achieved constant size for the ciphertexts and the tokens. We also suggested a technique to transform a semi-statically IND-CCA secure BEM scheme into an adaptively secure BEM scheme.

Certificate-Based Broadcast Encryption. We presented the certificate-based broadcast encryption (CBBE) primitive as a solution to let the hospital to send encrypted medical information to all the staff members while only members authorised by the hospital and holding valid licenses delivered by medical legislators are able to successfully recover the information in plain.

We carefully defined such CBBE primitive and its security models. We gave a first CBBE construction that is adaptively IND-CCA secure in the standard model, with components having size linear in the number of both the legislators and the staff members. We presented a second efficient CBBE scheme that is selectively IND-CCA secure and selectively collusion resistant in the random oracle model, with short ciphertexts, private keys and certificates.

In Chapter 5, we presented the following cryptographic primitive to access and retrieve EHRs stored on cloud servers in a secure and efficient way.

Certificate-Based Encryption with Keyword Search. We proposed a solution that we called certificate-based encryption with keyword search (CBEKS), to let a cloud server to check the requests sent by the hospital staff members when they wish to access medical information. To do so, the cloud server verifies two components: the keyword as a description of the searched information and the label as a representation of the access right held by the staff members.

We properly gave a definition for such CBEKS primitive along with the security models for computational consistency, IND-CKCA (indistinguishability against chosen keyword and ciphertext attack) security, IND-KGA (indistinguishability against keyword-guessing attack) security and collusion resistance. We constructed a CBEKS scheme that is provably secure regarding the aforementioned security models, in the standard model. We also managed to get all the components with constant size.

In Chapter 6, we presented the following cryptographic primitives to ensure patient privacy and save computational and communication resources in EHR-based storage in a secure and efficient way.

On-line/Off-line Ciphertext-Policy Attribute-Based Proxy Re-Encryption. The on-line/off-line ciphertext-policy attribute-based proxy re-encryption (OO-CP-AB-PRE) appeared to be a solution to enable hospital staff members to pre-encrypt medical information off-line and to complete the encryption online later, regarding requirements (credentials) that members should satisfy in order to retrieve the original information.

We provided the definition of such OO-CP-AB-PRE and its security models. Our OO-CP-AB-PRE construction is efficient and practical with components' size linear in the number of credentials, and proved selectively IND-CCA secure and selectively collusion resistant in the random oracle model.

Ciphertext-Policy DNA-Based Encryption. The cipertext-policy DNA-based encryption (CP-DBE) is a solution to a particular case, that is, encrypting and decrypting information using DNA sequences. We noticed that DNA properties are interesting to design public-key cryptography since the DNA of an individual is unique but has some parts shared with relatives. We recall that we considered the repetition of the minisatellites as the feature to base on our primitive.

We proposed a definition for such CP-DBE primitive and the security models that this primitive should meet. We constructed a CP-DBE scheme and proved it selectively IND-CCA secure and selectively collusion resistant in the random oracle model. This construction has components with size linear in the number of minisatellites.

In Chapter 7, we presented the following cryptographic primitive to manage EHRs in cloud computing in a secure and efficient way.

Dynamic Provable Data Possession with Public Verifiability and Data Privacy. A solution to allow hospital staff members and patients to upload, share and update medical information stored on a cloud server is the dynamic provable data possession (DPDP) primitive with public verifiability and data privacy.

We first defined the DPDP primitive with public verifiability and data privacy and the security models that are security against the cloud server and data privacy against the third party auditor (TPA). We then suggested a basic construction for a DPDP scheme with public verifiability and data privacy in the standard model. Unfortunately, the construction appeared to be vulnerable to three attacks, namely the replay attack, the replace attack and the attack against strong data privacy.

We thus provided two other constructions for a DPDP scheme with public verifiability and data privacy to solve the adversarial issues found in the basic scheme. These solutions manage to overcome the three aforementioned attacks by embedding IHT and MHT into the basic construction respectively. We finally showed that these two constructions are secure against the server and data private in the random oracle.

8.2 Future Work

Future work includes the improvement of the above primitives in terms of better efficiency and practicality, as well as in terms of stronger security level. For instance, all the constructions taking place in the random oracle model should be extended into the standard model in order to achieve a more realistic security goal. Future work also includes the design of new primitives as answers to existing and non-existing scenarios. Indeed, we do not argue that a primitive proposed in this thesis is the unique solution to a given scenario. Indeed, another primitive can be proposed as another solution to the same scenario.

Since EHRs will be propagated worldwide, current issues may become more prevalent while new scenarios will be formed and so, new issues may arise. In addition, we observe that EHRs will be information with the need of big data framework. Indeed, we imagine that most of the nations will end up by adopting EHRs as the option to archive their medical information, therefore the quantity of data that contain sensitive personal information about human beings will be huge from an earch point of view. Therefore, works on access control and cloud computing in the perspective of big EHR data have to be done.

Bibliography

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *Journal of Cryptology*, 21(3):350–391, March 2008.

[2] Rakesh Agrawal and Christopher Johnson. Securing electronic health records without impeding the flow of information. *International Journal of Medical Informatics*, 76(56):471 – 479, 2007.

[3] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast digital identity revocation. In *Proceedings of the 18th Annual International Cryptology Conference*, CRYPTO 1998, pages 137–152, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[4] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[5] Ahmed Al Faresi, Duminda Wijesekera, and Khaled Moidu. A comprehensive privacy-aware authorization framework founded on hipaa privacy rules. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI 2010, pages 637–646, New York, NY, USA, 2010. ACM.

[6] Saleh Al-zharani, Sababady Sarasvady, Harish Chandra, and Pit Pichappan. Controlled ehr access in secured health information system. In *Proceedings of the 1st International Conference on Digital Information Management*, pages 63–68, Dec 2007.

[7] José Luis Fernández Alemán, Inmaculada Carrión Señor, Pedro Ángel Oliver Lozoya, and Ambrosio Toval. Security and privacy in electronic health records: A systematic literature review. *Journal of Biomedical Informatics*, 46(3):541–562, 2013.

