KU LEUVEN

Techniques for Masking Saber and Kyber

Michiel Van Beirendonck Jan-Pieter D'Anvers

June 2021

Techniques for Masking Saber and Kyber

Synthesis presentation of two works

- **M. Van Beirendonck**, J.-P. D'anvers, A. Karmakar, J. Balasch, and I. Verbauwhede. 2021. A Side-Channel-Resistant Implementation of SABER. J. Emerg. Technol. Comput. Syst. 17, 2. [BDK+21]
- T. Fritzmann, **M. Van Beirendonck**, D. B. Roy, P. Karl, T. Schamberger, I. Verbauwhede, and G. Sigl. 2021. Masked Accelerators and Instruction Set Extensions for Post-Quantum Cryptography. Cryptology ePrint Archive. 2021/479. [FBR+21]

And related approaches

- [OSPG18, SPOG19, BGR+21] ...

Today's focus

Masking

- Technique to protect against DPA

Saber & Kyber : MLW(E/R)-based KEM finalists

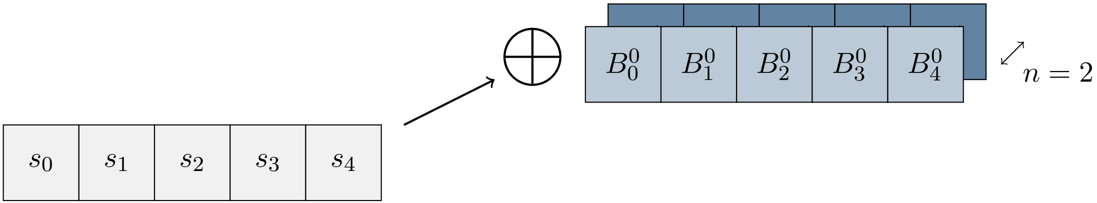
- KeyGen, Encaps, **Decaps**

In our experiments, we found Saber easier and more efficient to mask

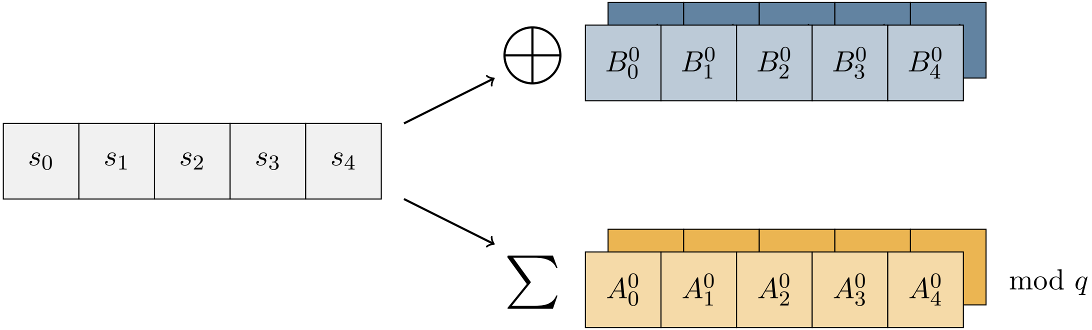
- Due to

Saber	Kyber
$q = 2^{13}$	$q = 3329$
MLWR	MLWE

Masking

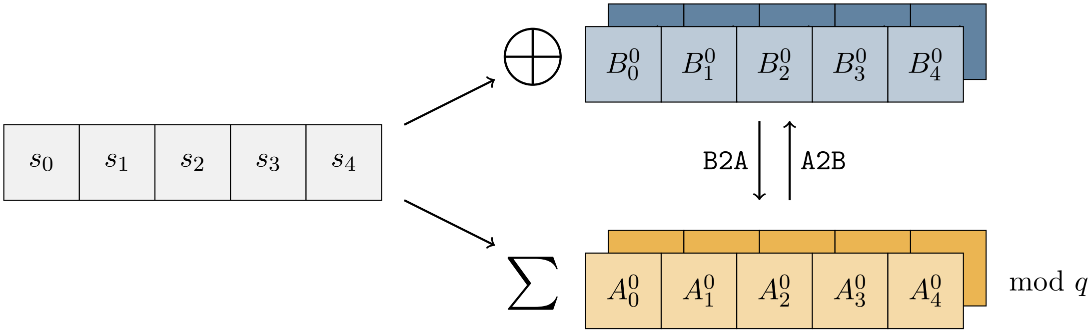


Masking



s_0 s_1 s_2 s_3 s_4
$\bigoplus$
B_0^0 B_1^0 B_2^0 B_3^0 B_4^0
$\sum$
A_0^0 A_1^0 A_2^0 A_3^0 A_4^0
$\bmod q$

