Masking Lattice-based Post-quantum Cryptography

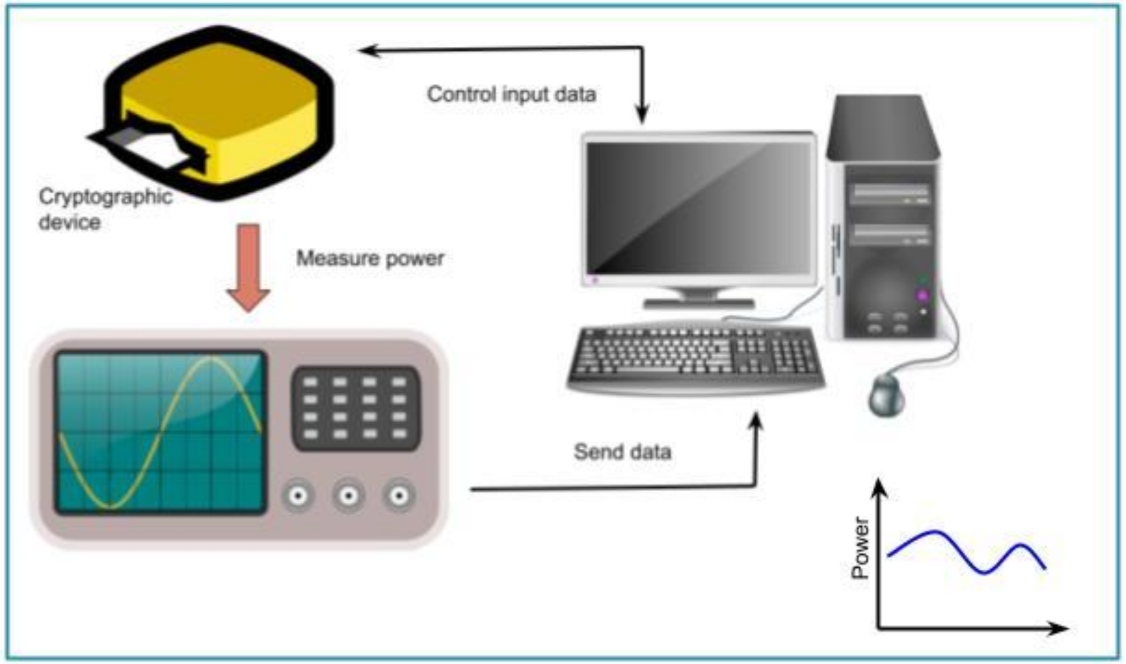
Suparna Kundu

imec-COSIC, KU Leuven

SAFEST 2023

Side-channel attacks (SCA) exploits implementation flaws

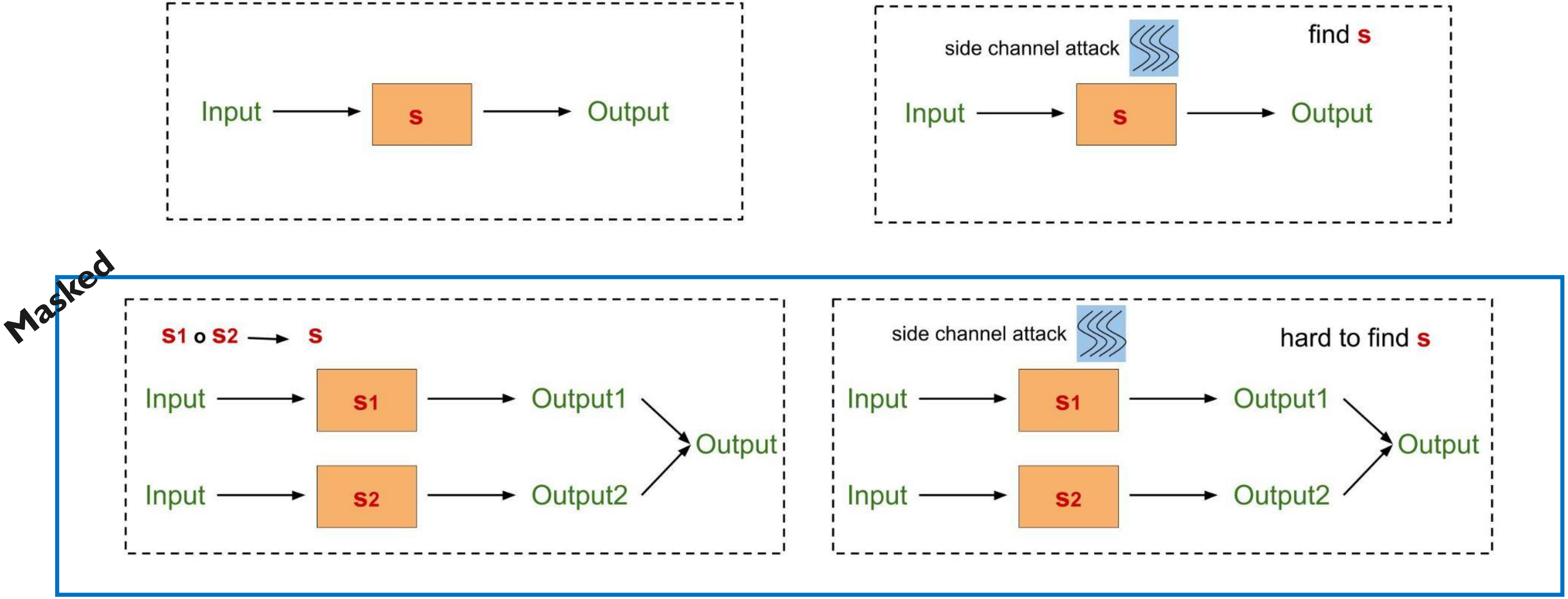
- Power consumption
- Electromagnetic radiation
- Timing information