[8] Bandar Alhaqbani and Colin Fidge. Access control requirements for processing electronic health records. In *Proceedings of the International Workshops on Business Process Management*, BPM 2007, pages 371–382, Berlin, Heidelberg, 2008. Springer.

[9] Bandar Alhaqbani and Colin Fidge. Privacy-preserving electronic health record linkage using pseudonym identifiers. In *Proceedings of the 10th International*

Conference on e-health Networking, Applications and Services, HealthCom 2008, pages 108–117, July 2008.

[10] Asma Aljarullah and Samir El-Masri. A novel system architecture for the national integration of electronic health records: A semi-centralized approach. *Journal of Medical Systems*, 37(4):1–20, August 2012.

[11] Claudio A. Ardagna, Sabrina De Capitani Di Vimercati, Sara Foresti, Tyrone W. Grandison, Sushil Jajodia, and Pierangela Samarati. Access control for smarter healthcare using policy spaces. *Computers and Security*, 29(8):848–858, November 2010.

[12] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS 2007, pages 598–609, New York, NY, USA, 2007. ACM.

[13] Giuseppe Ateniese, Roberto Di Pietro, Luigi V. Mancini, and Gene Tsudik. Scalable and efficient provable data possession. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, SecureComm 2008, pages 9:1–9:10, New York, NY, USA, 2008. ACM.

[14] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, February 2006.

[15] Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie de Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science*, 422:15–38, March 2012.

[16] Nuttapong Attrapadung, Benoît Libert, and Elie De Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography*, PKC 2011, pages 90–108, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic universal accumulators for ddh groups and their application to attribute-based anonymous credential systems. In *Proceedings of the Cryptographers' Track at the RSA Conference*, CT-RSA 2009, pages 295–308, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[18] Michael Backes, Christian Cachin, and Alina Oprea. Secure key-updating for lazy revocation. In *Proceedings of the 11th European Conference on Research in Computer Security*, ESORICS 2006, pages 327–346, Berlin, Heidelberg, 2006. Springer-Verlag.

[19] Michael Backes and Alina Oprea. Lazy revocation in cryptographic file systems. In *Proceedings of the 3rd IEEE International Security in Storage Workshop*, SISW 2005, pages 1–11, Washington, DC, USA, 2005. IEEE Computer Society.

[20] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In *Proceedings of the 9th International Conference on Information Security*, ISC 2006, pages 217–232, Berlin, Heidelberg, 2006. Springer.

[21] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In *Proceedings of the The 8th International Conference on Computational Science and Its Applications*, volume 5072 of *ICCSA 2008*, pages 1249–1259. Springer Berlin Heidelberg, 2008.

[22] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005. `http://eprint.iacr.org/2005/417`.

[23] Niko Baric and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT 1997, pages 480–494, Berlin, Heidelberg, 1997. Springer-Verlag.

[24] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel, 1996.

[25] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 2007, pages 535–552, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 1998, pages 26–45. Springer-Verlag, 1998.

[27] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS 1993, pages 62–73, New York, NY, USA, 1993. ACM.

[28] Josh Benaloh, Melissa Chase, Eric Horvitz, and Kristin Lauter. Patient controlled encryption: Ensuring privacy of electronic medical records. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW 2009, pages 103–114, New York, NY, USA, 2009. ACM.

[29] Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT 1993, pages 274–285. Springer-Verlag New York, 1994.

[30] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP 2007, pages 321–334. IEEE Computer Society, 2007.

[31] Eli Biham, Dan Boneh, and Omer Reingold. Breaking generalized diffie-hellman modulo a composite is no easier than factoring. *Information Processing Letters*, 70(2):83–87, April 1999.

[32] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, volume 1403 of *EUROCRYPT 1998*, pages 127–144. Springer-Verlag, June 1998.

[33] Alexandra Boldyreva and Nathan Chenette. Efficient fuzzy search on encrypted data. *IACR Cryptology ePrint Archive*, 2014:235, 2014.

[34] Dan Boneh. The decision diffie-hellman problem. In *Algorithmic Number Theory*, volume 1423, pages 48–63. Springer Berlin Heidelberg, 1998.

[35] Dan Boneh and Xavier Boyen. *Short Signatures Without Random Oracles*.

[36] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Proceedings of the 24th Annual International Cryptology Conference*, CRYPTO 2004, pages 443–459, Berlin, Heidelberg, 2004. Springer.

[37] Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption withoutrandom oracles. *Journal of Cryptology*, 24(4):659–693, 2010.

[38] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proceedings of the 24th International Conference on the Theory and Application of Cryptographic Techniques*, EUROCRYPT 2005, pages 440–456. Springer Berlin Heidelberg, 2005.

[39] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Proceedings of the 24th Annual International Cryptology Conference*, CRYPTO 2004, pages 41–55, Berlin, Heidelberg, 2004. Springer.

[40] Dan Boneh, Giovanni Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Proceedings of the 23rd International Conference on Theory and Applications to Cryptographic Techniques*, EUROCRYPT 2004, pages 506–522, Berlin, Heidelberg, 2004. Springer.

[41] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21th Annual International Cryptology Conference*, CRYPTO 2001, pages 213–229. Springer Berlin Heidelberg, 2001.

[42] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, CRYPTO 2005, pages 258–275, Berlin, Heidelberg, 2005. Springer-Verlag.

[43] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT 2001, pages 514–532, Berlin, Heidelberg, 2001. Springer-Verlag.

[44] Jan Jaap Bos. Digital signatures and the electronic health records: providing legal and security guarantees. *International Journal of Medical Informatics*, 42:157–163, 1996.

[45] Omar Bouhaddou, Tim Cromwell, Mike Davis, Sarah Maulden, Nelson Hsing, David Carlson, Jennifer Cockle, Catherine Hoang, and Linda Fischetti. Translating standards into practice: Experience and lessons learned at the department of veterans affairs. *Journal of Biomedical Informatics*, 45(4):813 – 823, 2012.

[46] Bart Bouwman, Sjouke Mauw, and Milan Petkovic. Rights management for role-based access control. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference*, pages 1085–1090, January 2008.

[47] Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Proceedings of the Third VLDB Workshop on Secure Data Management*, SDM 2006, pages 75–83, Berlin, Heidelberg, 2006. Springer.