Masking



s_0
s_1
s_2
s_3
s_4
$\bigoplus$
B_0^0
B_1^0
B_2^0
B_3^0
B_4^0
B2A
A2B
$\sum$
A_0^0
A_1^0
A_2^0
A_3^0
A_4^0
mod q

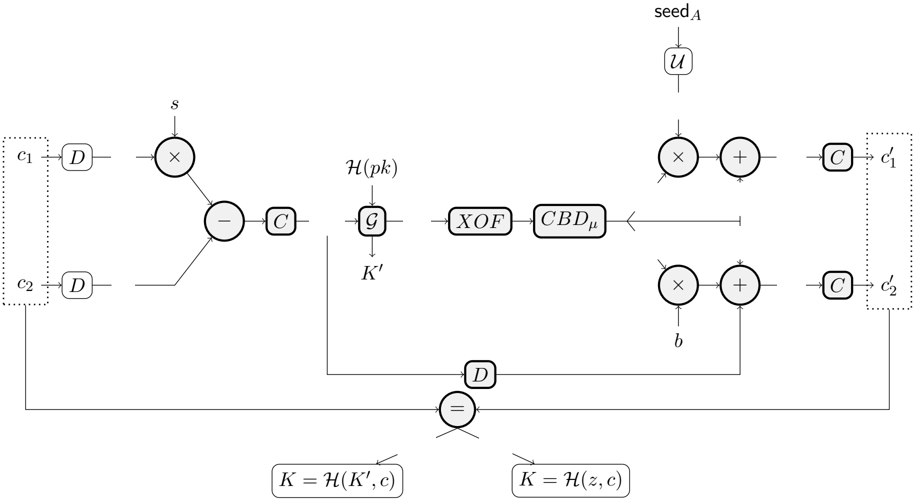
B2A and A2B

More efficient for power-of-two $q = 2^k$ than prime q

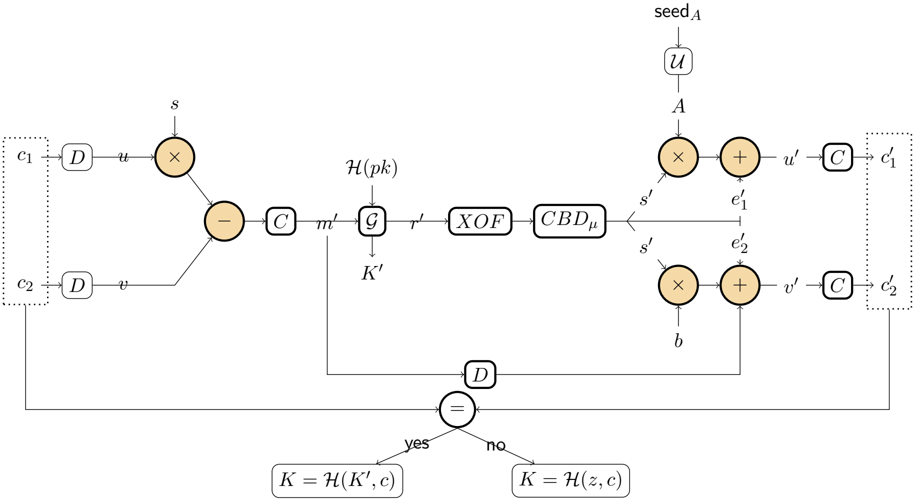
Algorithms

- In [BDK+21]: Goubin's $\texttt{B2A}_{2^k}$ [Gou01], table-based $\texttt{A2B}_{2^k}$ [Deb12, VBDV21]
  - Efficient first-order software masking
- In [BGR+21]: $\texttt{SecAdd}$-based $\texttt{B2A}$, $\texttt{A2B}$ [CGV14]
  - Common hardware for $\texttt{B2A}_{\{2^k,q\}}$, $\texttt{A2B}_{\{2^k,q\}}$
  - Efficient hardware with Threshold Implementations
  - Extensible to higher-order masking
- Additionally in this presentation: $\texttt{SecB2A}_q$ [SPOG19]