Leak sensitive information

KU LEUVEN

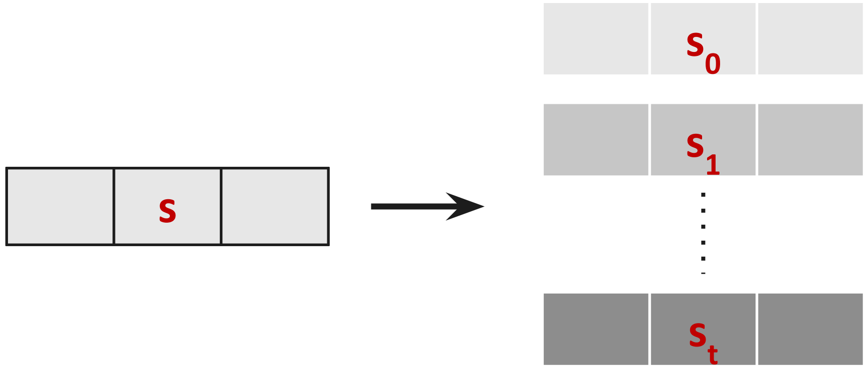
Masking is a countermeasure of SCA



KU LEUVEN

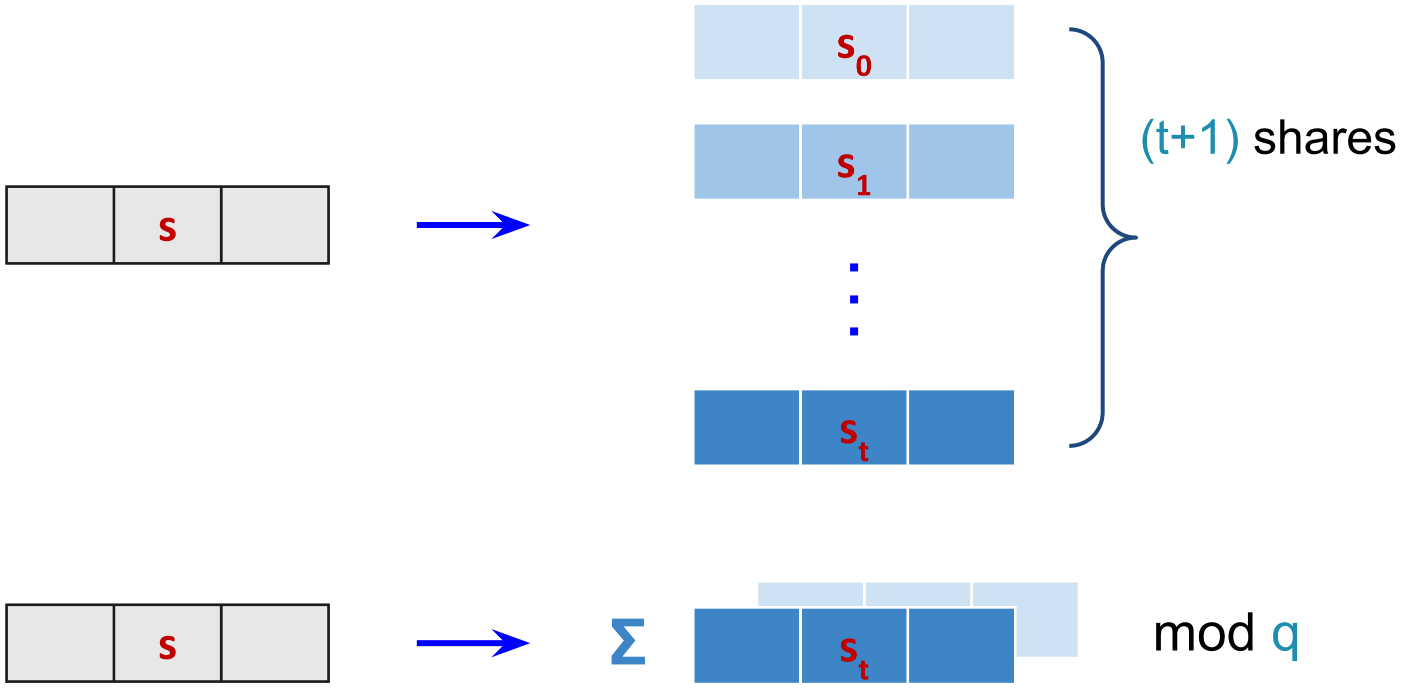
Higher-order (t-order) masking can prevent higher-order (t-order) SCA

- t-order masking:
  - Sensitive value splits into (t+1) shares
  - Perform all operations on each share separately



- Attack model:
  - Adversary can see up to t intermediate sensitive values

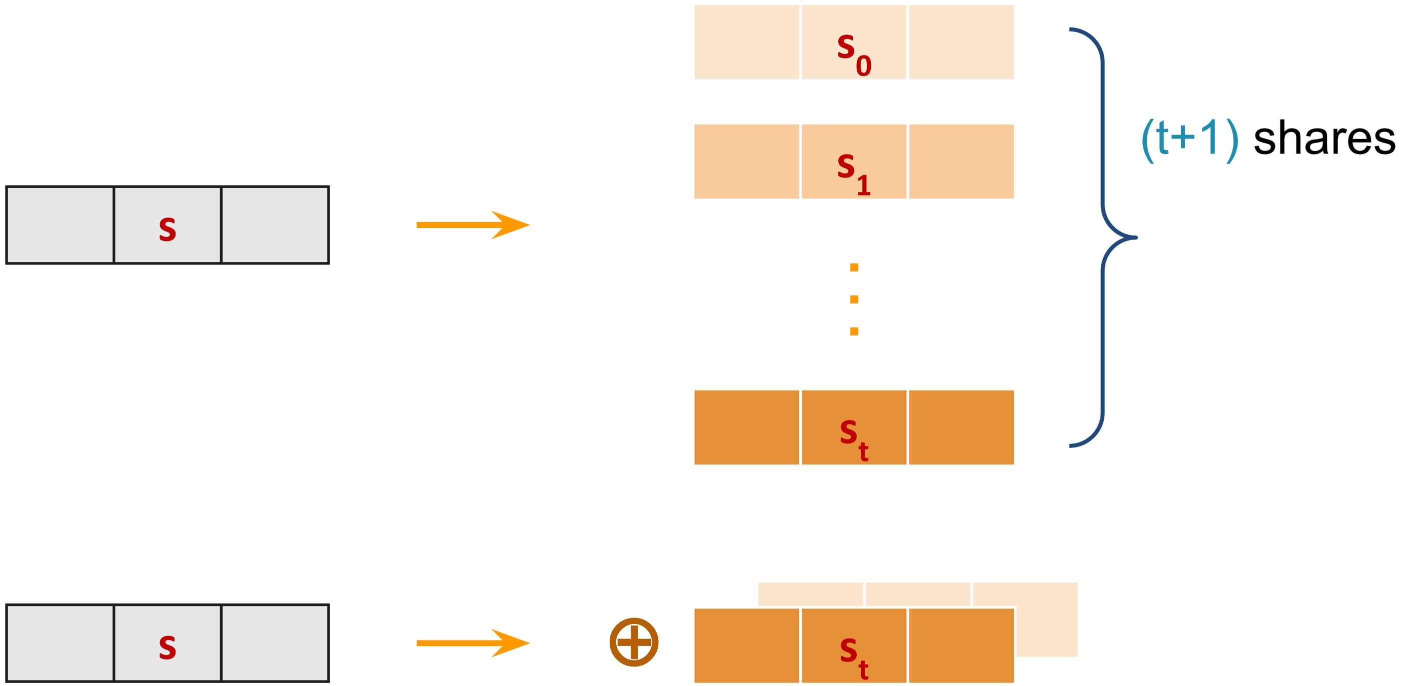
Arithmetic masking used for arithmetic operations



Useful for arithmetic operations:

- modular addition,
- modular subtraction,
- modular multiplication

Boolean masking used for bitwise operations



Useful for Boolean operations:

- xor
- and
- or
- shift

Goals

- Side-channel secure post-quantum MLWE/MLWR based KEM
    - Saber: MLWR based KEM (3rd round finalist)

      $\lfloor A \cdot \mathbf{s} \rceil = \mathbf{b}$ and $\mathbf{b}'$ random

      $\mathbf{b}'$ and $\mathbf{b}$ indistinguishable

- Generalized masked implementation
- Parameterized security order

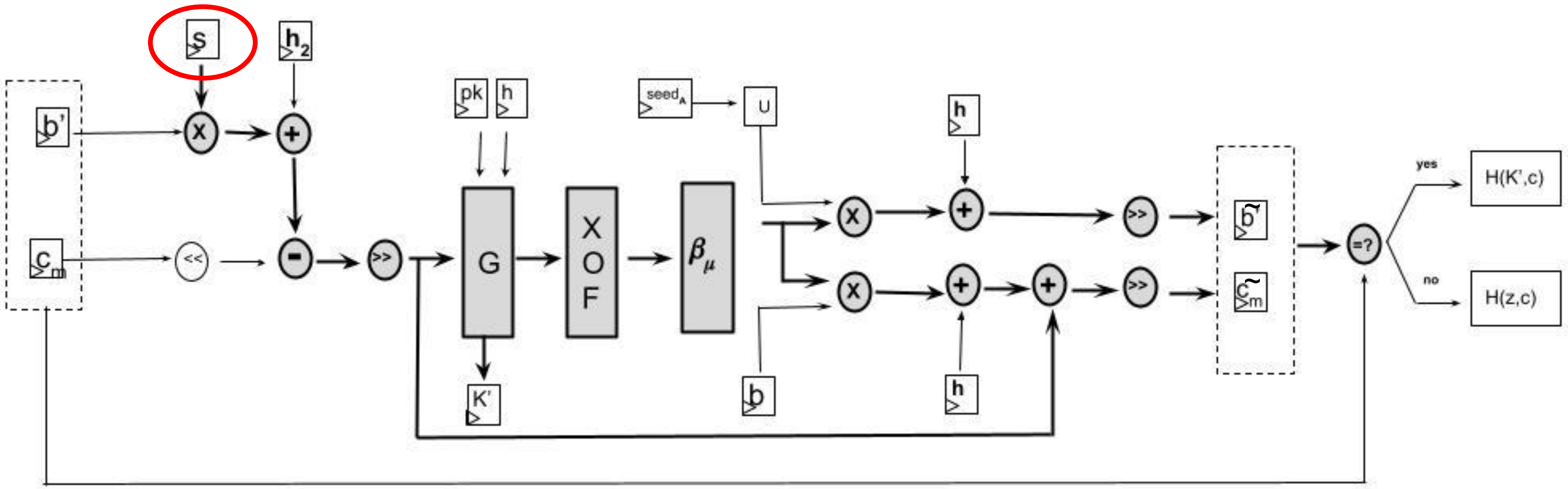
Challenges

- MLWE/MLWR based constructions uses arithmetic and Boolean operations
- Conversion algorithms are expensive
- Masked polynomial comparison
    - Easy for first-order
    - Costly for higher-order