[48] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT 2008, pages 234–252, Berlin, Heidelberg, 2008. Springer-Verlag.

[49] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography*, PKC 2009, pages 481–500, Berlin, Heidelberg, 2009. Springer-Verlag.

[50] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings of the 30th Annual International Cryptology Conference*, CRYPTO 2002, pages 61–76. Springer Berlin Heidelberg, 2002.

[51] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, EUROCRYPT 2003, pages 255–271. Springer Berlin Heidelberg, 2003.

[52] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS 2007, pages 185–194, New York, NY, USA, 2007. ACM.

[53] Melissa Chase. Multi-authority attribute based encryption. In *Proceedings of the 4th Conference on Theory of Cryptography*, TCC 2007, pages 515–534, Berlin, Heidelberg, 2007. Springer-Verlag.

[54] Melissa Chase and Sherman S.M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS 2009, pages 121–130, New York, NY, USA, 2009. ACM.

[55] David Chaum and Eugène Van Heyst. Group signatures. In *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT 1991, pages 257–265, Berlin, Heidelberg, 1991. Springer-Verlag.

[56] Rong-Jaye Chen and George Hsieh. Design for a secure interoperable cloud-based personal health record service. In *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science*, CLOUDCOM 2012, pages 472–479, Washington, DC, USA, 2012. IEEE Computer Society.

[57] Tzer-Shyong Chen, Chia-Hui Liu, Tzer-Long Chen, Chin-Sheng Chen, Jian-Guo Bau, and Tzu-Ching Lin. Secure dynamic access control scheme of phr in cloud computing. *Journal of Medical Systems*, 36(6):4005–4020, December 2012.

[58] Yu-Yi Chen, Jun-Chao Lu, and Jinn-Ke Jan. A secure ehr system based on hybrid clouds. *Journal of Medical Systems*, 36(5):3375–3384, October 2012.

[59] Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2006, pages 1–11, Berlin, Heidelberg, 2006. Springer-Verlag.

[60] Ling Cheung and Calvin Newport. Provably secure ciphertext policy abe. Cryptology ePrint Archive, Report 2007/183, 2007.

[61] Jun Choe and Sun K. Yoo. Web-based secure access from multiple patient repositories. *International Journal of Medical Informatics*, 77(4):242–248, 2008.

[62] Young Ju Choie, Eun Kyung Jeong, and Eun Jeong Lee. Supersingular hyperelliptic curves of genus 2 over finite fields. *Applied Mathematics and Computation*, 163(2):565–576, April 2005.

[63] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, Berlin, Heidelberg, 2001. Springer-Verlag.

[64] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 1994, pages 174–187, London, UK, UK, 1994. Springer-Verlag.

[65] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 1998, pages 13–25, London, UK, UK, 1998. Springer-Verlag.

[66] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, 33(1):167–226, 2004.

[67] David Daglish and Norm Archer. Electronic personal health record systems: A brief review of privacy, security, and architectural issues. In *Proceedings of the*

World Congress on Privacy, Security, Trust and the Management of e-Business, CONGRESS 2009, pages 110–120, Aug 2009.

[68] Ivan Damgård. Collision free hash functions and public key signature schemes. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, EUROCRYPT 1987, pages 203–216, Berlin, Heidelberg, 1988. Springer.

[69] Marnix A. C. Dekker and Sandro Etalle. Audit-based access control for electronic health records. *Electronic Notes in Theoretical Computer Science*, 168:221–236, feb 2007.

[70] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *Proceedings of the Advances in Crypotology 13th International Conference on Theory and Application of Cryptology and Information Security*, ASIACRYPT 2007, pages 200–215, Berlin, Heidelberg, 2007. Springer-Verlag.

[71] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Proceedings of the International Conference on Pairing-Based Cryptography*, Pairing 2007, pages 39–59. Springer Berlin Heidelberg, 2007.

[72] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, September 2006.

[73] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management*, volume 2696 of *LNCS*, pages 61–80. Springer Berlin Heidelberg, 2003.

[74] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC 1991, pages 542–552. ACM, 1991.

[75] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of the 4th Annual International Cryptology Conference*, CRYPTO 1984, pages 10–18. Springer-Verlag New York, 1985.

[76] Bernice S. Elger, Jimison Iavindrasana, Luigi Lo Iacono, Henning Müller, Nicolas Roduit, Paul Summers, and Jessica Wright. Strategies for health data exchange for secondary, cross-institutional clinical research. *Computer Methods and Programs in Biomedicine*, 99(3):230–251, September 2010.

[77] Chris Erway, Alptekin Küpçü, Charalampos Papamanthou, and Roberto Tamassia. Dynamic provable data possession. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS 2009, pages 213–222, New York, NY, USA, 2009. ACM.

[78] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, March 1996.

[79] Chun-I Fan, Pei-Jen Tsai, Jheng-Jia Huang, and Wen-Tsuen Chen. Anonymous multi-receiver certificate-based encryption. In *CyberC*, pages 19–26. IEEE, 2013.

[80] Xinyu Fan, Guomin Yang, Yi Mu, and Yong Yu. On indistinguishability in remote data integrity checking. *The Computer Journal*, 58(4):823–830, April 2015.

[81] Liming Fang, Willy Susilo, Chunpeng Ge, and Jiandong Wang. A secure channel free public key encryption with keyword search scheme without random oracle. In *Cryptology and Network Security*, volume 5888 of *LNCS*, pages 248–258. Springer Berlin Heidelberg, 2009.

[82] Liming Fang, Willy Susilo, Chunpeng Ge, and Jiandong Wang. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Information Science*, 238:221–241, 2013.

[83] Ana Ferreira, Ricardo Cruz-Correia, Luis Antunes, and David Chadwick. Access control: how can it improve patients' healthcare? In *Proceedings of the International Council on Medical and Care Compunetics*, ICMCC 2007, pages 182–196, 2007.

[84] Amos Fiat and Moni Naor. Broadcast encryption. In *Proceedings of the 13th Annual International Cryptology Conference*, CRYPTO 1993, pages 480–491. Springer-Verlag New York, Inc., 1994.

