Modular Arithmetic for PQC Implementations

Thomas Plantard

Nokia Bell Labs

https://thomas-plantard.github.io
thomas.plantard@nokia-bell-labs.com

Deployment of post-quantum cryptography
IHP Paris 2024

Outline

Outline

Construction - Cryptographer

Create a link between:

1. one hard to compute function
2. one easy to compute function

Cryptography

Construction - Cryptographer

Create a link between:

1. one hard to compute function RSA : Factorization
2. one easy to compute function RSA : Modular Exponentiation

Cryptography

Construction - Cryptographer

Create a link between:

1. one hard to compute function RSA : Factorization
2. one easy to compute function RSA : Modular Exponentiation

Analyze - Cryptanalyst

Find better way to compute the hard function

Cryptography

Construction - Cryptographer

Create a link between:

1. one hard to compute function RSA : Factorization
2. one easy to compute function RSA : Modular Exponentiation

Analyze - Cryptanalyst

Find better way to compute the hard function

Optimization - Computer Arithmetician

Find better way to compute the easy to compute function

Structure in cryptography

Structure in cryptography

Two type of structures

1. Identifiable: Able to check in efficiently way
2. Hidden: As hard to identify as to solve security problem

Structure in cryptography

Two type of structures

1. Identifiable: Able to check in efficiently way
2. Hidden: As hard to identify as to solve security problem

Example

- Code: Quasicyclic
- Multivariate: Oil/Vinegar Structure
- lattice based: Ideal Lattice — Diagonal Dominant
- LWE: Ring LWE — Module LWE

Structure in cryptography

Two type of structures

1. Identifiable: Able to check in efficiently way
2. Hidden: As hard to identify as to solve security problem

Example

- Code: Quasicyclic
- Multivariate: Oil/Vinegar Structure
- lattice based: Ideal Lattice — Diagonal Dominant
- LWE: Ring LWE — Module LWE

Advantage of Identifiable Structure

Advantage of Identifiable Structure

Memory - Compact object

Often a Polynomial Structure instead of a Matrix
Example: Quasicyclic

Advantage of Identifiable Structure

Memory - Compact object

Often a Polynomial Structure instead of a Matrix
Example: Quasicyclic

Speed - Operating on object more efficiently

Polynomial Multiplication faster than Vector-Matrix
Example: Module LWE

Advantage of Identifiable Structure

Memory - Compact object

Often a Polynomial Structure instead of a Matrix
Example: Quasicyclic

Speed - Operating on object more efficiently

Polynomial Multiplication faster than Vector-Matrix
Example: Module LWE

Functionality

Allow to perform new operation
Example: Ideal Lattice used by Gentry for first FHE Scheme

Advantage of Identifiable Structure

Memory - Compact object

Often a Polynomial Structure instead of a Matrix
Example: Quasicyclic

Speed - Operating on object more efficiently

Polynomial Multiplication faster than Vector-Matrix
Example: Module LWE

Functionality

Allow to perform new operation
Example: Ideal Lattice used by Gentry for first FHE Scheme

Security - Weaker Problem

Often unstudied problem

Outline

Kyber

Kyber

Q-Ary Lattice

$$\mathcal{L} = \begin{pmatrix} qId & 0 \\ A & Id \end{pmatrix} \text{ with } q = n^{O(1)}$$

Kyber

Q-Ary Lattice

$$\mathcal{L} = \begin{pmatrix} qId & 0 \\ A & Id \end{pmatrix} \text{ with } q = n^{O(1)}$$

Module Lattice

A composed of Toeplitz matrix block generated by polynomial

Kyber

Q-Ary Lattice

$$\mathcal{L} = \begin{pmatrix} qId & 0 \\ A & Id \end{pmatrix} \text{ with } q = n^{O(1)}$$

Module Lattice

A composed of Toeplitz matrix block generated by polynomial

With a Cyclotomic Polynomial Φ

A composed of Negacyclic block

Kyber

Q-Ary Lattice

$$\mathcal{L} = \begin{pmatrix} qId & 0 \\ A & Id \end{pmatrix} \text{ with } q = n^{O(1)}$$

Module Lattice

A composed of Toeplitz matrix block generated by polynomial

With a Cyclotomic Polynomial Φ

A composed of Negacyclic block

Φ Smooth in $\mathbb{F}_q$

$$\Phi = (X - \gamma_0) \dots (X - \gamma_n) \bmod q$$

Kyber

Q-Ary Lattice

$$\mathcal{L} = \begin{pmatrix} qId & 0 \\ A & Id \end{pmatrix} \text{ with } q = n^{O(1)}$$

