Fermat-FHE Cryptosystem

MACAO, Wollongong workshop
November 27th, 2019

Antoine Joux

Mersenne/Fermat systems

- Inside Ring and Noise family with
  - NTRU
  - Codes
  - Ideal Lattices, RLWE

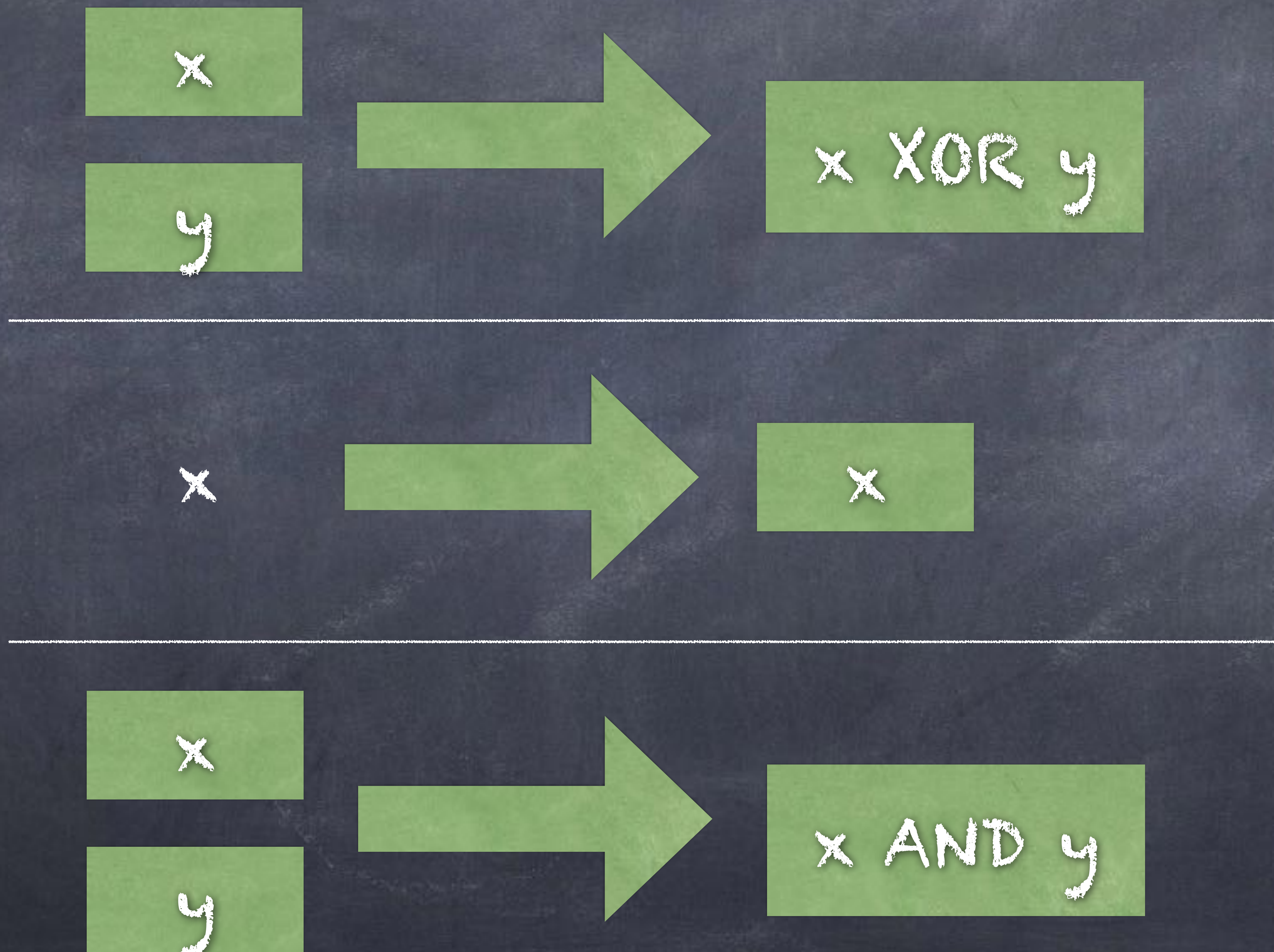
- With a different Ring
  - Z/pZ (p Mersenne prime)
  - Z/NZ (N Fermat number)