[85] Neil S. Fleming, Steven D. Culler, Russell McCorkle, Edmund R. Becker, and David J. Ballard. The financial and nonfinancial costs of implementing electronic health records in primary care practices. *Health Affairs*, 30(3):481–489, 2011.

[86] Alejandro Enrique Flores Zuniga, Khin Than Win, and Willy Susilo. Secure exchange of electronic health records. *User-Driven Healthcare: Concepts, Methodologies, Tools, and Applications*, pages 1403–1424, 2013.

[87] Francis. H. Roger France, Marc Bangels, and Etienne De Clercq. Purposes of health identification cards and role of a secure access platform (be-health) in belgium. *International Journal of Medical Informatics*, 76:84–88, 2007.

[88] Gerhard Frey, M. Muller, and Hans Georg Ruck. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, September 2006.

[89] Thomas Fuhr and Pascal Paillier. Decryptable searchable encryption. In *Proceedings of the First International Conference on Provable Security*, ProvSec 2007, pages 228–236, Berlin, Heidelberg, 2007. Springer.

[90] Randike Gajanayake, Renato Iannella, and Tony R. Sahama. Privacy oriented access control for electronic health records. In *Proceedings of the International World Wide Web Workshop on Data Usage Management on the Web*. ACM, 2012.

[91] Steven D. Galbraith. Supersingular curves in cryptography. In *Proceedings of the 7th International Conference on Theory and Application of Cryptology and Information Security*, ASIACRYPT 2001, pages 495–513, London, UK, 2001. Springer-Verlag.

[92] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, September 2008.

[93] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2003, pages 272–293, Berlin, Heidelberg, 2003. Springer-Verlag.

[94] Craig Gentry. Practical identity-based encryption without random oracles. In *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2006, pages 445–464, Berlin, Heidelberg, 2006. Springer-Verlag.

[95] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, TCC 2009, pages 437–456, Berlin, Heidelberg, 2009. Springer-Verlag.

[96] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Proceedings of the 8th International Conference on Theory and Application of Cryptology and Information Security*, ASIACRYPT 2002, pages 548–566. Springer Berlin Heidelberg, 2002.

[97] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *Proceedings of the International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT 2009, pages 171–188. Springer Berlin Heidelberg, 2009.

[98] Oded Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6(1):21–53, March 1993.

[99] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[100] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[101] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, STOC 1982, pages 365–377. ACM, 1982.

[102] Michael T. Goodrich, Roberto Tamassia, and Andrew Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proceedings of the DARPA Information Survivability Conference*, DISCEX 2001, pages 68–82 vol.2, 2001.

[103] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, ICALP 2008, pages 579–591, Berlin, Heidelberg, 2008. Springer-Verlag.

[104] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS 2006, pages 89–98, New York, NY, USA, 2006. ACM.

[105] Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In *Proceedings of the 5th International Conference on Applied Cryptography and Network Security*, ACNS 2007, pages 288–306, Berlin, Heidelberg, 2007. Springer-Verlag.

[106] Chunxiang Gu, Yuefei Zhu, and Heng Pan. Efficient public key encryption with keyword search schemes from pairings. In *Proceedings of the Third SKLOIS Conference on Information Security and Cryptology*, Inscrypt 2007, pages 372–383, Berlin, Heidelberg, 2008. Springer.

[107] Tracy D. Gunter and Terry Nicholas P. The emergence of national electronic health record architectures in the united states and australia: Models, costs, and questionss. *Journal of Medical Internet Research*, 7(1):e3, 2005.

[108] Fuchun Guo, Yi Mu, and Zhide Chen. Mutative identity-based signatures or dynamic credentials without random oracles. In *Proceedings of the 6th International Conference on Cryptology and Network Security*, CANS 2007, pages 1–14, Berlin, Heidelberg, 2007. Springer.

[109] Fuchun Guo, Yi Mu, and Zhide Chen. Identity-based online/offline encryption. In *Proceedings of the 12th International Conference*, volume 5143 of *FC 2008*, pages 247–261. Springer-Verlag, 2008.

[110] Fuchun Guo, Yi Mu, Willy Susilo, and Vijay Varadharajan. Membership encryption and its applications. In *Proceedings of the 18th Australasian Conference on Information Security and Privacy*, ACISP 2013, pages 219–234. Springer Berlin Heidelberg, 2013.

[111] Sebastian Haas, Sven Wohlgemuth, Isao Echizen, Noboru Sonehara, and G˙ Aspects of privacy for electronic health records. *International Journal of Medical Informatics*, 80(2):e26 – e31, 2011.

[112] Zhuo Hao, Sheng Zhong, and Nenghai Yu. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. *IEEE Transactions on Knowledge and Data Engineering*, 23(9):1432–1437, September 2011.

[113] Guy C. Hembroff and Sead Muftic. Samson: Secure access for medical smart cards over networks. In *Proceedings of the 2010 IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks*, WOWMOM 2010, pages 1–6, Washington, DC, USA, 2010. IEEE Computer Society.

[114] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[115] Susan Hohenberger and Brent Waters. Online/offline attribute-based encryption. In *Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography*, PKC 2014, pages 293–310. Springer-Verlag, 2014.

[116] Monica M. Horvath, Stephanie Winfield, Steve Evans, Steve Slopek, Howard Shang, and Jeffrey Ferranti. The deduce guided query tool: Providing simplified access to clinical data for research and quality improvement. *Journal of Biomedical Informatics*, 44(2):266 – 276, 2011.

[117] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2002, pages 466–481, Berlin, Heidelberg, 2002. Springer-Verlag.

[118] Jie Huang, Mohamed Sharaf, and Chin-Tser Huang. A hierarchical framework for secure and scalable ehr sharing and access control in multi-cloud. In *Proceedings of the 41st International Conference on Parallel Processing Workshops*, ICPPW 2012, pages 279–287, Washington, DC, USA, 2012. IEEE Computer Society.

[119] Lu-Chou Huang, Huei-Chung Chu, Chung-Yueh Lien, Chia-Hung Hsiao, and Tsair Kao. Privacy preservation and information security protection for patients' portable electronic health records. *Computers in Biology and Medicine*, 39(9):743–750, September 2009.