SPEED/MEMORY

No Multi Precision Arithmetic

Module Lattice

A composed of Toeplitz matrix block generated by polynomial

With a Cyclotomic Polynomial Φ

A composed of Negacyclic block

Φ Smooth in $\mathbb{F}_q$

$$\Phi = (X - \gamma_0)\dots(X - \gamma_n) \bmod q$$

Kyber

Q-Ary Lattice

$$\mathcal{L} = \begin{pmatrix} qId & 0 \\ A & Id \end{pmatrix} \text{ with } q = n^{O(1)}$$

Module Lattice

A composed of Toeplitz matrix block generated by polynomial

With a Cyclotomic Polynomial Φ

A composed of Negacyclic block

Φ Smooth in $\mathbb{F}_q$

$$\Phi = (X - \gamma_0) \dots (X - \gamma_n) \bmod q$$

SPEED/MEMORY

No Multi Precision Arithmetic

MEMORY/SPEED

Polynomial Multiplication faster than Matrix Multiplication

Kyber

Q-Ary Lattice

$$\mathcal{L} = \begin{pmatrix} qId & 0 \\ A & Id \end{pmatrix} \text{ with } q = n^{O(1)}$$

SPEED/MEMORY

No Multi Precision Arithmetic

Module Lattice

A composed of Toeplitz matrix block generated by polynomial

MEMORY/SPEED

Polynomial Multiplication faster than Matrix Multiplication

With a Cyclotomic Polynomial Φ

A composed of Negacyclic block

MEMORY

Less Expansion lead smaller q

Φ Smooth in $\mathbb{F}_q$

$$\Phi = (X - \gamma_0) \ldots (X - \gamma_n) \bmod q$$

Kyber

Q-Ary Lattice

$\mathcal{L} = \begin{pmatrix} qId & 0 \\ A & Id \end{pmatrix}$ with $q = n^{O(1)}$

SPEED/MEMORY

No Multi Precision Arithmetic

Module Lattice

A composed of Toeplitz matrix block generated by polynomial

MEMORY/SPEED

Polynomial Multiplication faster than Matrix Multiplication

With a Cyclotomic Polynomial Φ

A composed of Negacyclic block

MEMORY

Less Expansion lead smaller q

Φ Smooth in $\mathbb{F}_q$

$\Phi = (X - \gamma_0) \ldots (X - \gamma_n) \bmod q$

SPEED

Polynomial Multiplication using NTT in $O(n \log n)$

Complexity

Complexity

Polynomial Multiplication $A \times B$

- Schoolbook, $O(n^2)$
- Karatsuba, $O(n^{1.585})$
- Toom-Cook $O(n^{1.465})\ldots$
- NTT $O(n \log n)$

Complexity

Polynomial Multiplication $A \times B$

- Schoolbook, $O(n^2)$
- Karatsuba, $O(n^{1.585})$
- Toom-Cook $O(n^{1.465})\dots$
- NTT $O(n \log n)$

$A \times B \bmod \Phi$

Naturally included with NTT

Φ smooth in F_q

Φ smooth in F_q

Polynomial Multiplication

- $A \times B \bmod \Phi$
- with $\Phi = (X - \gamma_1)(X - \gamma_2)\ldots(X - \gamma_n) \bmod q$

Φ smooth in F_q

Polynomial Multiplication

- $A \times B \bmod \Phi$
- with $\Phi = (X - \gamma_1)(X - \gamma_2)\ldots(X - \gamma_n) \bmod q$

Polynomial Multiplication using NTT

1. $\tilde{A} \leftarrow NTT(A)$ with $\tilde{A} = [A(\gamma_1), A(\gamma_2), \ldots, A(\gamma_n)]$
2. $\tilde{B} \leftarrow NTT(B)$ with $\tilde{B} = [B(\gamma_1), B(\gamma_2), \ldots, B(\gamma_n)]$
3. $\tilde{C} \leftarrow \tilde{A}.\tilde{B}$ with $\tilde{C} = [A(\gamma_1)B(\gamma_1), A(\gamma_2)B(\gamma_2), \ldots, A(\gamma_n)B(\gamma_n)]$
4. $C \leftarrow NTT^{-1}(\tilde{C})$

Φ not friable in F_q

Φ not friable in F_q

Find Q

- $Q \geq nq^2$
- $\Phi = (X - \gamma_1)(X - \gamma_2)\dots(X - \gamma_n) \bmod Q$

Φ not friable in F_q

Find Q

- $Q \geq nq^2$
- $\Phi = (X - \gamma_1)(X - \gamma_2) \dots (X - \gamma_n) \bmod Q$