Fermat FHE

Fully Homomorphic Encryption (FHE)

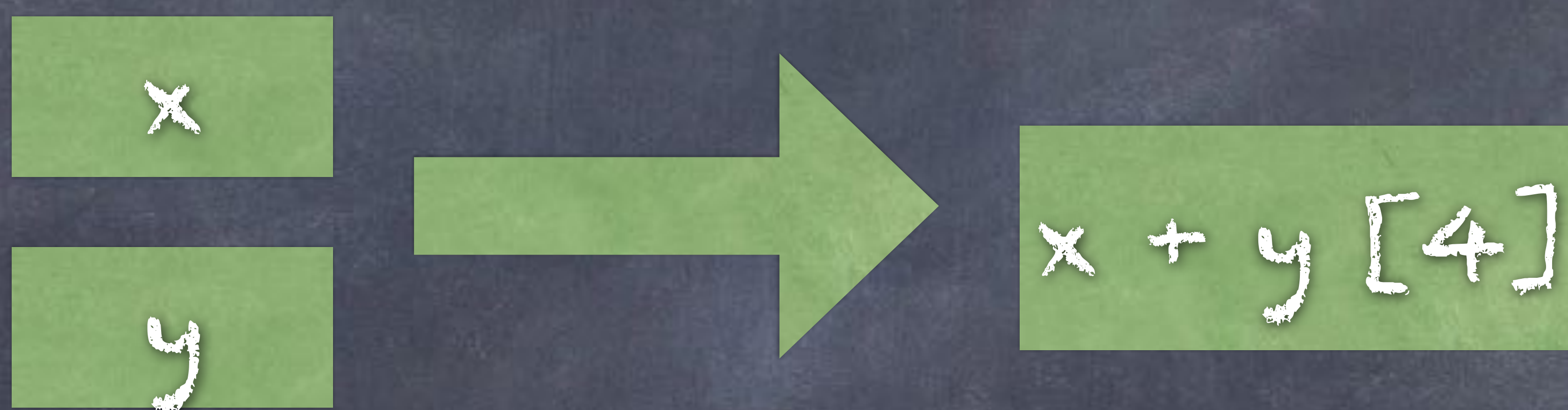
- Encryption Scheme allowing arbitrary computations on encrypted data
- Usually, from universal set of gates
- Principle Rivest, Adleman, Dertouzos (78)
- First system Gentry (2009)