Saber is a Key-encapsulation mechanism

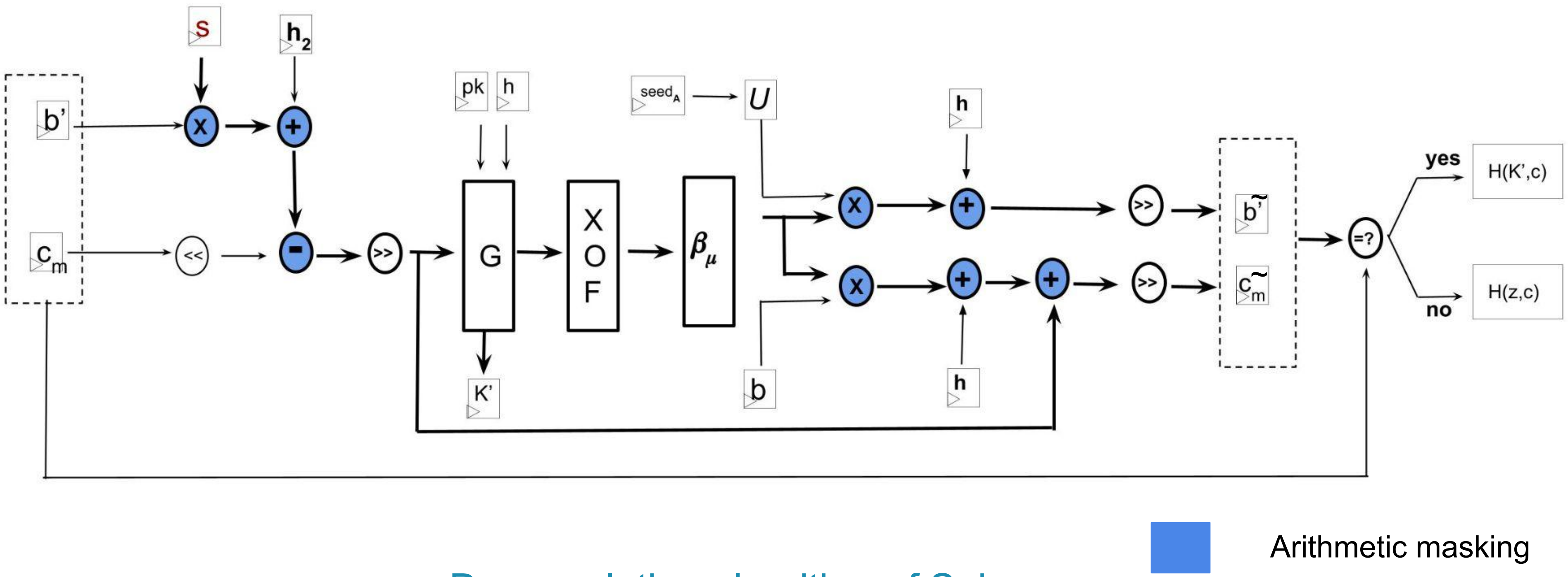
- Based on module learning with rounding (MLWR) problem $\lfloor A \cdot \mathbf{s} \rceil = \mathbf{b}$
  - Variant of learning with errors (LWE) problem $A \cdot \mathbf{s} + \mathbf{e} = \mathbf{b}$
- NIST 3rd-round finalist
- Algorithms: Key-Generation, Encapsulation, and Decapsulation
- CCA secure KEM
  - Same secret-key used for multiple Decapsulation

Decapsulation is SCA sensitive operation of Saber



Decapsulation algorithm of Saber

Arithmetic masked operations of Saber is represented with color blue



Decapsulation algorithm of Saber

Addition and multiplication operations are duplicates for each shares

Addition

- $z = x + y$
- $x = x_0 + x_1 + \ldots + x_t$
- When y is not masked
  - $z_0 = x_0 + y$
  - $z_i = x_i \quad \forall\ i = 1, \ldots, t$
  - $z = z_0 + z_1 + \ldots + z_t$

KU LEUVEN

Addition and multiplication operations are duplicates for each shares

Addition

- $z = x + y$
- $x = x_0 + x_1 + \ldots + x_t$
- When y is not masked
  - $z_0 = x_0 + y$
  - $z_i = x_i \quad \forall\ i = 1, \ldots, t$
  - $z = z_0 + z_1 + \ldots + z_t$
- When y is masked
  - $y = y_0 + y_1 + \ldots + y_t$
  - $z_i = x_i + y_i \quad \forall\ i = 0, \ldots, t$
  - $z = z_0 + z_1 + \ldots + z_t$

Addition and multiplication operations are duplicates for each shares

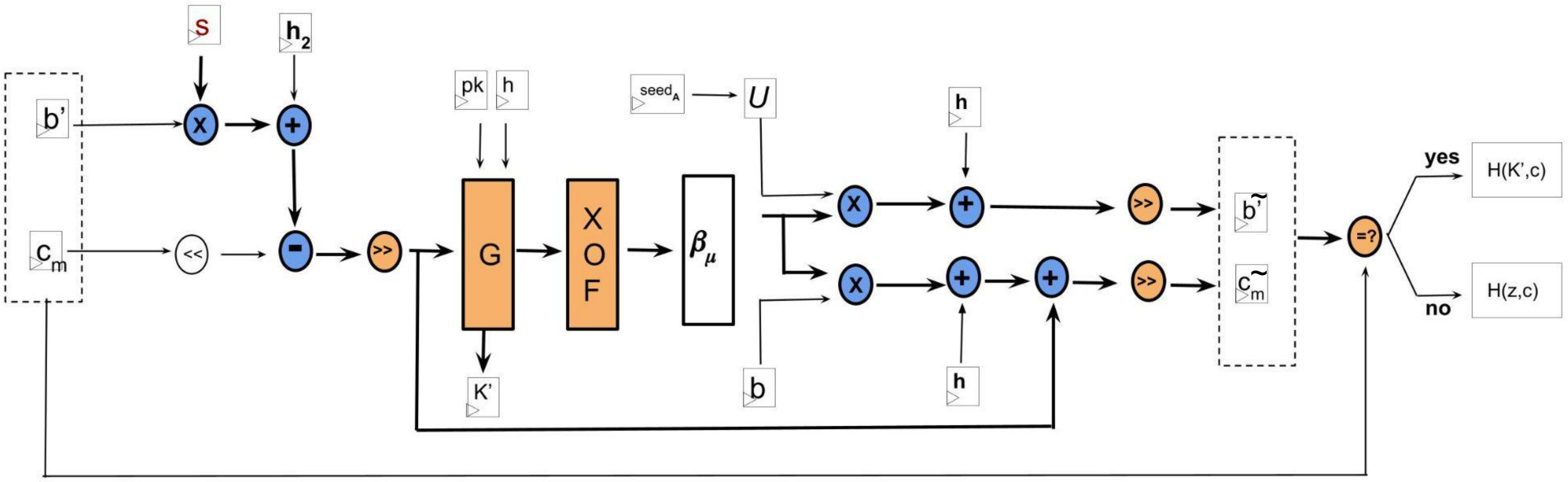
Addition

- $z = x + y$
- $x = x_0 + x_1 + \dots + x_t$
- When y is not masked
  - $z_0 = x_0 + y$
  - $z_i = x_i \quad \forall\ i = 1, \dots, t$
  - $z = z_0 + z_1 + \dots + z_t$
- When y is masked
  - $y = y_0 + y_1 + \dots + y_t$
  - $z_i = x_i + y_i \quad \forall\ i = 0, \dots, t$
  - $z = z_0 + z_1 + \dots + z_t$