Polynomial Multiplication using NTT

1. $\tilde{A} \leftarrow NTT(A)$ with $\tilde{A} = [A(\gamma_1), A(\gamma_2), \dots, A(\gamma_n)]$
2. $\tilde{B} \leftarrow NTT(B)$ with $\tilde{B} = [B(\gamma_1), B(\gamma_2), \dots, B(\gamma_n)]$
3. $\tilde{C} \leftarrow A$ with $\tilde{C} = [A(\gamma_1)B(\gamma_1), A(\gamma_2)B(\gamma_2), \dots, A(\gamma_n)B(\gamma_n)]$
4. $C \leftarrow NTT^{-1}(C)$
5. $C \leftarrow C \bmod q$

Conclusion

Gain of Φ Smooth in $\mathbb{F}_q$

- NTT on $\mathbb{F}_q$ instead of NTT on $\mathbb{F}_Q$ with $Q = nq^2$
- $q \lesssim$ 12bits and $Q \lesssim$ 30 bits
- For Kyber: need to operate on 64bits instead of 32bits

Conclusion

Gain of Φ Smooth in $\mathbb{F}_q$

- NTT on $\mathbb{F}_q$ instead of NTT on $\mathbb{F}_Q$ with $Q = nq^2$
- $q \lesssim$ 12bits and $Q \lesssim$ 30 bits
- For Kyber: need to operate on 64bits instead of 32bits

Structure

- Structure have an **unknown cost**
- Gain need to be at least **significant**
- With no other solutions

Outline

Main Problematic

Modular Multiplication

- Compute
$$AB \bmod P = AB - \left\lfloor \frac{AB}{P} \right\rfloor P.$$
- A, B Input Data
- P Input Data on which **Precomputation** is possible.

Main Problematic

Modular Multiplication

- Compute
$$AB \bmod P = AB - \left\lfloor \frac{AB}{P} \right\rfloor P.$$
- A, B Input Data
- P Input Data on which **Precomputation** is possible.

Constraints

- Size of moduli: form 10bits to 10000bits
- Form of Moduli
- Side-Channel resistance
- Speed vs Memory

Notation

Compute $AB \bmod P$

- with $0 \le A, B < P$
- with $P < 2^n$

Notation

Compute $AB \bmod P$

- with $0 \leq A, B < P$
- with $P < 2^n$

Notation for $2n-$register

1. $[A]_k$ the k lowest bit of A i.e.

$$[A]_k = A \bmod 2^k$$

2. $[A]^k$ the k highest bit of A i.e.

$$[A]_k = \left\lfloor \frac{A}{2^{2n-k}} \right\rfloor$$

Outline

Barrett 1986

General Methodology

1. an integer multiplication is performed, followed by
2. approximation of the euclidean division quotient, followed by
3. some final corrections are performed.

Barrett 1986

General Methodology

1. an integer multiplication is performed, followed by
2. approximation of the euclidean division quotient, followed by
3. some final corrections are performed.

Barrett Modular Multiplication

Input : $0 \leq A, B < P$ and $R = \lfloor \frac{2^{2n}}{P} \rfloor$
Output: $C = AB \bmod P$
begin
    $C \leftarrow AB$
    $Q \leftarrow [[C]^{n-1}R]^n$
    $C \leftarrow C - QP$
    if $\underline{C \geq P}$ **then** $C \leftarrow C - P$;
    if $\underline{C \geq P}$ **then** $C \leftarrow C - P$;
end

Barrett

Example

- Constant: $P = 8461$
- Precomputation: $R = \left\lfloor \frac{10^8}{8461} \right\rfloor = 11818$
- Input $A = 6932$ $B = 4121$

Barrett

Example

- Constant: $P = 8461$
- Precomputation: $R = \left\lfloor \frac{10^8}{8461} \right\rfloor = 11818$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$

Barrett

Example

- Constant: $P = 8461$
- Precomputation: $R = \left\lfloor \frac{10^8}{8461} \right\rfloor = 11818$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[C]^4 R = 2856 * 11818 = 33752208$

Barrett

Example

- Constant: $P = 8461$
- Precomputation: $R = \left\lfloor \frac{10^8}{8461} \right\rfloor = 11818$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[C]^4 R = 2856 * 11818 = 33752208$
3. $C - QP = 28566772 - 3375 * 8461$

Barrett

Example

- Constant: $P = 8461$
- Precomputation: $R = \left\lfloor \frac{10^8}{8461} \right\rfloor = 11818$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[C]^4 R = 2856 * 11818 = 33752208$
3. $C - QP = 28566772 - 3375 * 8461 = 28566772 - 28555875 = 10897$

Barrett

Example

- Constant: $P = 8461$
- Precomputation: $R = \left\lfloor \frac{10^8}{8461} \right\rfloor = 11818$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[C]^4 R = 2856 * 11818 = 33752208$
3. $C - QP = 28566772 - 3375 * 8461 = 28566772 - 28555875 = 10897$
4. $10897 - 8461 = 2436$