Decapsulation : Decrypt and Re-encrypt



Polynomial Arithmetic



Polynomial Arithmetic

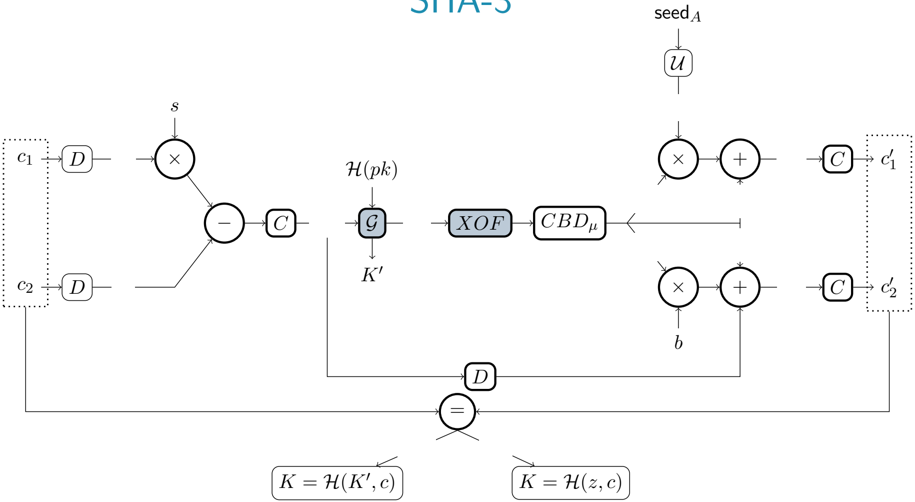
Easy to protect using arithmetic masking

Small overhead factors

- $(n = 2)$: 1.7* [BGR+21] - 2.0† [BDK+21]
- $(n = 3)$: 2.96* [BGR+21]

* with amortized precomputation
† w/o amortized precomputation, precomputation possible using techniques from [MKV20] or [CHK+21]

SHA-3



seed$_A$
$\mathcal{U}$
s
c_1
D
$\times$
$-$
C
$\mathcal{H}(pk)$
$\mathcal{G}$
K'
XOF
CBD_μ
$\times$
$+$
C
c'_1
c_2
D
$\times$
$+$
C
c'_2
b
D
$=$
$K = \mathcal{H}(K', c)$
$K = \mathcal{H}(z, c)$

SHA-3

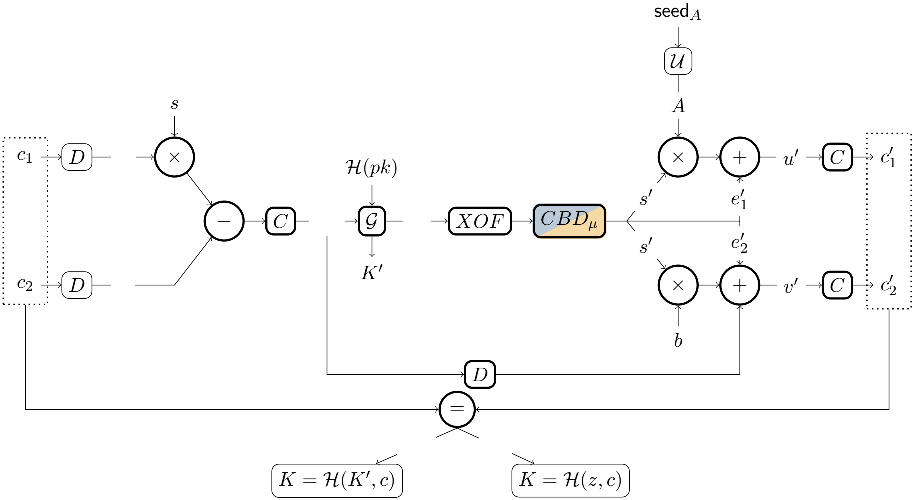
Typically protected using Boolean masking [BDPVA10, BBD+16]

Overhead factors

- $n = 2$: 5.9* [BGR+21] - 9.26† [BDK+21]
- $n = 3$: 73.1* [BGR+21]

*w.r.t plain-C
†w.r.t optimized assembly

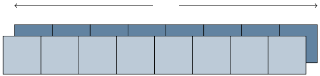
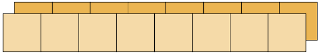
Binomial Sampling