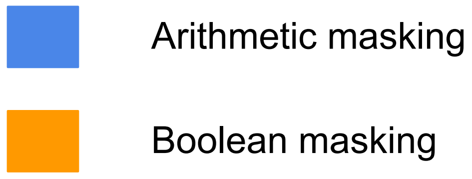
Multiplication

- $z = x \cdot y$
- Only one polynomial masked (x)
- $x = x_0 + x_1 + \dots + x_t$
  - $z_i = x_i \cdot y \quad \forall\ i = 0, \dots, t$
  - $z = z_0 + z_1 + \dots + z_t$

Boolean masked operations of Saber is represented with color Orange



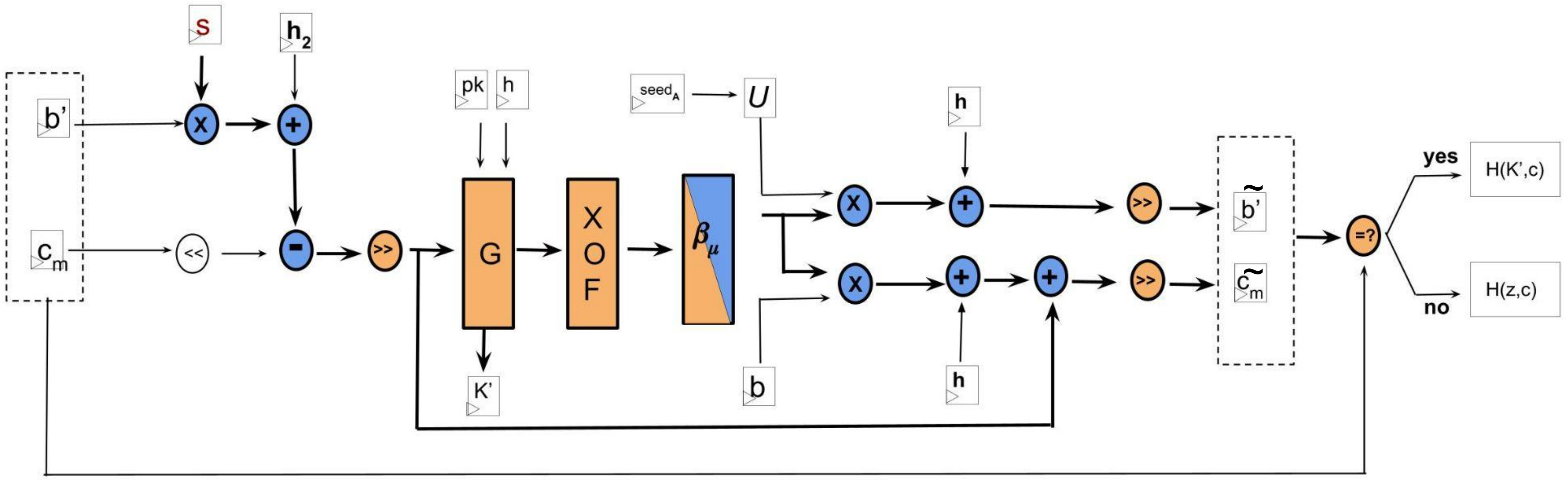
Decapsulation algorithm of Saber



KU LEUVEN

Masked Centered Binomial Distribution ($\beta\mu$) use both making techniques

- Uses Boolean to Arithmetic conversion (B2A)
- B2A is one of the costly operations



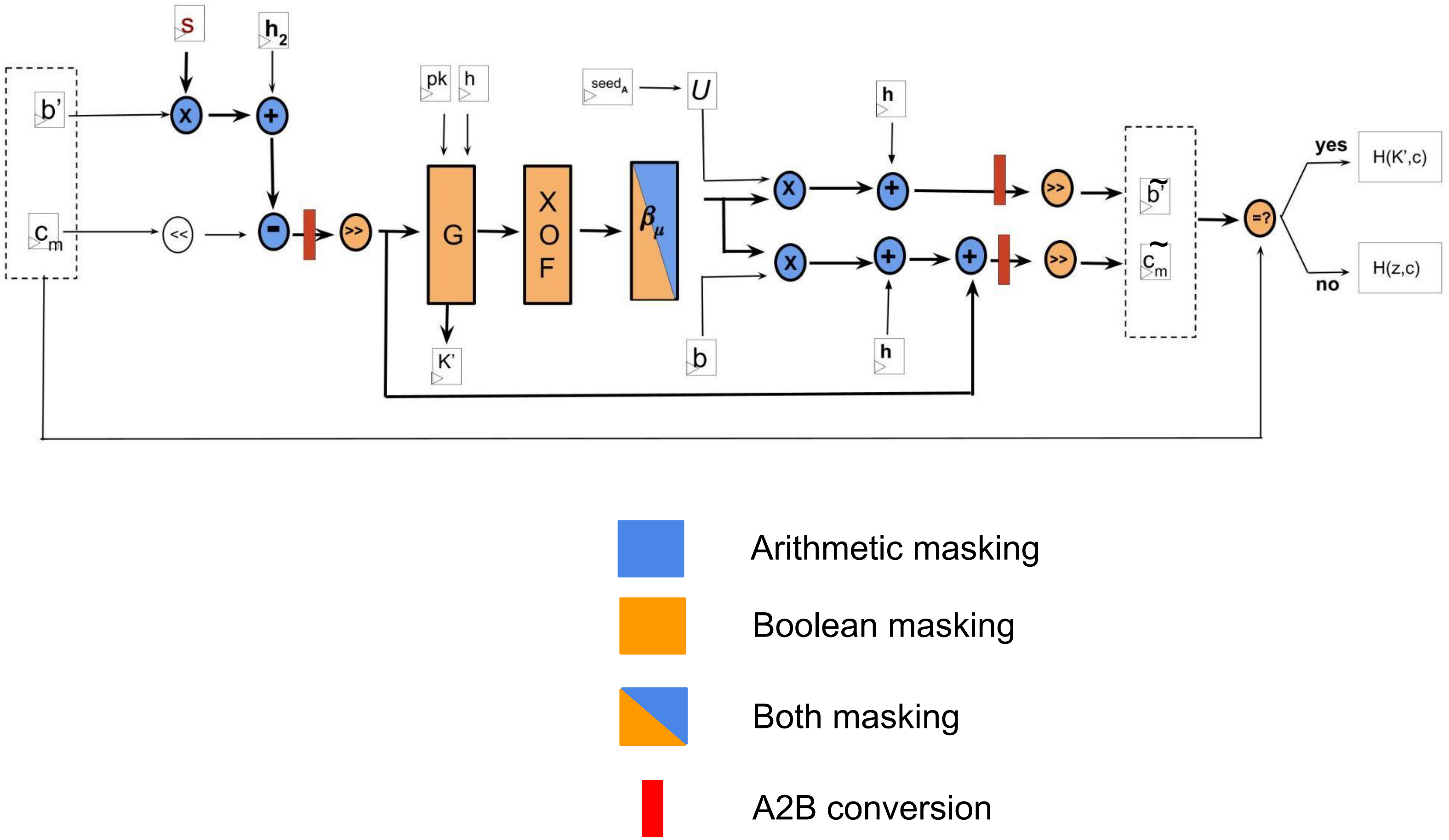
Arithmetic masking

Boolean masking

Both masking

A2B is needed to convert arithmetic shares to Boolean shares

- Before shift we use arithmetic to Boolean conversion (A2B)
- A2B is another costly operation



KU LEUVEN

Implementation results of masked Saber are in the table

Saber Decapsulation	Unmasked	1st-order	2nd-order	3rd-order
Performance [k]CPU cycles	1,121	3,022 (2.69x)	5,567 (4.96x)	8,649 (7.71x)
Random bytes	0	12k	42k	91k