Barrett

Advantage

- Avoid division
- Only requires cheap precomputation
- Final Correction can be omitted using redundancy

Barrett

Advantage

- Avoid division
- Only requires cheap precomputation
- Final Correction can be omitted using redundancy

Drawback

There is better

Outline

Montgomery

General Methodology

1. Eliminate Lower bits instead of higher bits
2. Then shift to avoid division

Montgomery

General Methodology

1. Eliminate Lower bits instead of higher bits
2. Then shift to avoid division

Montgomery Modular Multiplication

Input : $R = (-P^{-1}) \bmod 2^n$
Output: $C = AB2^{-n} \bmod P$
begin
  $C \leftarrow AB$
  $Q \leftarrow [CR]_n$
  $C \leftarrow [C + QP]^n$
  if $\underline{C \geq P}$ **then** $C \leftarrow C - P$;
end

Montgomery

Example

- Constant: $P = 8461$
- Precomputation: $R = -P^{-1} \bmod 10^4 = 2859$
- Input $A = 6932$ $B = 4121$

Montgomery

Example

- Constant: $P = 8461$
- Precomputation: $R = -P^{-1} \bmod 10^4 = 2859$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$

Montgomery

Example

- Constant: $P = 8461$
- Precomputation: $R = -P^{-1} \bmod 10^4 = 2859$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $Q = [CR]_4 = [6772 * 2859] = [19361148]_4$

Montgomery

Example

- Constant: $P = 8461$
- Precomputation: $R = -P^{-1} \bmod 10^4 = 2859$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $Q = [CR]_4 = [6772 * 2859] = [19361148]_4$
3. $C + QP = 28566772 + 1148 * 8461 = 38280000$

Montgomery

Example

- Constant: $P = 8461$
- Precomputation: $R = -P^{-1} \bmod 10^4 = 2859$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $Q = [CR]_4 = [6772 * 2859] = [19361148]_4$
3. $C + QP = 28566772 + 1148 * 8461 = 38280000$
4. $C = 3828$

Montgomery Representation

Shifted output of Montgomery algorithm

- Indeed $AB2^{-n} \bmod P$ instead of $AB \bmod P$
- Return 3828 instead of 2436
- $3828 * 10^4 \bmod P = 38280000 \bmod 8461 = 2436$

Montgomery Representation

Shifted output of Montgomery algorithm

- Indeed $AB{\color{red}2^{-n}} \bmod P$ instead of $AB \bmod P$
- Return 3828 instead of 2436
- $3828 * 10^4 \bmod P = 38280000 \bmod 8461 = 2436$

Montgomery representation, $\overline{A} = A2^n \bmod P$

$$
\begin{aligned}
Mont(\overline{A}, \overline{B}) &= \overline{A} * \overline{B} * 2^{-n} \bmod P \\
&= (A2^n) * (B2^n) * 2^{-n} \bmod P \\
&= AB2^n \bmod P \\
&= \overline{AB}
\end{aligned}
$$

Outline

Modular Multiplication for Word Size moduli - 2021

Idea

- Eliminate $2n$-bits lower bits.
- Compute $AB(-2^{-2n}) \bmod P$ even if $AB < 2^{2n}$

Modular Multiplication for Word Size moduli - 2021

Idea

- Eliminate $2n$-bits lower bits.
- Compute $AB(-2^{-2n}) \bmod P$ even if $AB < 2^{2n}$

Word Size Modular Multiplication

Input : $R = P^{-1} \bmod 2^{2n}$
Output: $C = AB(-2^{-2n}) \bmod P$
begin
  $C \leftarrow [(([[ABR]_{2n}]^n + 1)P]^n$
  if $\underline{C = P}$ **then** $C \leftarrow 0$;
end

Example

- Constant: $P = 8461$
- Precomputation: $R = P^{-1} \bmod 10^8 = 40787141$
- Input $A = 6932$ $B = 4121$

Example

- Constant: $P = 8461$
- Precomputation: $R = P^{-1} \bmod 10^8 = 40787141$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$

Example

- Constant: $P = 8461$
- Precomputation: $R = P^{-1} \bmod 10^8 = 40787141$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[CR]_8 = [28566772 * 40787141]_8 = [1165156957478852]_8$

Example

- Constant: $P = 8461$
- Precomputation: $R = P^{-1} \bmod 10^8 = 40787141$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[CR]_8 = [28566772 * 40787141]_8 = [1165156957478852]_8$
3. $[CR]_8 = 57478852$ [For Free]