[120] Anca Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS 2003, 2003.

[121] Mohammad Jafari, Reihaneh Safavi-Naini, Chad Saunders, and Nicholas Paul Sheppard. Using digital rights management for securing data in a medical research environment. In *Proceedings of the 10th Annual ACM Workshop on Digital Rights Management*, DRM 2010, pages 55–60, New York, NY, USA, 2010. ACM.

[122] Mohammad Jafari, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. A rights management approach to protection of privacy in a cloud of electronic health records. In *Proceedings of the 11th Annual ACM Workshop on Digital Rights Management*, DRM 2011, pages 23–30, New York, NY, USA, 2011. ACM.

[123] Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In *Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, PKC 1999, pages 112–121, London, UK, UK, 1999. Springer-Verlag.

[124] Ik Rae Jeong, Jeong Ok Kwon, Dowon Hong, and Dong Hoon Lee. Constructing peks schemes secure against keyword guessing attacks is possible? *Computer Communications*, 32(2):394–396, February 2009.

[125] Wen-Shan Jian, Hsyien-Chia Wen, Jeremiah Scholl, Syed Abdul Shabbir, Peisan Lee, Chien-Yeh Hsu, and Yu-Chuan Li. The taiwanese method for providing patients data from multiple hospital ehr systems. *Journal of Biomedical Informatics*, 44(2):326–332, 2011.

[126] Antoine Joux and Kim Nguyen. Separating decision diffie–hellman from computational diffie–hellman in cryptographic groups. *Journal of Cryptology*, 16(4):239–247, 2003.

[127] Ari Juels and Burton S. Kaliski, Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS 2007, pages 584–597, New York, NY, USA, 2007. ACM.

[128] Stasia Kahn and Vikram Sheshadri. Medical record privacy and security in a digital environment. *IT Professional*, 10(2):46–52, 2008.

[129] Atif Khan and Ian McKillop. Privacy-centric access control for distributed heterogeneous medical information systems. In *Proceedings of the 2013 IEEE International Conference on Healthcare Informatics*, ICHI 2013, pages 297–306, 2013.

[130] M. Fahim Ferdous Khan and Ken Sakamura. Security in healthcare informatics: Design and implementation of a robust authentication and a hybrid access control mechanism. In *Proceedings of the Mosharaka International Conference on Communications, Computers and Applications*, MIC-CCA 2012, pages 159–164, 2012.

[131] Eike Kiltz. A tool box of cryptographic functions related to the diffie-hellman function. In *Proceedings of the 2nd International Conference on Cryptology in India*, INDOCRYPT 2001, pages 339–350, Berlin Heidelberg, 2001. Springer-Verlag.

[132] Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In *Proceedings of the 11th Australasian Conference on Information Security and Privacy*, ACISP 2006, pages 336–347, 2006.

[133] Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. *Theoretical Computer Science*, 410(47–49):5093–5111, 2009.

[134] Hugo Krawczyk and Tal Rabin. Chameleon hashing and signatures. Cryptology ePrint Archive, Report 1998/010, 1998. `http://eprint.iacr.org/`.

[135] Serge Lang. *Elliptic Functions*, chapter Elliptic Functions, pages 5–21. Springer, New York, NY, 1987.

[136] Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An efficient protocol for authenticated key agreement. Technical report, Designs, Codes and Cryptography, 1998.

[137] Anh Le and Athina Markopoulou. Nc-audit: Auditing for network coding storage. *CoRR*, abs/1203.1730, 2012.

[138] Edward D. Lemaire, Dan Deforge, Shawn Marshall, and Dorothyann Curran. A secure web-based approach for accessing transitional health information for people with traumatic brain injury. *Computer Methods and Programs in Biomedicine*, 81:213–219.

[139] Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2012, pages 318–335, Berlin, Heidelberg, 2012. Springer-Verlag.

[140] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2010, pages 62–91, Berlin, Heidelberg, 2010. Springer-Verlag.

[141] Allison Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. Cryptology ePrint Archive, Report 2008/309, 2008. `http://eprint.iacr.org/`.

[142] Allison Lewko and Brent Waters. Unbounded hibe and attribute-based encryption. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT 2011, pages 547–567, Berlin, Heidelberg, 2011. Springer-Verlag.

[143] Allison Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Proceedings of the 32th Annual International Cryptology Conference*, CRYPTO 2012, pages 180–198. Springer Berlin Heidelberg, 2012.

[144] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):131–143, January 2013.

[145] Kaitai Liang, Liming Fang, Duncan S. Wong, and Willy Susilo. A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security. In *Proceedings of the 5th International Conference on Intelligent Networking and Collaborative Systems*, INCoS 2013, pages 552–559, 2013.

[146] Xiaohui Liang, Zhenfu Cao, Huang Lin, and Jun Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS 2009, pages 276–286, New York, NY, USA, 2009. ACM.

[147] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In *Proceedings of the 15th International Conference on Practice and Theory in Public Key Cryptography*, PKC 2012, pages 206–224, Berlin, Heidelberg, 2012. Springer-Verlag.

[148] Chang Liu, Rajiv Ranjan, Chi Yang, Xuyun Zhang, Lizhe Wang, and Jinjun Chen. Mur-dpa: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud. *IEEE Transactions on Computers*, 64(9):2609–2622, September 2015.

[149] Song Luo, Jianbin Hu, and Zhong Chen. Ciphertext policy attribute-based proxy re-encryption. In *Proceedings of the 12th International Conference on Information and Communications Security*, ICICS 2010, pages 401–415, Berlin, Heidelberg, 2010. Springer-Verlag.

[150] Masahiro Mambo and Eiji Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E80-A(1):54–63, January 1997.

[151] Kenneth D. Mandl, Peter Szolovits, and Isaac S. Kohane. Public standards and patients control: how to keep electronic medical records accessible but private. *British Medical Journal*, 322(7281):283–287, 2001.

[152] Russell F. Martin. Group selection and key management strategies for ciphertext-policy attribute-based encryption. Master's thesis, Rochester Institute of Technology, 2013.