Platform: ARM Cortex-M4

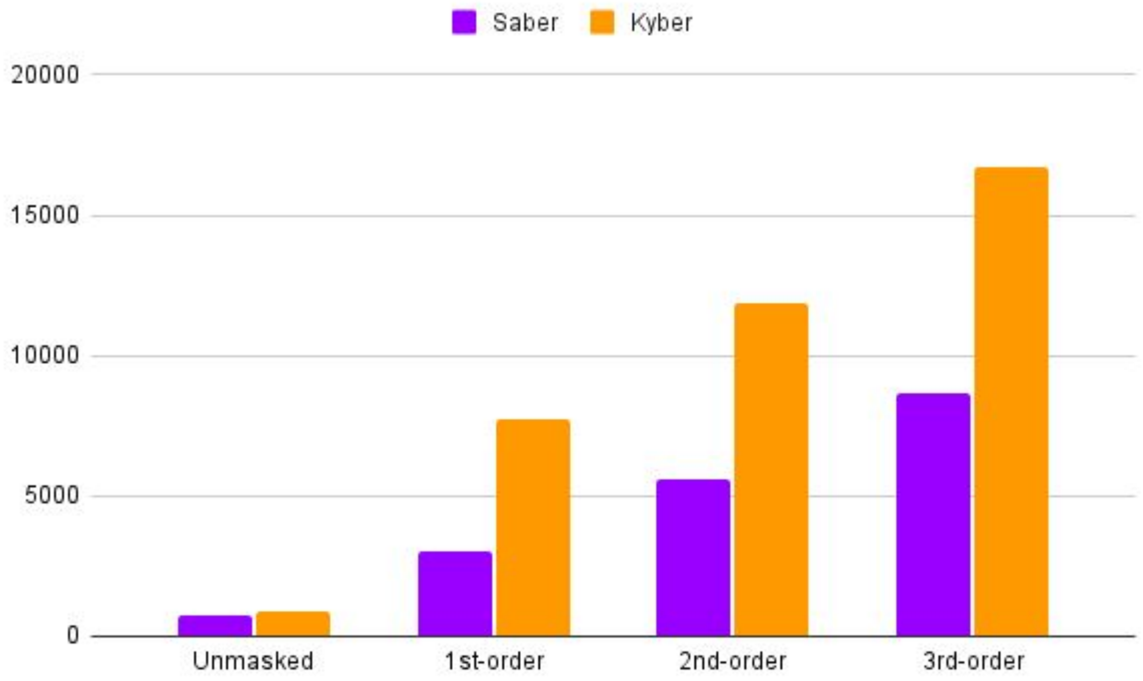
Framework: PQM4 [1]

Compiled with: arm-none-eabi-gcc

Version: 9.2.1

[1] PQM4: Post-quantum crypto library for the ARM Cortex-M4, Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K., https://github.com/mupq/pqm4.

Masked Saber perform better than masked Kyber

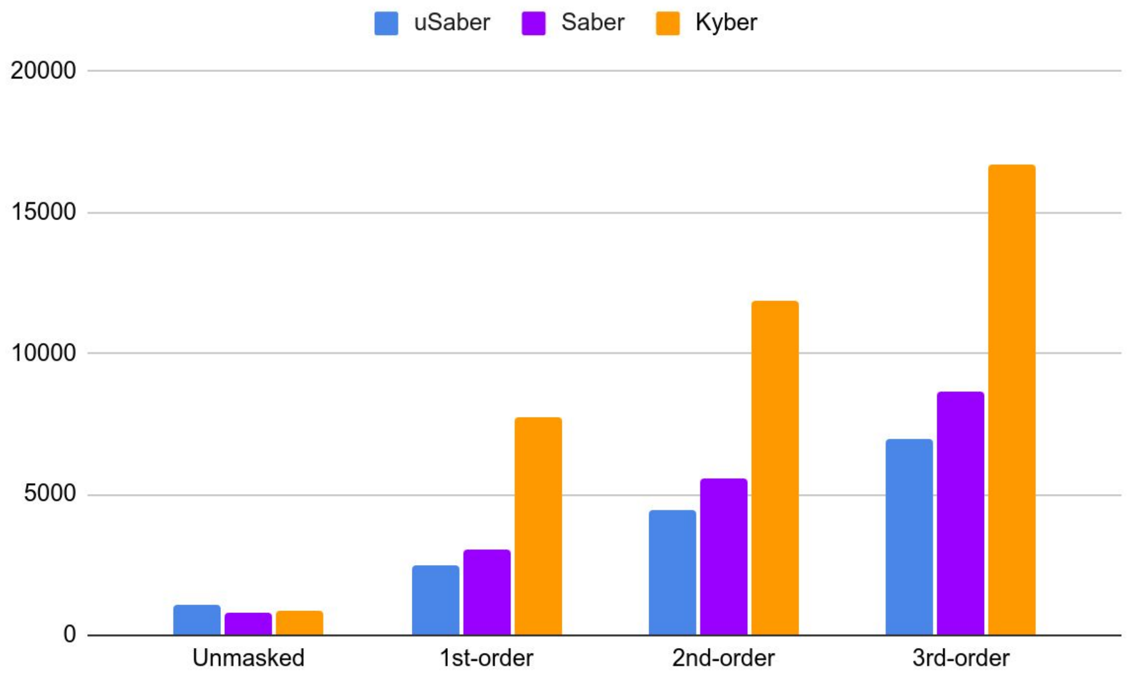


Performance (x1000 cpucycle)

Saber	Ours
Kyber	[1]

[1] Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based kems, Bronchain, O., Cassiers, G., https://eprint.iacr.org/2022/158.

Masked uSaber perform even better than masked Saber



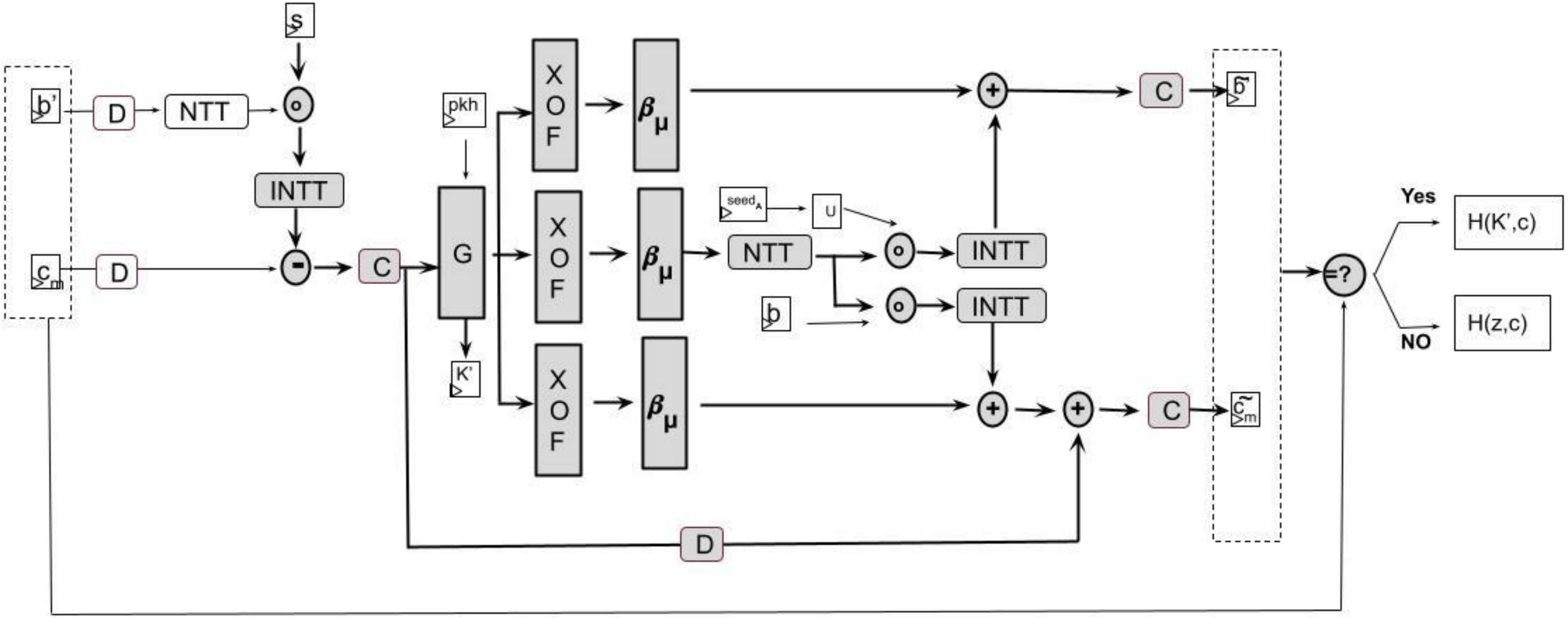
Performance (x1000 cpucycle)

uSaber	Ours
Saber	Ours
Kyber	[1]

[1] Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based kems, Bronchain, O., Cassiers, G., https://eprint.iacr.org/2022/158.