Example

- Constant: $P = 8461$
- Precomputation: $R = P^{-1} \bmod 10^8 = 40787141$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[CR]_8 = [28566772 * 40787141]_8 = [1165156957478852]_8$
3. $[CR]_8 = 57478852$ [For Free]
4. $Q = [[CR]_8]^4 + 1 = 5747 + 1 = 5748$

Example

- Constant: $P = 8461$
- Precomputation: $R = P^{-1} \bmod 10^8 = 40787141$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[CR]_8 = [28566772 * 40787141]_8 = [1165156957478852]_8$
3. $[CR]_8 = 57478852$ [For Free]
4. $Q = [[CR]_8]^4 + 1 = 5747 + 1 = 5748$
5. $QP = 5748 * 8461 = 48633828$ [No more C]

Example

- Constant: $P = 8461$
- Precomputation: $R = P^{-1} \bmod 10^8 = 40787141$
- Input $A = 6932$ $B = 4121$

Computation

1. $C \leftarrow A * B = 28566772$
2. $[CR]_8 = [28566772 * 40787141]_8 = [1165156957478852]_8$
3. $[CR]_8 = 57478852$ [For Free]
4. $Q = [[CR]_8]^4 + 1 = 5747 + 1 = 5748$
5. $QP = 5748 * 8461 = 48633828$ [No more C]
6. $C = 4863$

Require Representation

Montgomery's like Representation

1. Return $AB(-2^{-2n}) \bmod P$ instead of $AB \bmod P$
2. $4863 * (-10^8) \bmod 8461 = 2436$
3. Same type of representation than Montgomery

Modular Multiplication for Word Size moduli - 2021

Advantage

1. Small Number of Operation with Register shift only
2. Minimal Redundancy: No correction/IF result always between $[0, P]$
3. One register only

Modular Multiplication for Word Size moduli - 2021

Advantage

1. Small Number of Operation with Register shift only
2. Minimal Redundancy: No correction/IF result always between $[0, P]$
3. One register only

Faster Multiplication by a constant

- for ABR, ompute only one time BR for A_0BR and A_1BR
- Gain for multiplication by "constant": 2 MUL instead of 3
- Sometime naturally done by compiler

Modular Multiplication for Word Size moduli - 2021

Advantage

1. Small Number of Operation with Register shift only
2. Minimal Redundancy: No correction/IF result always between $[0, P]$
3. One register only

Faster Multiplication by a constant

- for ABR, ompute only one time BR for A_0BR and A_1BR
- Gain for multiplication by "constant": 2 MUL instead of 3
- Sometime naturally done by compiler

Word Size

$P \lesssim 0.6180 \times 2^n$ on a $2n$-processor. Example: 31.3bits on 64bits

Outline

Number systems

Positional number system with radix β

$$X = \sum_{i=0}^{n-1} x_i \beta^i \qquad \text{with } x_i \in \{0, ...\beta - 1\}$$

Example: $X = (1315)_{10} = (2, 4, 4, 3)_8 = 3 + 4 \times 8 + 4 \times 8^2 + 2 \times 8^3$

Number systems

Positional number system with radix β

$$X = \sum_{i=0}^{n-1} x_i \beta^i \qquad \text{with } x_i \in \{0, ... \beta - 1\}$$

Example: $X = (1315)_{10} = (2, 4, 4, 3)_8 = 3 + 4 \times 8 + 4 \times 8^2 + 2 \times 8^3$

Modular number system **MNS**(p, n, γ, ρ)

$$X = \sum_{i=0}^{n-1} x_i \gamma^i \bmod P \qquad \text{with } x_i \in \{0, \ldots, \rho - 1\}$$

Example

- $MNS(p = 17, n = 3, \gamma = 7, \rho = 3)$
- $a = \sum_{i=0}^{2} x_i 7^i \bmod 17$ with $a_i \in \{0, 1, 2\}$

0	1	2	3	4

5	6	7	8	9

10	11	12	13	14

15	16

Example

- $MNS(p = 17, n = 3, \gamma = 7, \rho = 3)$
- $a = \sum_{i=0}^{2} x_i 7^i$ mod 17 with $a_i \in \{0, 1, 2\}$

0	1	2	3	4
0				
5	6	7	8	9
10	11	12	13	14
15	16			

Example

- $MNS(p = 17, n = 3, \gamma = 7, \rho = 3)$
- $a = \sum_{i=0}^{2} x_i 7^i$ mod 17 with $a_i \in \{0, 1, 2\}$

0	1	2	3	4
0	1			
5	6	7	8	9
10	11	12	13	14
15	16			

Example

- $MNS(p = 17, n = 3, \gamma = 7, \rho = 3)$
- $a = \sum_{i=0}^{2} x_i 7^i \bmod 17$ with $a_i \in \{0, 1, 2\}$