[153] I. Mavridis, C. Georgiadis, G. Pangalos, and M. Khair. Access control based on attribute certificates for medical intranet applications. *Journal of Medical Internet Research*, 3(1):E9, March 2001.

[154] Nir Menachemi and Taleah H. Collum. Benefits and drawbacks of electronic health record systems. *Risk Management and Healthcare Policy*, 2011(4):47–55, May 2011.

[155] Alfred J. Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, September 2006.

[156] Ralph C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, April 1978.

[157] Ralph Charles Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford, CA, USA, 1979.

[158] Silvio Micali. Novomondo: Scalable certificate validation and simplified pki management. In *Proceedings of the first Annual PKI Research Workshop*, 2002.

[159] Silvio Micali, Charles Rackoff, and Bob Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, April 1988.

[160] Victor S. Miller. Short programs for functions on curves. In *IBM Thomas J. Watson Research Center*. Unplublised manuscript, 1986.

[161] Shigeo Mitsunari, Ryuichi Sakai, and M. Kasahara. A new traitor tracing. *IEICE Transactions On Fundamentals of Electronics Communications and Computer Sciences*, E85-A(2):481–484, 2002.

[162] Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of elliptic curve traces under fr-reduction. In *Proceedings of the Third International Conference on Information Security and Cryptology*, ICISC 2000, pages 90–108, London, UK, 2001. Springer-Verlag.

[163] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for fr-reduction. *IEICE Transactions Fundamentals*, E84 A(5), May 2001.

[164] Takeo Mizuno and Hiroshi Doi. Hybrid proxy re-encryption scheme for attribute-based encryption. In *Proceedings of the 5th International Conference on Information Security and Cryptology*, Inscrypt 2009, pages 288–302. Springer-Verlag, December 2010.

[165] Gustavo H. M. B. Motta and Sérgio Shiguemi Furuie. A contextual role-based access control authorization model for electronic patient record. *IEEE Transactions on Information Technology in Biomedicine*, 7(3):202–207, 2003.

[166] Tracey L. Murray, Mona Calhoun, and Nayna C. Philipsen. Privacy, confidentiality, hipaa, and hitech: Implications for the health care practitioner. *The Journal for Nurse Practitioners*, 7(9):747–752, 2016.

[167] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proceedings of the 21th Annual International Cryptology Conference*, CRYPTO 2001, pages 41–62. Springer Berlin Heidelberg, 2001.

[168] Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium*, USENIX 1998, 1998.

[169] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, March 2004.

[170] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC 1990, pages 427–437. ACM, 1990.

[171] Shivaramakrishnan Narayan, Martin Gagné, and Reihaneh Safavi-Naini. Privacy preserving ehr system using attribute-based infrastructure. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, CCSW 2010, pages 47–52, New York, NY, USA, 2010. ACM.

[172] Thomas Neubauer and Johannes Heurix. A methodology for the pseudonymization of medical data. *International Journal of Medical Informatics*, 80(3):190 – 204, 2011.

[173] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Proceedings of the 2005 International Conference on Topics in Cryptology*, CT-RSA 2005, pages 275–292, Berlin, Heidelberg, 2005. Springer-Verlag.

[174] Tatsuaki Okamoto and David Pointcheval. React: Rapid enhanced-security asymmetric cryptosystem transform. In *Proceedings of the International Conference on Topics in Cryptology*, CT-RSA 2001, pages 159–174. Springer, Heidelberg, Berlin, 2001.

[175] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Proceedings of the 30th Annual International Cryptology Conference*, CRYPTO 2010, pages 191–208. Springer-Verlag, August 2010.

[176] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS 2007, pages 195–203, New York, NY, USA, 2007. ACM.

[177] Ronald Ostrowski. Paternity indices. In *Proceedings of the 2nd Annual Conference on Forensic Bioinformatics, Statistics and DNA Profiling*, Wright State University, 2003.

[178] Mor Peleg, Dizza Beimel, Dov Dori, and Yaron Denekamp. Situation-based access control: Privacy management via modeling of patient data access scenarios. *Journal of Biomedical Informatics*, 41(6):1028–1040, December 2008.

[179] Birgit Pfitzmann and Ahmad-Reza Sadeghi. Anonymous fingerprinting with direct non-repudiation. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT 2000, pages 401–414, London, UK, 2000. Springer-Verlag.

[180] Duong-Hieu Phan, David Pointcheval, Siamak F. Shahandashti, and Mario Strefler. Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts. In *Proceedings of the 17th Australasian Conference on Information Security and Privacy*, ACISP 2012, pages 308–321. Springer Berlin Heidelberg, 2012.

[181] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS 2006, pages 99–112, New York, NY, USA, 2006. ACM.

[182] Catherine Quantin, David-Olivier Jaquet-Chiffelle, Gouenou Coatrieux, Eric Benzenine, and François-André Allaert. Medical record search engines, using pseudonymised patient identity: An alternative to centralised medical records. *International Journal of Medical Informatics*, 80(2):e6–e11, 2011.

[183] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 1991, pages 433–444. Springer-Verlag, 1992.

[184] Gianluigi Reni, Massimo Molteni, Stefano Arlotti, and Francesco Pinciroli. Chief medical officer actions on information security in an italian rehabilitation centre. *International Journal of Medical Informatics*, 73:271–279, 2004.

[185] Fatemeh Rezaeibagha, Khin Than Win, and Willy Susilo. A systematic literature review on security and privacy of electronic health record systems: Technical perspectives. *Health Information Management Journal*, 44(3):23–38, 2015.

[186] Hyun Sook Rhee, Jong Hwan Park, and Dong Hoon Lee. Generic construction of designated tester public-key encryption with keyword search. *Information Science*, 205:93–109, November 2012.

[187] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Improved searchable public key encryption with designated tester. In *Proceedings of the 4th International Symposium on Information, Computer and Communications Security*, ASIACCS 2009, pages 376–379, New York, NY, USA, 2009. ACM.

[188] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, 83(5):763–771, May 2010.

[189] Bernahard Riedl, Thomas Neubauer, Gernot Goluch, Oswald Boehm, Gert Reinauer, and Alexander Krumboeck. A secure architecture for the pseudonymization of medical data. In *Proceedings of the 2nd International Conference on Availability, Reliability and Security*, ARES 2007, pages 318–324, April 2007.