KU LEUVEN

Decapsulation operation of Kyber is similar to Saber



Decapsulation algorithm of Kyber

KU LEUVEN

A2B conversion algorithm [1]

Input : $\{x_i\}_{0\leq i\leq t}$ such that $x = \sum x_i \bmod 2^k$

Output : $\{c_i\}_{0\leq i\leq t}$ such that $\oplus c_i = \sum x_i \bmod 2^k$

1. $\{y_i\}_{0\leq i\leq t/2} \leftarrow$ A2B ($\{x_i\}_{0\leq i\leq t/2}$)
2. $\{y_i\}_{0\leq i\leq t} \leftarrow$ Expand ($\{y_i\}_{0\leq i\leq t/2}$)
3. $\{z_i\}_{0\leq i\leq t/2} \leftarrow$ A2B ($\{x_i\}_{(t/2+1)\leq i\leq t/2}$)
4. $\{z_i\}_{0\leq i\leq t} \leftarrow$ Expand ($\{z_i\}_{0\leq i\leq t/2}$)
5. $\{c_i\}_{0\leq i\leq t} \leftarrow$ SecAdd ($\{y_i\}_{0\leq i\leq t}$, $\{z_i\}_{0\leq i\leq t}$)
6. return $\{c_i\}_{0\leq i\leq t}$

[1] Tobias Schneider, Clara Paglialonga, Tobias Oder, and Tim Güneysu. Efficiently masking binomial sampling at arbitrary orders for lattice based crypto. In Dongdai Lin and Kazue Sako, editors, PKC 2019, Part II, volume 11443 of LNCS, pages 534–564. Springer, Heidelberg, April 2019.

Steps of A2B conversion algorithm

Input : $\{x_i\}_{0\leq i\leq 3}$ such that $x = \sum x_i \bmod 2^k$

Output : $\{c_i\}_{0\leq i\leq 3}$ such that $\oplus c_i = \sum x_i$

Input

x_0	x_1	x_2	x_3

Output

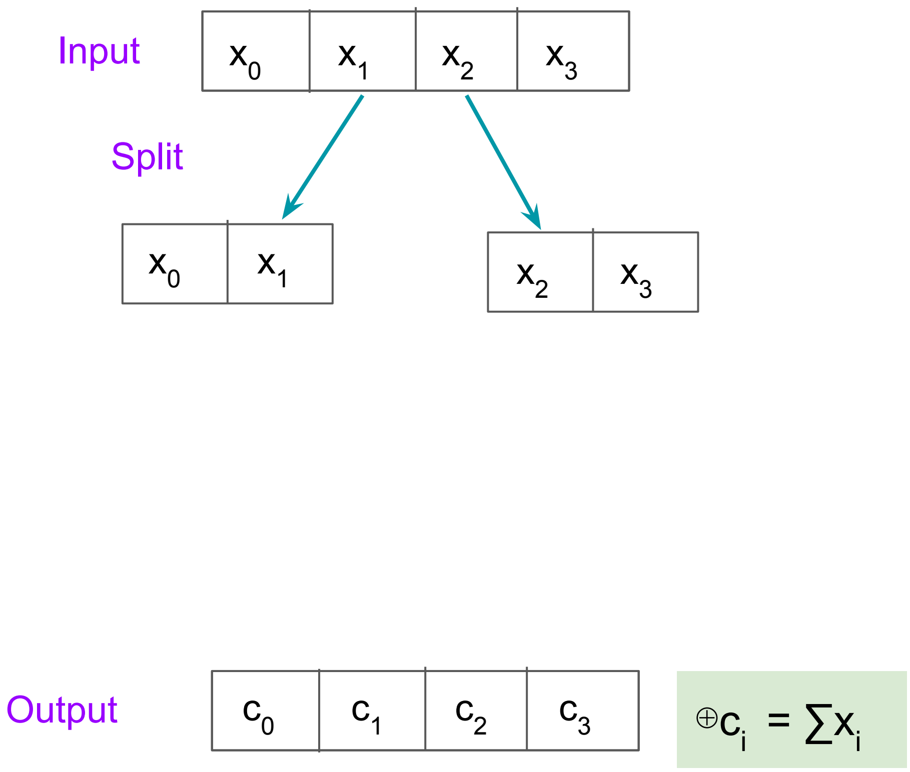
c_0	c_1	c_2	c_3

$\oplus c_i = \sum x_i$

Steps of A2B conversion algorithm

Input : $\{x_i\}_{0 \le i \le 3}$ such that $x = \sum x_i \bmod 2^k$

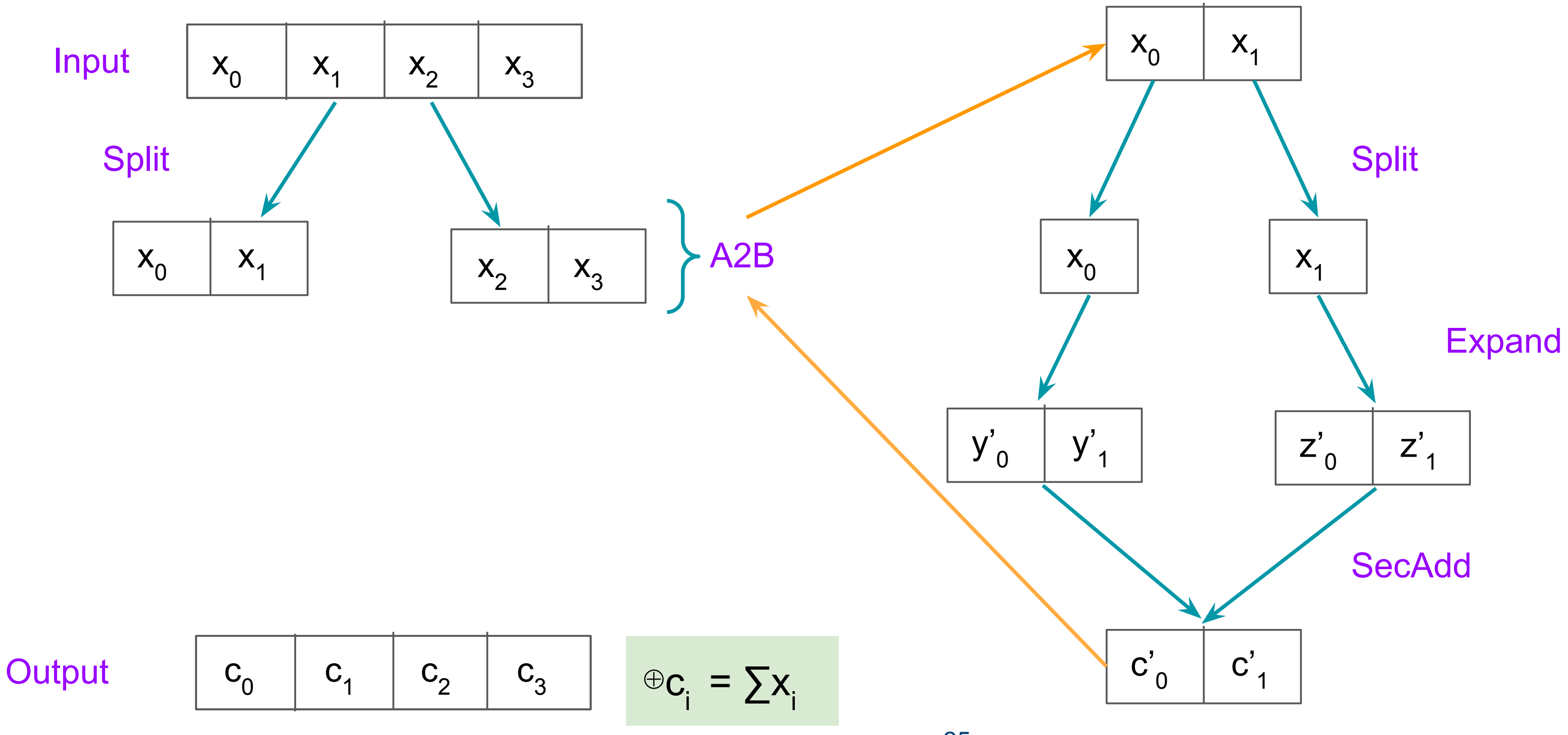
Output : $\{c_i\}_{0 \le i \le 3}$ such that $\oplus c_i = \sum x_i$