0	1	2	3	4
0	1	2		

5	6	7	8	9

10	11	12	13	14

15	16

Example

- $MNS(p = 17, n = 3, \gamma = 7, \rho = 3)$
- $a = \sum_{i=0}^{2} x_i 7^i \bmod 17$ with $a_i \in \{0, 1, 2\}$

0	1	2	3	4
0	1	2		
5	6	7	8	9
		X	$X+1$	$X+2$
10	11	12	13	14
15	16			

Example

- $MNS(p = 17, n = 3, \gamma = 7, \rho = 3)$
- $a = \sum_{i=0}^{2} x_i 7^i$ mod 17 with $a_i \in \{0, 1, 2\}$

0	1	2	3	4
0	1	2		
5	6	7	8	9
		X	$X+1$	$X+2$
10	11	12	13	14
				$2X$
15	16			
$2X+1$	$2X+2$			

Example

- $MNS(p = 17, n = 3, \gamma = 7, \rho = 3)$
- $a = \sum_{i=0}^{2} x_i 7^i \bmod 17$ with $a_i \in \{0, 1, 2\}$

0	1	2	3	4
0	1	2		
5	6	7	8	9
$X^2 + X$	$X^2 + X + 1$	X	$X + 1$	$X + 2$
10	11	12	13	14
		$X^2 + 2X$	$X^2 + 2X + 1$	$2X$
15	16			
$2X + 1$	$2X + 2$			

Example

- $MNS(p = 17, n = 3, \gamma = 7, \rho = 3)$
- $a = \sum_{i=0}^{2} x_i 7^i$ mod 17 with $a_i \in \{0, 1, 2\}$

0	1	2	3	4
0	1	2	$2X^2 + X$	$2X^2 + X + 1$
5	6	7	8	9
$X^2 + X$	$X^2 + X + 1$	X	$X + 1$	$X + 2$
10	11	12	13	14
$2X^2 + 2X$	$2X^2 + 2X + 1$	$X^2 + 2X$	$X^2 + 2X + 1$	$2X$
15	16			
$2X + 1$	$2X + 2$			

How find a “good” Modular Number System?

What do we need?

1. Compact: A PMNS where ρ is small (about $\rho \sim p^{1/n}$)
2. Efficient: “fast” arithmetic on the MNS

How find a “good” Modular Number System?

What do we need?

1. Compact: A PMNS where ρ is small (about $\rho \sim p^{1/n}$)
2. Efficient: “fast” arithmetic on the MNS

Polynomial MNS

A modular number system (p, n, γ, ρ) with a **nice polynomial** E of degree n such that

1. E is irreducible over $\mathbb{Q}$,
2. $E(\gamma) = 0 \bmod P$

How find a “good” Modular Number System?

What do we need?

1. Compact: A PMNS where ρ is small (about $\rho \sim p^{1/n}$)
2. Efficient: “fast” arithmetic on the MNS

Polynomial MNS

A modular number system (p, n, γ, ρ) with a **nice polynomial** E of degree n such that

1. E is irreducible over $\mathbb{Q}$,
2. $E(\gamma) = 0 \bmod P$

and $A \times B \to AB \bmod E$ can be done efficiently

Existing Solutions

Option for E

a) power of 2 cyclotomic polynomial i.e. $X^{2^k} + 1$,
b) polynomial of the form $X^n - c$,
c) polynomial of the form $X^n - aX - b$,
d) trinomial $X^n \pm X^k \pm 1$ with $k \leq \frac{n}{2}$,
e) quadrinomial $X^n \pm X^k \pm X^l \pm 1$ with $l < k \leq \frac{n}{2}$.

Existing Solutions

Option for E

a) power of 2 cyclotomic polynomial i.e. $X^{2^k} + 1$,
b) polynomial of the form $X^n - c$,
c) polynomial of the form $X^n - aX - b$,
d) trinomial $X^n \pm X^k \pm 1$ with $k \leq \frac{n}{2}$,
e) quadrinomial $X^n \pm X^k \pm X^l \pm 1$ with $l < k \leq \frac{n}{2}$.

Existence Issue

No guarantee, there exists a polynomial both **irreducible** and with a **root** in $\mathbb{F}_p$ for a given p.