[190] Bernhard Riedl, Veronika Grascher, Stefan Fenz, and Thomas Neubauer. Pseudonymization for improving the privacy in e-health applications. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, pages 255–255, January 2008.

[191] Bernhard Riedl, Veronika Grascher, and Thomas Neubauer. Applying a threshold scheme to the pseudonymization of health data. In *Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing*, PRDC 2007, pages 397–400, Washington, DC, USA, 2007. IEEE Computer Society.

[192] Lillian Røstad. An initial model and a discussion of access control in patient controlled health records. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security*, ARES 2008, pages 935–942, March 2008.

[193] Lillian Røstad and Ole Edsberg. A study of access control requirements for healthcare systems based on audit trails from access logs. In *Proceedings of the 22nd Annual Computer Security Applications Conference*, ACSAC 2006, pages 175–186, December 2006.

[194] Jolt Roukema, Renske K. Los, Sacha E. Bleeker, Astrid M. van Ginneken, Johan van der Lei, and Henriette A. Moll. Paper versus computer: Feasibility of an electronic medical record in general pediatrics. *Pediatrics*, 117(1):15–21, 2006.

[195] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *Proceedings of the ACM Conference on Computer and Communications Security*, CCS 2013, pages 463–474. ACM, 2013.

[196] Yannis Rouselakis and Brent Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. In *Proceedings of the 19th International Conference on Financial Cryptography and Data Security*, FC 2015, pages 315–332, Berlin, Heidelberg, 2015. Springer-Verlag.

[197] Karl Rubin and Alice Silverberg. The best and worst of supersingular abelian varieties in cryptology. Cryptology ePrint Archive, Report 2002/006, 2002. `http://eprint.iacr.org/2002/006`.

[198] Karl Rubin and Alice Silverberg. Supersingular abelian varieties in cryptology. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 2002, pages 336–353, Berlin, Heidelberg, 2002. Springer.

[199] Pekka Ruotsalainen. A cross-platform model for secure electronic health record communication. *International Journal of Medical Informatics*, 73:291–295.

[200] Pekka Ruotsalainen and Bryan Manning. A notary archive model for secure preservation and distribution of electrically signed patient documents. *International Journal of Medical Informatics*, 76(5-6):449–453, 2007.

[201] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2005, pages 457–473, Berlin, Heidelberg, 2005. Springer-Verlag.

[202] Mike Scott. Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/2002/164`.

[203] MIRACL CRYPTO SDK. `https://certivox.com/solutions/miracl-crypto-sdk/`.

[204] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In *Proceedings of the International Conference on Public Key Cryptography*, PKC 2007, pages 166–180. Springer Berlin Heidelberg, 2007.

[205] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT 2008, pages 90–107, Berlin, Heidelberg, 2008. Springer-Verlag.

[206] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of the 4th Annual International Cryptology Conference*, CRYPTO 1984, pages 47–53, New York, NY, USA, 1985. Springer-Verlag.

[207] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In *Proceedings of the 21st Annual International Cryptology Conference*, CRYPTO 2001, pages 355–367. Springer-Verlag, 2001.

[208] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT 1997, pages 256–266, Berlin, Heidelberg, 1997. Springer-Verlag.

[209] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*. Springer-Verlag New York, 1986.

[210] Ramkinker Singh, Vipra Gupta, and Mohan K. Dynamic federation in identity management for securing and sharing personal health records in a patientcentric model in cloud. *Engg Journals Publications*, pages 2201–2209, 2013.

[211] Dean F. Sittig and Hardeep Singh. Legal, ethical, and financial dilemmas in electronic health record adoption and use. *Pediatrics*, 127(4):1042–1047, 2011.

[212] Lindi A. Slevin and Alex Macfie. Role based access control for a medical database. In *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications*, SEA 2007, pages 226–233, Anaheim, CA, USA, 2007. ACTA Press.

[213] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, CCS 1996, pages 31–37. ACM, 1996.

[214] Snezana Sucurovic. Implementing security in a distributed web-based ehcr. *International Journal of Medical Informatics*, 76(5-6):491–496, 2007.

[215] Snezana Sucurovic. An approach to access control in electronic health record. *Journal of Medical Systems*, 34(4):659–666, 2010.

[216] Jinyuan Sun and Yuguang Fang. Cross-domain data sharing in distributed electronic health record systems. *IEEE Transactions on Parallel and Distributed Systems*, 21(6):754–764, June 2010.

[217] Chul Sur, Chae Duk Jung, and Kyung-Hyune Rhee. Multi-receiver certificate-based encryption and application to public key broadcast encryption. In *Proceedings of the ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security*, BLISS 2007, pages 35–40. IEEE Computer Society, 2007.

[218] Manolis Tsiknakis, Dimitrios G. Katehakis, and Stelios C. Orphanoudakis. A health information infrastructure enabling secure access to the life-long multimedia electronic health record. In *Proceedings of the 18th International Congress and Exhibition on Computer Assisted Radiology and Surgery*, CARS 2004, pages 289–294, 2004.

[219] Frank Ueckert, Michael Goerz, Maximilian Ataian, Sven Tessmann, and Hans-Ulrich Prokosch. Empowerment of patients and communication with health care professionals through an electronic health record. *International Journal of Medical Informatics*, 70(2–3):99–108, 2003.

[220] Frank K. Ueckert and Hans-Ulrich Prokosch. Implementing security and access control mechanisms for an electronic healthcare record. *Proceedings of the Annual AMIA Symposium*, page 825829, 2002.

[221] Minne van der Haak, Astrid Corinna Wolff, Ralf Brandner, Peter Drings, Michael Wannenmacher, and Thomas Wetter. Data security and protection in cross-institutional electronic patient records. *International Journal of Medical Informatics*, 70(23):117 – 130, 2003.

[222] Helma van der Linden, Dipak Kalra, Arie Hasman, and Jan Talmon. Inter-organizational future proof ehr systems: A review of the security and privacy related issues. *International Journal of Medical Informatics*, 78(3):141–160, 2009.