Steps of A2B conversion algorithm

Input : $\{x_i\}_{0 \le i \le 3}$ such that $x = \sum x_i \bmod 2^k$

Output : $\{c_i\}_{0 \le i \le 3}$ such that $\oplus c_i = \sum x_i$

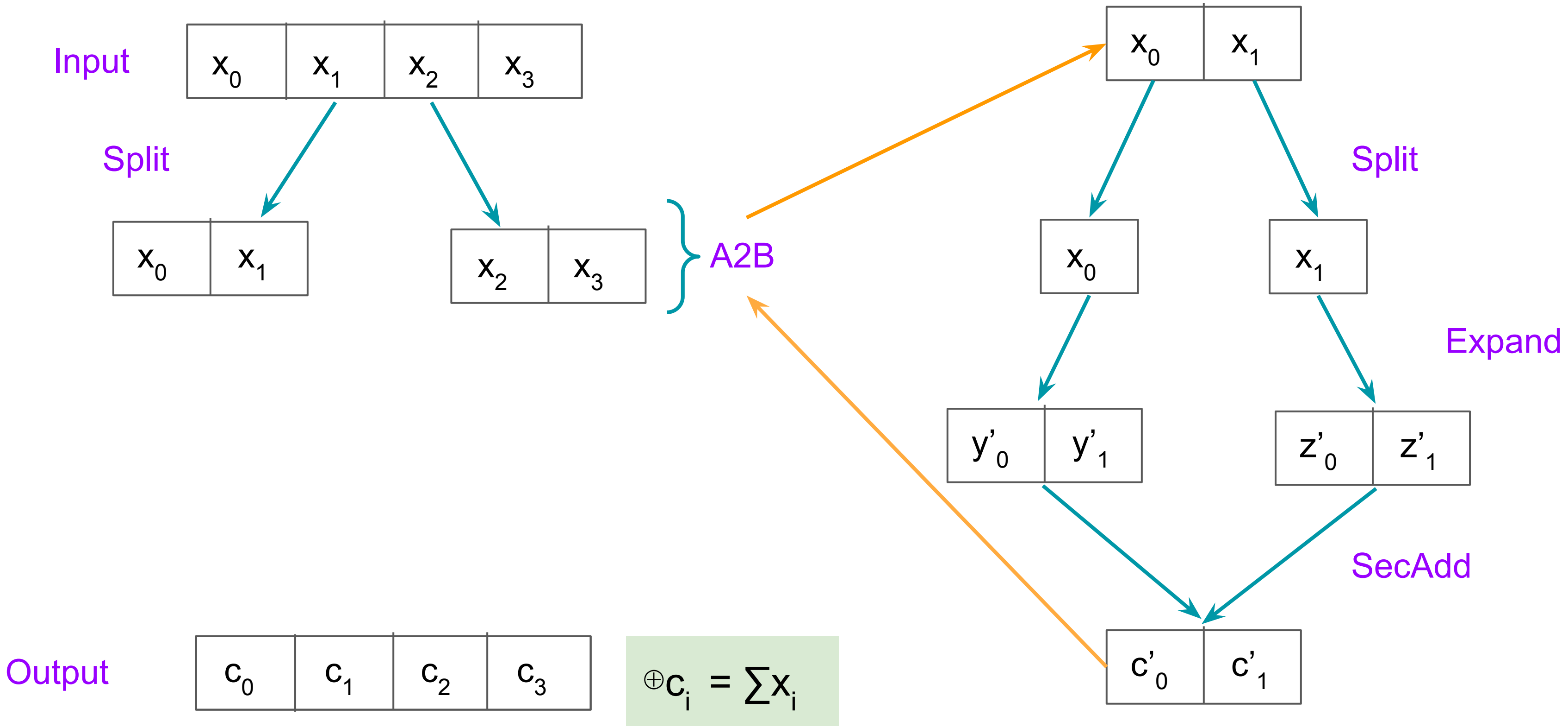


KU LEUVEN

Steps of A2B conversion algorithm

Input : $\{x_i\}_{0 \le i \le 3}$ such that $x = \sum x_i \bmod 2^k$

Output : $\{c_i\}_{0 \le i \le 3}$ such that $\oplus c_i = \sum x_i$



Expand

$x_0 = y'_0 \oplus y'_1$

$x_1 = z'_0 \oplus z'_1$

SecAdd

$(x_0 + x_1) \bmod 2^k =$

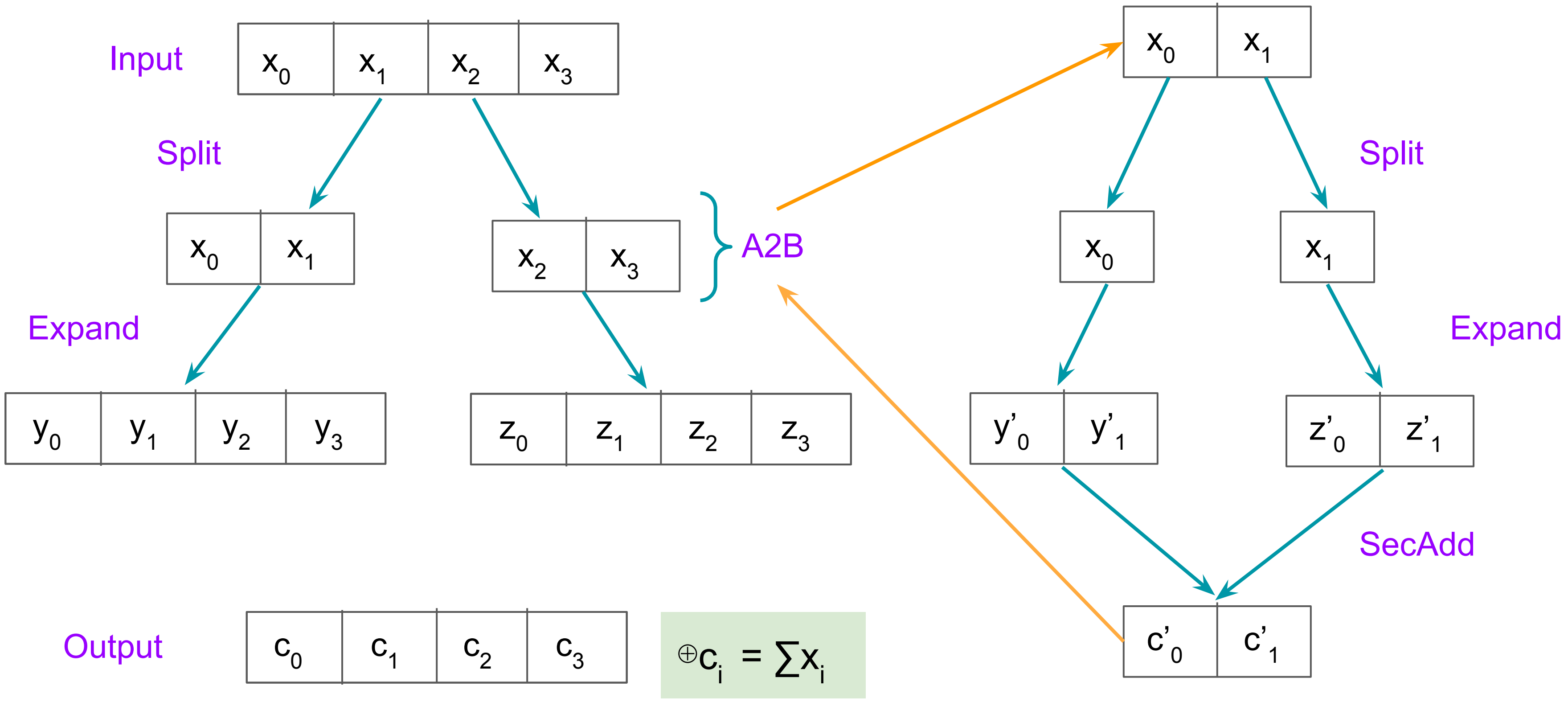
$(y'_0 \oplus y'_1) + (z'_0 \oplus z'_1) \bmod 2^k$