Binomial $aX^n - b$

For **any prime** p and $n \in \mathbb{N}_{\geq 2}$, there exists a binomial $E = aX^n - b$ with

1) E is irreducible over $\mathbb{Q}$,
2) E have at least one root, γ, in $\mathbb{F}_p$,
3) $1 \leq a < b \leq \frac{57}{34} n$

Binomial $aX^n - b$

For **any prime** p and $n \in \mathbb{N}_{\geq 2}$, there exists a binomial $E = aX^n - b$ with

1) E is irreducible over $\mathbb{Q}$,
2) E have at least one root, γ, in $\mathbb{F}_p$,
3) $1 \leq a < b \leq \frac{57}{34} n$

Example

- $P = 8461$, $n = 4$
- $\frac{57}{34} n \sim 6.71$
- There exist an irreducible polynomial $aX^4 - b$ with $1 \leq a < b \leq 6$
  1. E is irreducible over $\mathbb{Q}$,
  2. E has a root modulo 8461

Binomial $aX^n - b$

For **any prime** p and $n \in \mathbb{N}_{\geq 2}$, there exists a binomial $E = aX^n - b$ with

1) E is irreducible over $\mathbb{Q}$,
2) E have at least one root, γ, in $\mathbb{F}_p$,
3) $1 \leq a < b \leq \frac{57}{34} n$

Example

- $P = 8461$, $n = 4$
- $\frac{57}{34} n \sim 6.71$
- There exist an irreducible polynomial $aX^4 - b$ with $1 \leq a < b \leq 6$
  1. E is irreducible over $\mathbb{Q}$,
  2. E has a root modulo 8461

Solution

$$3X^4 - 4, X^4 - 5$$

PMNS Properties

Advantage

- Polynomial avoid costly carry management
- Natural Side-Channel resistance due too small redundancy
- Polynomial multiplication acceleratation (Karatsuba, Toom-Cook,NTT)

PMNS Properties

Advantage

- Polynomial avoid costly carry management
- Natural Side-Channel resistance due too small redundancy
- Polynomial multiplication acceleratation (Karatsuba, Toom-Cook,NTT)

Drawback

- Costly Comparison $<$
- Costly Euclidean division

Outline

Residue Number System

Chinese Remainder Theorem

- A RNS basis is $(m_1, m_2, \ldots, m_n)$ with $M = \prod_{i=1}^{n} m_i$
- If we consider $(x_1, x_2, \ldots, x_n)$ with $0 \le x_i < m_i$
- Then $\exists ! X < M$ with $x_i = |X|_{m_i} = X \bmod m_i$

Residue Number System

Chinese Remainder Theorem

- A RNS basis is $(m_1, m_2, \ldots, m_n)$ with $M = \prod_{i=1}^{n} m_i$
- If we consider $(x_1, x_2, \ldots, x_n)$ with $0 \leq x_i < m_i$
- Then $\exists! X < M$ with $x_i = |X|_{m_i} = X \bmod m_i$

Arithmetic

- Addition: $A + B = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$
- Multiplication: $A \times B = (a_1 \times b_1, a_2 \times b_2, \ldots, a_n \times b_n)$

Residue Number System

Example

- A RNS basis: $(31, 32, 33)$ with $M = 32736$

Residue Number System

Example

- A RNS basis: $(31, 32, 33)$ with $M = 32736$
- $A = 6932 \rightarrow (19, 20, 2)$
- $B = 4121 \rightarrow (29, 25, 29)$

Residue Number System

Example

- A RNS basis: $(31, 32, 33)$ with $M = 32736$
- $A = 6932 \rightarrow (19, 20, 2)$
- $B = 4121 \rightarrow (29, 25, 29)$
- $A \times B = (24, 20, 25) = 20980 \bmod 32736$

Residue Number System

Example

- A RNS basis: $(31, 32, 33)$ with $M = 32736$
- $A = 6932 \rightarrow (19, 20, 2)$
- $B = 4121 \rightarrow (29, 25, 29)$
- $A \times B = (24, 20, 25) = 20980 \bmod 32736$
- $M = 32736 \neq P = 8461$

Residue Number System

Example

- A RNS basis: $(31, 32, 33)$ with $M = 32736$
- $A = 6932 \rightarrow (19, 20, 2)$
- $B = 4121 \rightarrow (29, 25, 29)$
- $A \times B = (24, 20, 25) = 20980 \bmod 32736$
- $M = 32736 \neq P = 8461$

Solution: RNS to RNS conversion

- Modular Multiplication can be done modulo P
- Using **two basis** larger than P
- A second RNS basis: $(29, 35, 37)$ with $M = 37555$

RNS Properties

Advantage

- Distribute operations from large number to small residues
- Addition and multiplication can be parallelized.
- Best case: $X_0 Y_0 + X_1 Y_1 + \cdots + X_k Y_k$ in vector-matrix multiplication.

RNS Properties

Advantage

- Distribute operations from large number to small residues
- Addition and multiplication can be parallelized.
- Best case: $X_0 Y_0 + X_1 Y_1 + \cdots + X_k Y_k$ in vector-matrix multiplication.