KU LEUVEN

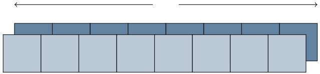
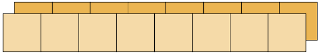
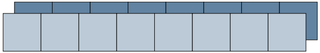
Binomial Sampling

Add/Sub 2μ Boolean masked bits

- $y =$ `BitAddSub`($\bigoplus$ )
- Naive approach needs 2μ `B2A` conversions
  - $y =$ `BitAddSub`()

Binomial Sampling

Add/Sub 2μ Boolean masked bits

- $y = \texttt{BitAddSub}(\bigoplus$  $)$
- Naive approach needs 2μ $\texttt{B2A}$ conversions
  - $y = \texttt{BitAddSub}($  $)$
- Use masked half-adders [SPOG19]
  - $y = \texttt{B2A}(\texttt{SecBitAddSub}($  $))$

MLWE vs MLWR in Masking

	XOF	CBD_μ		
	# Keccak-f	# poly	`SecBitAddSub`	`B2A`
{Light/·/Fire}Saber	**4/5/5**	$\boldsymbol{l}$	$\mu = \{5/4/3\}$	$\mathbf{2^k}$
Kyber{512/768/1024}	7/7/9	$2l+1$	$\boldsymbol{\mu = \{3/2/2\}}$	q

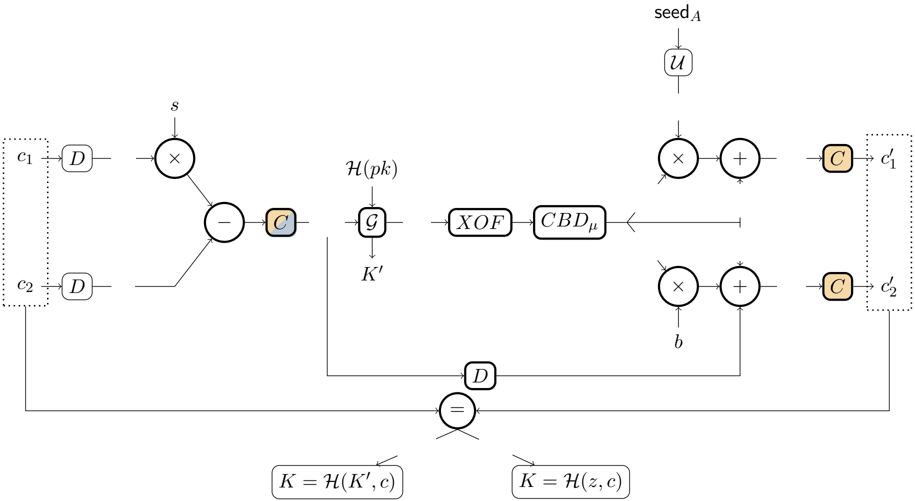
MLWE vs MLWR in Masking

	XOF	CBD_μ		
	# Keccak-f	# poly	`SecBitAddSub`	`B2A`
{Light/·/Fire}Saber	**4/5/5**	$\boldsymbol{l}$	$\mu = \{5/4/3\}$	$\boldsymbol{2^k}$
Kyber{512/768/1024}	7/7/9	$2l+1$	$\boldsymbol{\mu = \{3/2/2\}}$	q

	XOF	CBD_μ		
ARM Cortex-M4 cycles ($n=2$)	Keccak-f	`PolySecBitAddSub`	`PolyB2A`	Total
Saber [BDK+21]	5×123k	3×50k	3×17k	**815k (1.00x)**
Kyber768	7×123k	7×32k	$7 \times 118\text{k}^{\dagger}$	**1914k (2.35x)**

† we use $\texttt{SecB2A}_q$ [SPOG19] for this experiment, more efficient than $\texttt{SecAdd-B2A}_q$ in software