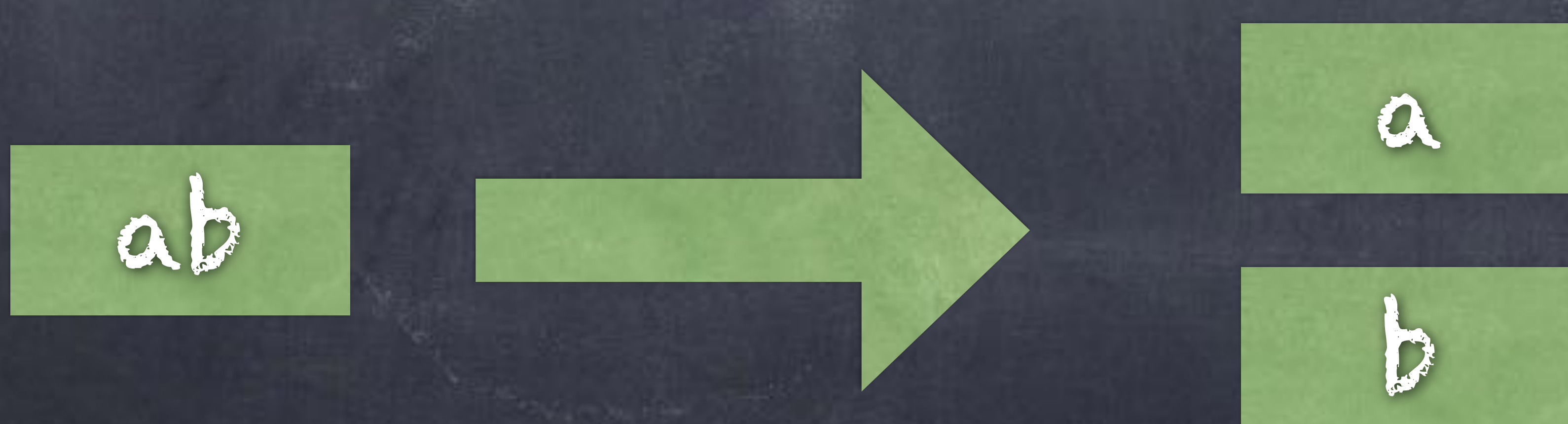
FHE basic need



Addition of bits (mod 4)

$x + y = 2\,(x \text{ AND } y) + (x \text{ XOR } y)$





Modulo Fermat Numbers

$$F = 2^{2^f} + 1$$

Write $f = l + h$; define $L = 2^l$ and $H = 2^h$

$X \bmod F$ (except -1) is an LH bit number

$X =$

X_{L-1}	...	X_2	X_1	X_0

L blocks of H bits

Approximate encryption

Params: $F = 2^{2^f} + 1$, L, H

Priv Key: $S \pmod F$

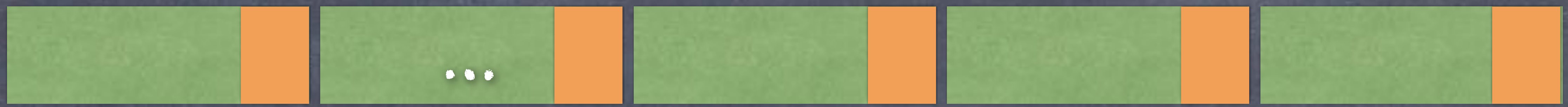
Approx encryption of X:
pair (A, B) with

$$B = AS + X + E \pmod F$$

$E =$ 

Noise E

E =



$$E = e_0 + 2^H e_1 + \ldots + 2^{(L-1)H} e_{L-1}$$

With each e_i small (from some err. dist)

Key owner operations

- Approx Encryption/Decryption are easy
  - From (A, B) compute B−AS
- Encrypt/decrypt bits (exactly)
  - Encrypt $X = b\ 2^M$
  - Decryption $\lfloor (X \bmod 2^H)/2^M \rfloor$

FHE operations

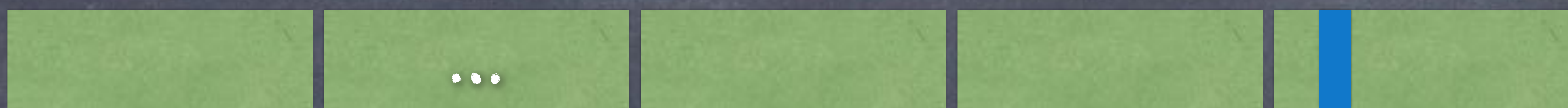
- (A_0+A_1, B_0+B_1) encrypts X_0+X_1
  (with slightly larger noise)

XOR and AND : Need an extraction technique

Bit Extraction (a.k.a. gate bootstrapping)

Encrypted Mux

- (A_0, B_0) encryption of X_0



- (A_1, B_1) encryption of X_1

- Produce Re-encryption of (A_c, B_c)

Using a special encryption of c

Using Multiplication by c

- From (A, B) Approx Encrypt of X
- Compute (A', B') App Enc of cX

- Then Mux(c, (A_0, B_0), (A_1, B_1))
  $= (A_0, B_0) + c\ (A_1\text{-}A_0, B_1\text{-}B_0)$

How to multiply by (specially) encrypted c ?