$= c'_0 \oplus c'_1$

Steps of A2B conversion algorithm

Input : $\{x_i\}_{0\le i\le 3}$ such that $x = \sum x_i \bmod 2^k$

Output : $\{c_i\}_{0\le i\le 3}$ such that $\oplus c_i = \sum x_i$



Expand

$x_0 = y'_0 \oplus y'_1$

$x_1 = z'_0 \oplus z'_1$

SecAdd

$(x_0 + x_1) \bmod 2^k =$

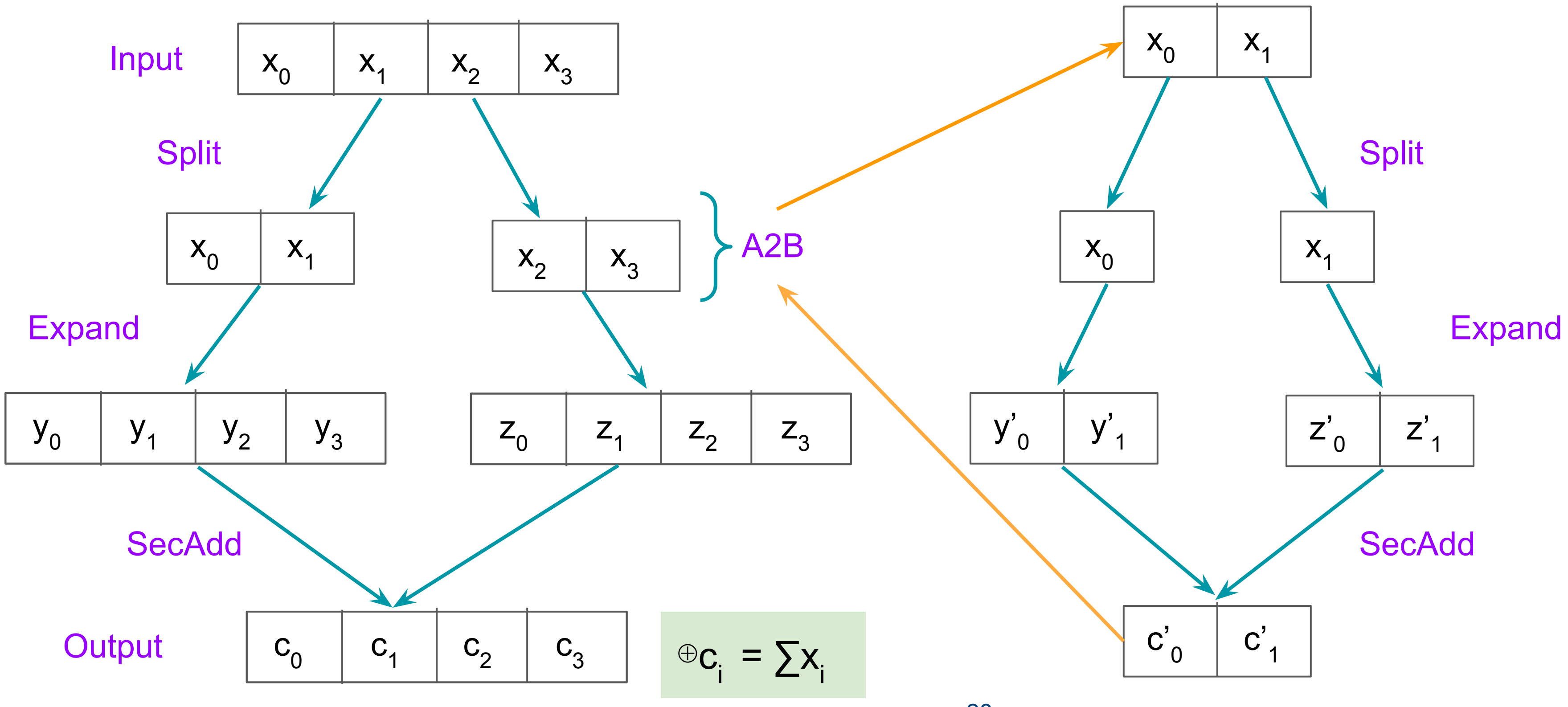
$(y'_0 \oplus y'_1) + (z'_0 \oplus z'_1) \bmod 2^k$

$= c'_0 \oplus c'_1$

Steps of A2B conversion algorithm

Input : $\{x_i\}_{0 \le i \le 3}$ such that $x = \sum x_i \bmod 2^k$

Output : $\{c_i\}_{0 \le i \le 3}$ such that $\oplus c_i = \sum x_i$



Expand

$x_0 = y'_0 \oplus y'_1$

$x_1 = z'_0 \oplus z'_1$

SecAdd

$(x_0 + x_1) \bmod 2^k =$

$(y'_0 \oplus y'_1) + (z'_0 \oplus z'_1) \bmod 2^k$

$= c'_0 \oplus c'_1$

Compression operation in Saber is just shift operation

Input : $\{x_i\}_{0 \le i \le t}$ such that $x = \sum x_i \bmod 2^k$

Output : $\{m_i\}_{0 \le i \le t}$ such that $\oplus m_i = \text{MSB}(\sum x_i \bmod 2^k)$

1. $\{c_i\}_{0 \le i \le t} \leftarrow$ A2B ($\{x_i\}_{0 \le i \le t}$)
2. $\{m_i\}_{0 \le i \le t} \leftarrow$ MSB ($\{c_i\}_{0 \le i \le t/2}$)
3. return $\{m_i\}_{0 \le i \le t}$

Compression operation in Kyber has lot more steps than Saber

Input : $\{x_i\}_{0 \le i \le t}$ such that $x = \sum x_i \bmod q$

Output : $\{m_i\}_{0 \le i \le t}$ such that $\oplus m_i$ = Compression($\sum x_i \bmod q$)

1. $x_0 \leftarrow x_0 - \lfloor q/4 \rfloor$
2. $\{y_i\}_{0 \le i \le t} \leftarrow$ transform-power-of-2 ($\{x_i\}_{0 \le i \le t}$, 13)
3. $y_0 \leftarrow y_0 - \lfloor q/2 \rfloor$
4. $\{c_i\}_{0 \le i \le t} \leftarrow$ A2B ($\{y_i\}_{0 \le i \le t}$)
5. $\{m_i\}_{0 \le i \le t} \leftarrow$ MSB ($\{c_i\}_{0 \le i \le t/2}$)
6. return $\{m_i\}_{0 \le i \le t}$

Future works

- Masking friendly post-quantum schemes
- Improve the performances of masking building block
- Reduce random bytes requirements in masked scheme

Publication:

Suparna Kundu, Jan-Pieter D’Anvers, Michiel Van Beirendonck, Angshuman Karmakar, Ingrid Verbauwhede “Higher-order masked Saber”. SCN 2022.

Thank you!