Drawback

- RNS to RNS Conversion is Quadratic
- Moduli Picking
- Costly Comparison $<$
- Costly Euclidean division

Outline

Mersenne Moduli

Mersenne Modular Reduction - Lehmer 51 (Thanks Markku)

Input : $P = 2^n - 1$
Output: $C = AB \bmod P$
begin
  $C \leftarrow AB$
  $C \leftarrow [C]_n + [C]^n$
  if $\underline{C \geq P}$ **then** $C \leftarrow C - P$;
end

Mersenne Moduli

Advantage

Cheap Reduction

Mersenne Moduli

Advantage

Cheap Reduction

Drawback

Low density for **Prime** Mersenne

$2^n - 1$ is prime for $n =$

$2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217 \ldots$

Pseudo Mersenne - Crandall 92

Idea

Extend Mersenne Number to grow density to ensure some prime options

Pseudo Mersenne - Crandall 92

Idea

Extend Mersenne Number to grow density to ensure some prime options

Pseudo Mersenne Modular Reduction

Input : $P = 2^n - c$ and $0 \leq c^2 < 2^n$

Output: $C = AB \bmod P$

begin

  $C \leftarrow AB$

  $C \leftarrow [C]_n + [C]^n c$

  $C \leftarrow [C]_n + [C]^n c$

  if $\underline{C \geq P}$ **then** $C \leftarrow C - P$;

end

Pseudo-Mersenne Moduli: $2^n - c$

Advantage

- Density allows to have some prime. Example: $2^{32} - 5$
- Great for Random Sampling

Pseudo-Mersenne Moduli: $2^n - c$

Advantage

- Density allows to have some prime. Example:$2^{32} - 5$
- Great for Random Sampling

Modular Inversion

- If p is large, **Euclid Algorithm** or its variant
- If p is small, **Modular exponentiation** using Fermat Theorem
- If p is small and Pseudo Mersenne, **Takagi Algorithm**

Pseudo-Mersenne Moduli: $2^n - c$

Advantage

- Density allows to have some prime. Example:$2^{32} - 5$
- Great for Random Sampling

Modular Inversion

- If p is large, **Euclid Algorithm** or its variant
- If p is small, **Modular exponentiation** using Fermat Theorem
- If p is small and Pseudo Mersenne, **Takagi Algorithm**

Drawback

Reduction not so cheap for word size moduli

Special Moduli

Mersenne like moduli - Barrett Friendly

- Mersenne Numbers $P = 2^n - 1$
- Pseudo Mersenne $P = 2^n - c$ (Crandall 92)
- Pseudo Mersenne with $c > 2^{n/2}$ (BIP'02)
- Generalized Mersenne $P = 2^n - 2^k \pm 1$ (Solinas 99)
- More Generalized Mersenne (Chung-Hassan'04)

Special Moduli

Mersenne like moduli - Barrett Friendly

- Mersenne Numbers $P = 2^n - 1$
- Pseudo Mersenne $P = 2^n - c$ (Crandall 92)
- Pseudo Mersenne with $c > 2^{n/2}$ (BIP'02)
- Generalized Mersenne $P = 2^n - 2^k \pm 1$ (Solinas 99)
- More Generalized Mersenne (Chung-Hassan'04)

Some other Special Moduli

- Montgomery-Friendly: $P = c2^k - 1$ (Hamburg'12)
- Moduli adapted to PMNS (BIP'04)
- For RNS, $P = M^2 - 2$ (Bigou-Tisserand'15)
- NFLlib: $2^{n-k} - 2^{n-2k} + 2^{n-3k} < P < 2^{n-k}$

Outline

Conclusion

	Size $\sim$	Operation	Solutions
RSA *	$2000 - 4000$	$\times$	Montgomery, RNS
ECC	$200 - 500$	$+, \times, x^{-1}$	Gen/Pseudo Mersenne RNS, PMNS
LWE	$10 - 30$	$+, \times$	Montgomery Friendly, RNS
Lattice	1000	$+, \times$	Pseudo-Mersenne, RNS
Isogeny *	$300 - 1000$	$+, \times, x^{-1}$	Specific, Montgomery, PMNS
Multivariate	10	$+, \times$	Mersenne, Pseudo Mersenne
Tensor Alternating	$10 - 30$	$+, \times, \$$	Pseudo Mersenne Word Size Modular Arithmetic

Conclusion

Modular Arithmetic Tools

1. There is a wild range of options
2. Adaptable for different applications/constraints

Conclusion

Modular Arithmetic Tools

1. There is a wild range of options
2. Adaptable for different applications/constraints

Structure in Cryptography

1. Structure have an unknown cost
2. Gain need to be at least significant
3. With no other solutions