Special Encryption of c

- (K_i, L_i) Approx Encrypt of $2^i c$
- (M_i, N_i) Approx Encrypt of $-2^i cS$
- Used in conjunction with the binary decomposition of (A,B)

  => Multiplication

Block decomposition

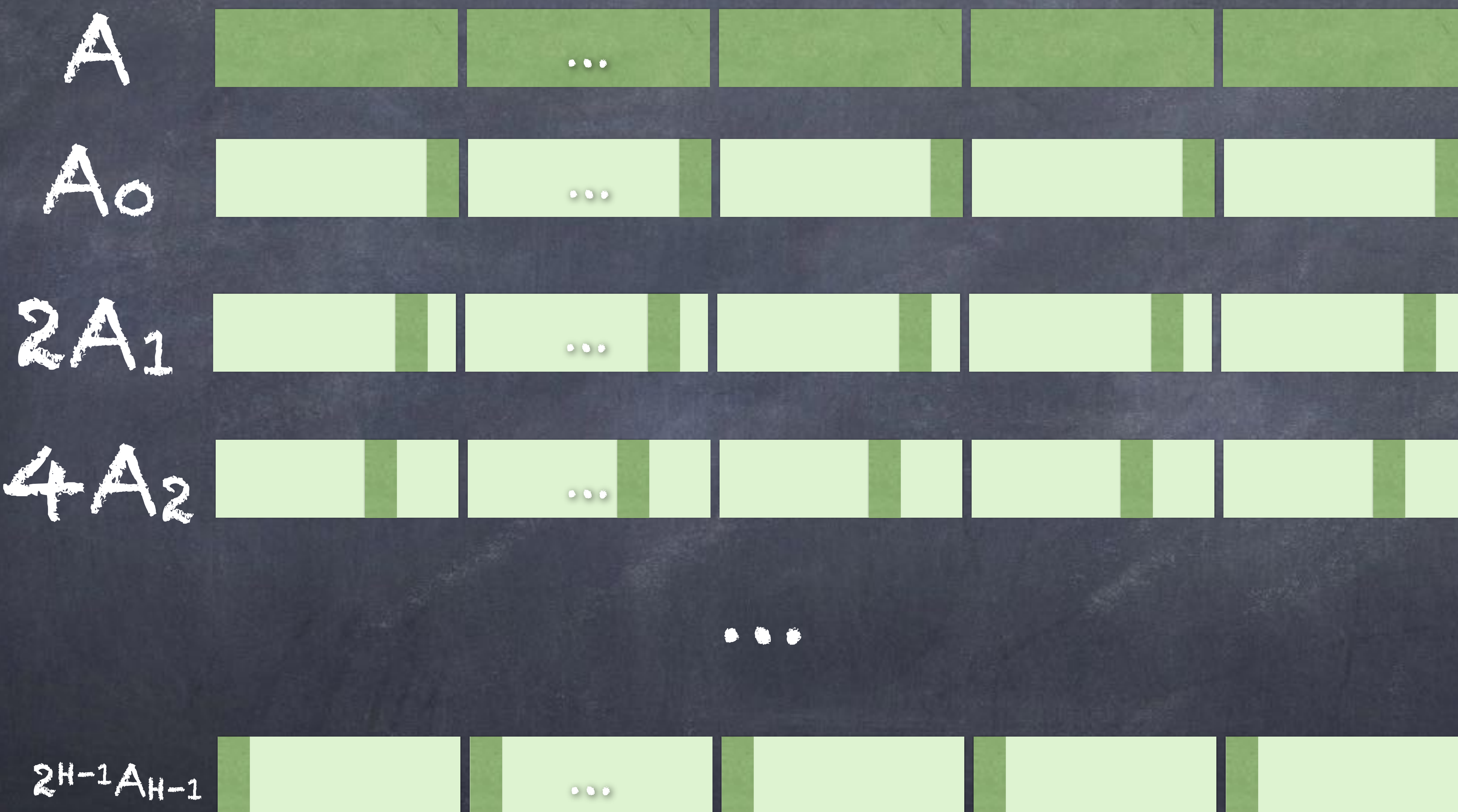
From Binary decomp $$A = \sum_{i=0}^{HL-1} a_i 2^i$$