[223] Eric R. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. *Journal of Cryptology*, 17(4):277–296, September 2004.

[224] Boyang Wang, Baochun Li, and Hui Li. Knox: Privacy-preserving auditing for shared data with large groups in the cloud. In *Proceedings of the 10th International Conference on Applied Cryptography and Network Security*, ACNS 2012, pages 507–525, Berlin, Heidelberg, 2012. Springer-Verlag.

[225] Boyang Wang, Baochun Li, and Hui Li. Oruta: privacy-preserving public auditing for shared data in the cloud. *IEEE Transactions on Cloud Computing*, 2(1):43–56, 2012.

[226] Boyang Wang, Baochun Li, and Hui Li. Panda: Public auditing for shared data with efficient user revocation in the cloud. *IEEE Transactions on Services Computing*, 8(1):92–106, 2015.

[227] Cong Wang, Sherman S.M. Chow, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for secure cloud storage. *IEEE Transactions on Computers*, 62(2):362–375, 2013.

[228] Cong Wang, Qian Wang, Kui Ren, Ning Cao, and Wenjing Lou. Toward secure and dependable storage services in cloud computing. *IEEE Trans. Serv. Comput.*, 5(2):220–232, 2012.

[229] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Ensuring data storage security in cloud computing. In *Proceedings of the 17th International Workshop on Quality of Service*, IWQoS 2009, 2009.

[230] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *Proceedings of the 29th Conference on Information Communications*, INFOCOM 2010, pages 525–533. IEEE Press, 2010.

[231] Huaqun Wang, Qianhong Wu, Bo Qin, and Josep Domingo-Ferrer. Frr: Fair remote retrieval of outsourced private medical records in electronic health networks. *Journal of Biomedical Informatics*, 50:226 – 233, 2014.

[232] Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. In *Proceedings of the 14th European Conference on Research in Computer Security*, ESORICS 2009, pages 355–370, Berlin, Heidelberg, 2009. Springer-Verlag.

[233] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(5):847–859, May 2011.

[234] Samuel J. Wang, Blackford Middleton, Lisa A. Prosser, Christiana G. Bardon, Cynthia D. Spurr, Patricia J. Carchidi, Anne F. Kittler, Robert C. Goldszer, David G. Fairchild, Andrew J. Sussman, Gilad J. Kuperman, and David W. Bates. A cost-benefit analysis of electronic medical records in primary care. *The American Journal of Medicine*, 114(5):397–403, 2003.

[235] Brent Waters. Efficient identity-based encryption without random oracles. In *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT 2005, pages 114–127, Berlin, Heidelberg, 2005. Springer.

[236] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *Proceedings of the 29th Annual International Cryptology Conference*, CRYPTO 2009, pages 619–636. Springer Berlin Heidelberg, 2009.

[237] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Proceedings of the Conference on Public Key Cryptography*, PKC 2011, pages 53–70, Berlin, Heidelberg, 2011. Springer-Verlag.

[238] Khin Than Win, Willy Susilo, and Yi Mu. Personal health record systems and their security protection. *Journal of Medical Systems*, 30(4):309–315, 2006.

[239] Stefan Wolf. *Information-Theoretically and Computionally Secure Key Agreement in Cryptography*. PhD thesis, ETH Zurich, 1999.

[240] Ruoyu Wu, Gail-Joon Ahn, and Hongxin Hu. Secure sharing of electronic health records in clouds. In *Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, CollaborateCom 2012, pages 711–718, 2012.

[241] Kan Yang and Xiaohua Jia. Data storage auditing service in cloud computing: Challenges, methods and opportunities. *World Wide Web*, 15(4):409–428, July 2012.

[242] Wei-Chuen Yau, Raphael C. W. Phan, Swee-Huay Heng, and Bok-Min Goi. Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. *International Journal of Computer Mathematics (Advanced Computer Mathematics based Cryptography and Security Technologies)*, 90(12):2581–2587, December 2013.

[243] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the 29th Conference on Information Communications*, INFOCOM 2010, pages 534–542. IEEE Press, 2010.

[244] Weider D. Yu and Mark A. Chekhanovskiy. An electronic health record content protection system using smartcard and pmr. In *Proceedings of the 9th International Conference on e-Health Networking, Application and Services*, pages 11–18, June 2007.

[245] Yong Yu, Man Ho Au, Yi Mu, Shaohua Tang, Jian Ren, Willy Susilo, and Liju Dong. Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage. *International Journal of Information Security*, pages 1–12, 2014.

[246] Yong Yu, Jianbing Ni, Man Ho Au, Yi Mu, Boyang Wang, and Hui Li. On the security of a public auditing mechanism for shared cloud data service. *IEEE Transactions on Services Computing*, PP(99):1–1, 2014.

[247] Yong Yu, Lei Niu, Guomin Yang, Yi Mu, and Willy Susilo. On the security of auditing mechanisms for secure cloud storage. *International Journal of Grid Computing on Future Generation Computer Systems (theory, methods and applications)*, 30(1):127–132, 2014.

[248] Rui Zhang and Ling Liu. Security models and requirements for healthcare application clouds. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing*, CLOUD 2010, pages 268–275, Washington, DC, USA, 2010. IEEE Computer Society.

[249] Rui Zhang, Ling Liu, and Rui Xue. Role-based and time-bound access and management of ehr data. *Security and Communication Networks*, 7(6):994–1015, 2014.

[250] Lidong Zhou, Michael A. Marsh, Fred B. Schneider, and Anna Redz. Distributed blinding for distributed elgamal re-encryption. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, ICDCS 2005, page 815824. IEEE Computer Society, June 2005.

[251] Yan Zhu, Gail-Joon Ahn, Hongxin Hu, Stephen S. Yau, Ho G. An, and Chang-Jun Hu. Dynamic audit services for outsourced storages in clouds. *IEEE Transactions on Services Computing*, 6(2):227–238, 2013.

[252] Yan Zhu, Huaixi Wang, Zexing Hu, Gail-Joon Ahn, Hongxin Hu, and Stephen S. Yau. Dynamic audit services for integrity verification of outsourced storages in clouds. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC 2011, pages 1550–1557, New York, NY, USA, 2011. ACM.