Make blocks $$A_i = \sum_{j=0}^{L-1} a_{jH+i} 2^{jH}$$

We have $$A = \sum_{i=0}^{H-1} A_i 2^i$$

Idem for B

Block decomposition



Encrypted Multiplication

Let $$A' = \sum_{i=0}^{H-1} \left(B_i K_i + A_i M_i\right)$$

$$B' = \sum_{i=0}^{H-1} \left(B_i L_i + A_i N_i\right)$$

By linearity, it is an encryption of

$$X' = \sum_{i=0}^{H-1} \left(B_i 2^i c - A_i 2^i c S\right)$$

Encrypted Multiplication

We have
$$X' = c\left(\sum_{i=0}^{H-1} 2^i B_i - S \sum_{i=0}^{H-1} 2^i A_i\right)$$

Thus
$$X' = c\,(B - AS)$$

I.e. desired encryption c(A,B)

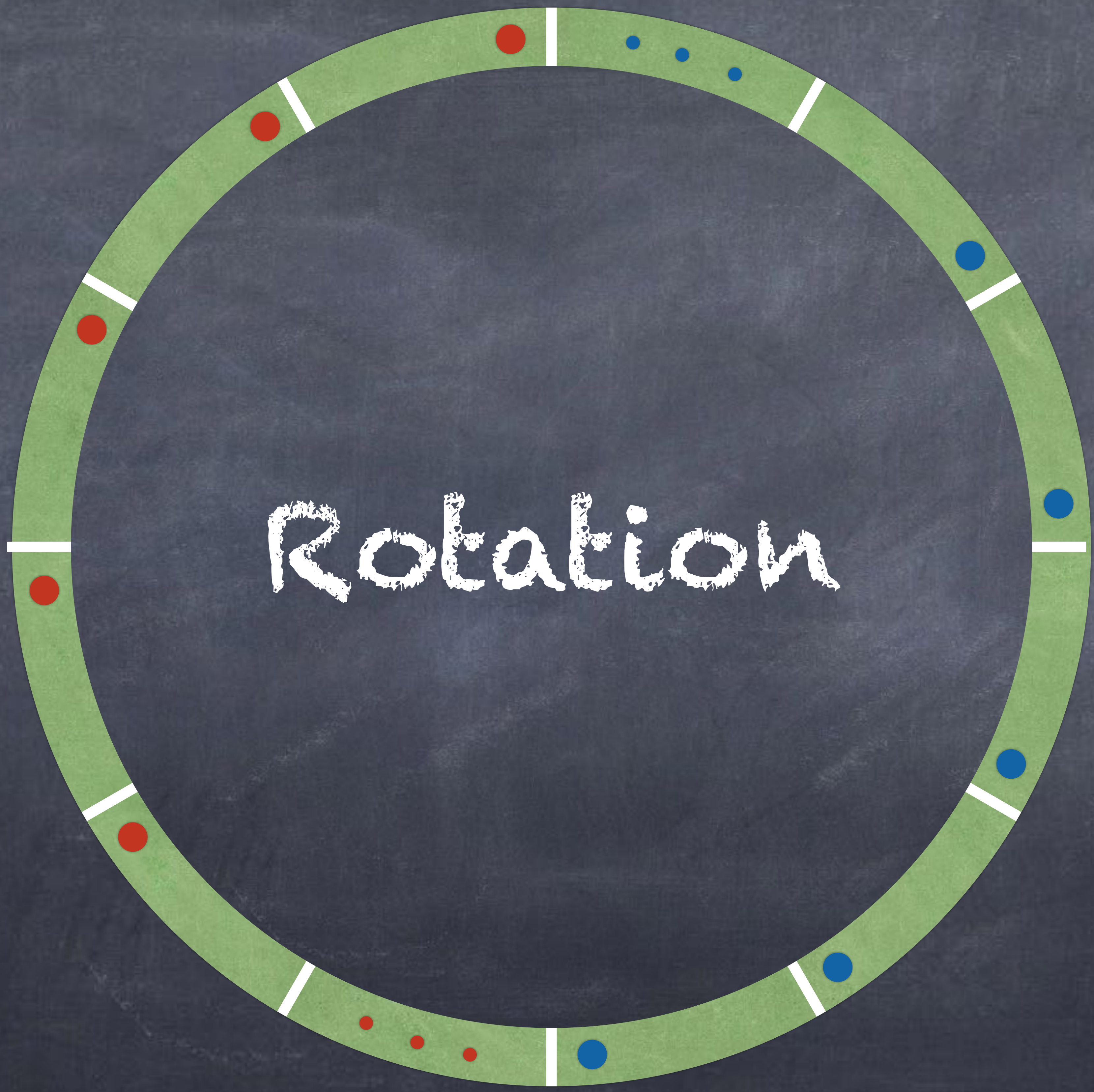
A remark: rotation

Rotate to the left

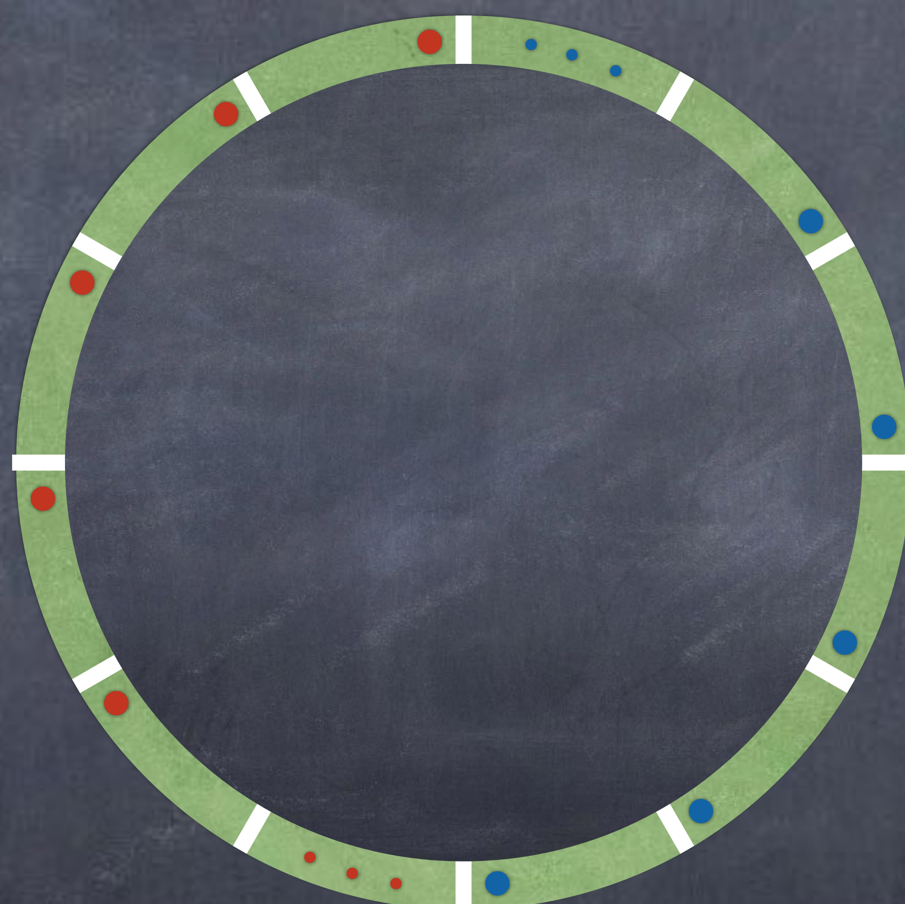
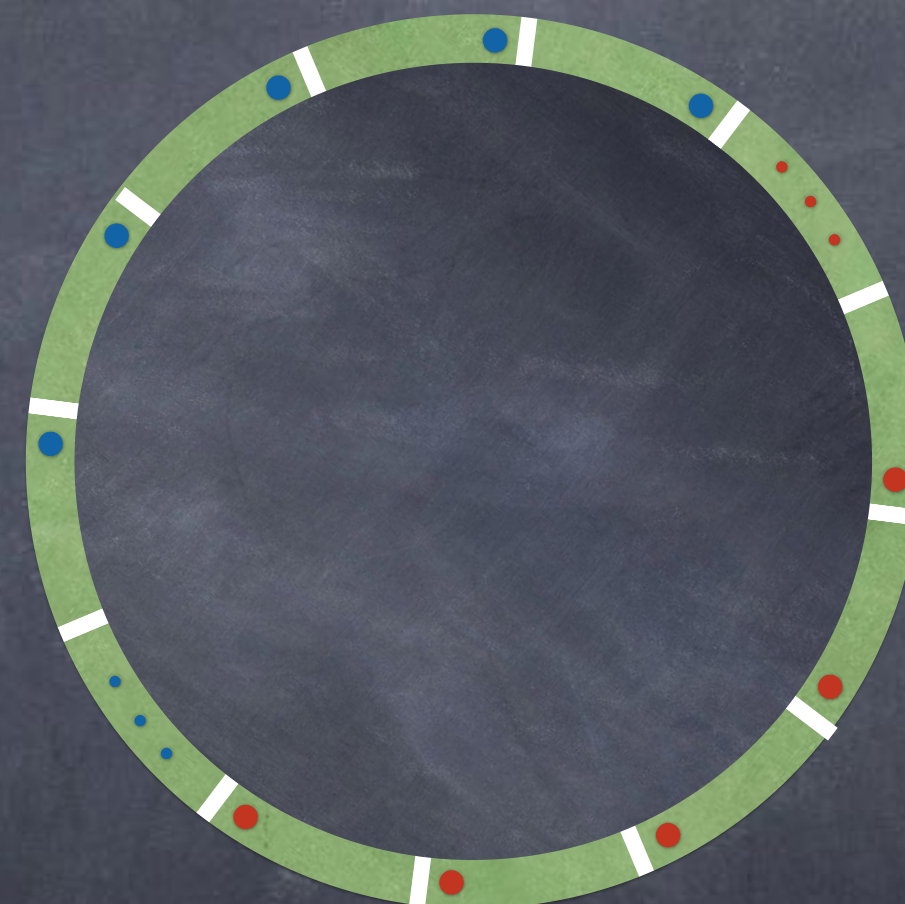
Rotate to the left

Rotate to the left
until L rotations

Virtual 2L blocks buffer



Conditional Rotation

Mux(c,  , )

Back to Bit Extraction

Perform rotation by an approximation of the decryption mod 2L

Exact decryption formula

$$V = \left\lfloor \frac{\left(B - \mathcal{S}A \bmod F\right) \bmod 2^H}{2^{b_P}} \right\rceil \pmod 2$$

Approximate it by :

$$\left\lfloor \frac{B_0 - s_0 A_0}{2^{b_P - \ell}} \right\rceil + \sum_{i=1}^{L-1} \left\lfloor \frac{A_i}{2^{b_P - \ell}} \right\rceil s_i \pmod{2L}$$

Execute as a sequence of cond. rotation

How to approximate

- Forget the carry of reduction mod F
- Decompose multiplication by blocks
- Lower power of 2 in second modulus
- Closest integer : free from rotating buffer

Low-block of output

or

Final step

Low-block of output

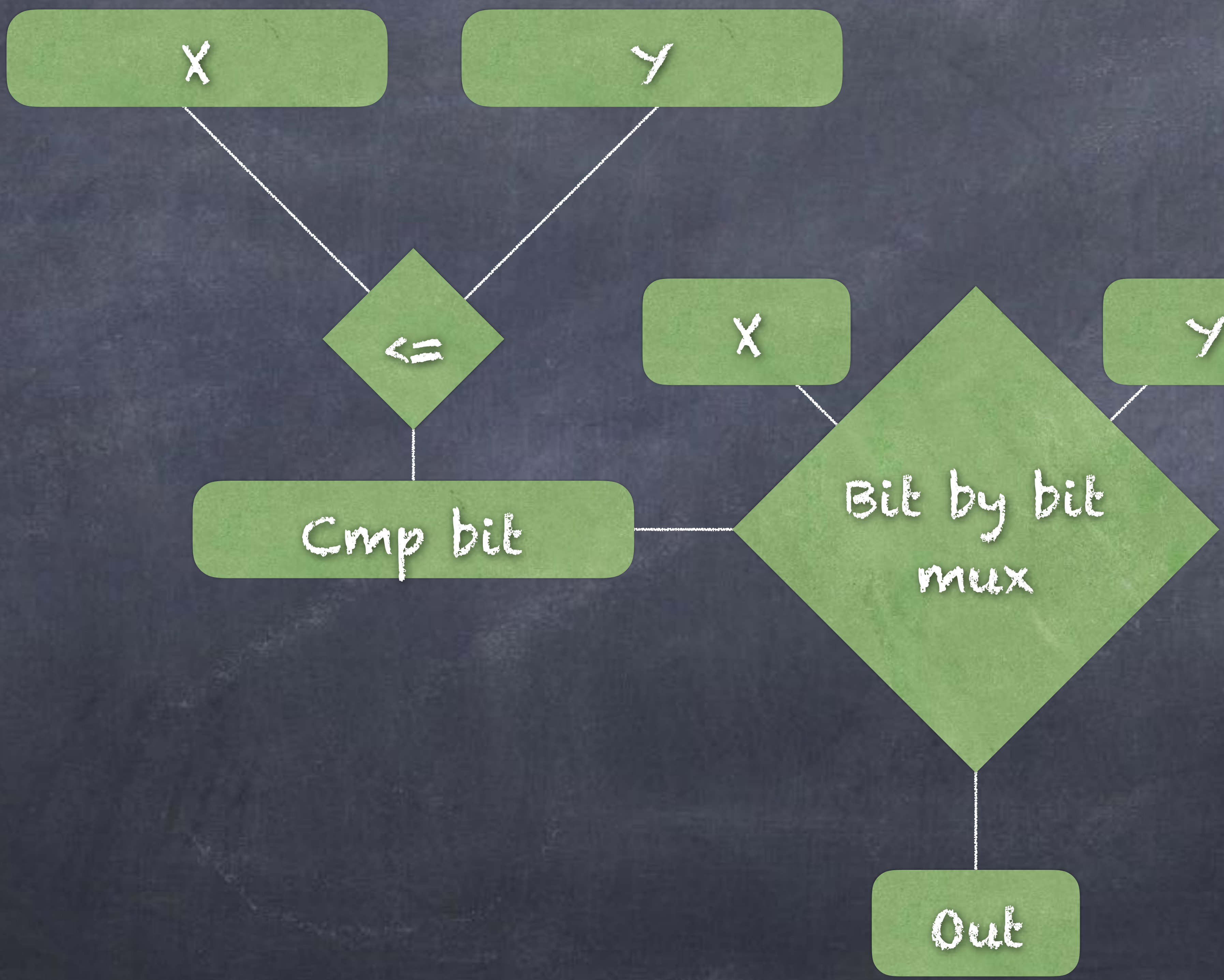
-2^{b_M-1} $\qquad$ 2^{b_M-1}

Add 2^{b_M-1}

A test application

- User encrypts X and Y (two 16-bit integers)
- Cloud computes min(X,Y)
- Proposed as test by TFHE

Circuit for min



Logical Mux

If (b, Y, X)

=

(b AND Y) XOR (NOT(b) AND X)

Bit-by-Bit Comparator

- Input: b [comparison mod 2^i], x_i, y_i
- Output: b' [comparison mod 2^{i+1}]

$$b' = \text{If}\,(x_i = y_i,\ b,\ x_i) = \text{If}\,(x_i \text{ XOR } y_i,\ x_i,\ b)$$

High bits equal => dont change comparison

High bits diff => x_i (0 if x is min)

Repeat 16 times !

Test implementation parameters

$L=1024$ $H=32$

Direct GMP implem : 16.2 s

Portable NTT-based approx mult : 9.4 s

TFHE 500-bit key, 32-bit words

Portable FFT implem : 7.6 s

Highly optimised float implem : 0.8 s

Conclusion