$\texttt{Compress}_q$



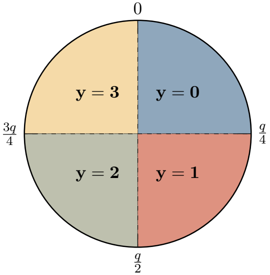
KU LEUVEN

$\texttt{Compress}_q$

$$y = \texttt{Compress}'_q(x, d) = \lfloor (2^d/q) \cdot x \rceil \bmod 2^d$$

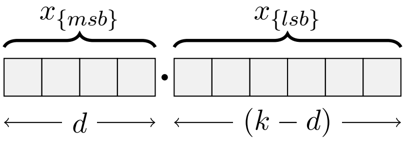
Interval comparison with 2^d intervals

- 2^2 intervals on the right

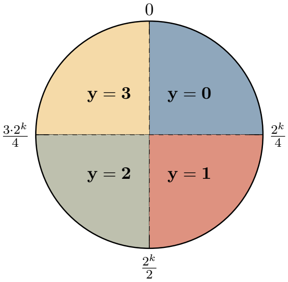


$\texttt{Compress}_{2^k}$ - Saber

$$y = \texttt{Compress}'_{2^k}(x, d) = \lfloor (2^d/2^k) \cdot x \rceil \bmod 2^d$$

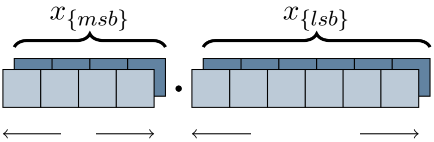
- $x =$ 

- $y = x_{\{msb\}}$



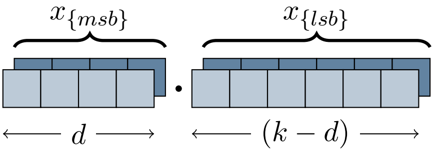
$\texttt{MaskedCompress}_{2^k}$ - Saber

$$y = \texttt{Compress}'_{2^k}(x, d) = \lfloor (2^d / 2^k) \cdot x \rfloor \bmod 2^d$$

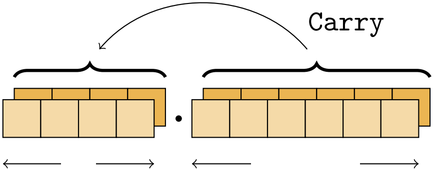
- $x =$  $, \; y = x_{\{msb\}}$

$\texttt{MaskedCompress}_{2^k}$ - Saber

$$y = \texttt{Compress}'_{2^k}(x, d) = \lfloor (2^d/2^k) \cdot x \rfloor \bmod 2^d$$

- $x =$ 

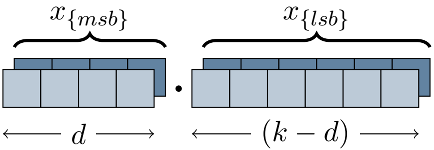
$, \; y = x_{\{msb\}}$

- $x =$ 

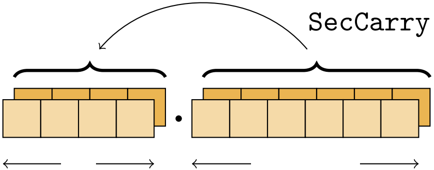
$,$

MaskedCompress$_{2^k}$ - Saber

$$y = \texttt{Compress}'_{2^k}(x, d) = \lfloor (2^d/2^k) \cdot x \rceil \bmod 2^d$$

- $x =$ 

$, \; y = x_{\{msb\}}$

- $x =$ 

$, \; y = x_{\{msb\}} + \texttt{SecCarry}(x_{\{lsb\}})^{\dagger}$ [BDK+21]

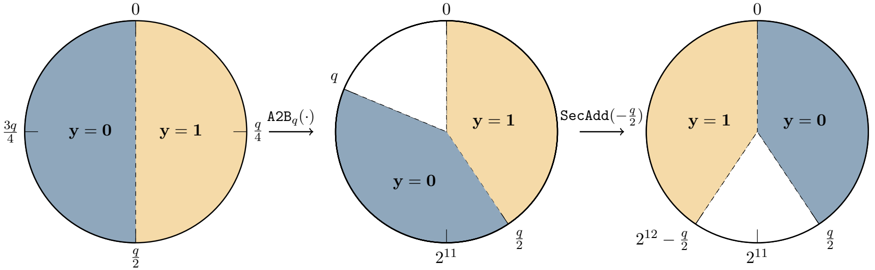
† SecCarry is a pruned A2B conversion

KU LEUVEN

MaskedDecode $\equiv$ MaskedCompress$_q(x, 1)$ - Kyber

[OSPG18]: Transform$_{2^k}$ and A2B$_{2^k}$
[FBR$^+$21]: A2B$_q$ and SecAdd$(-\frac{q}{2})$
[BGR$^+$21]: A2B$_q$ and BitSliceSecSearch

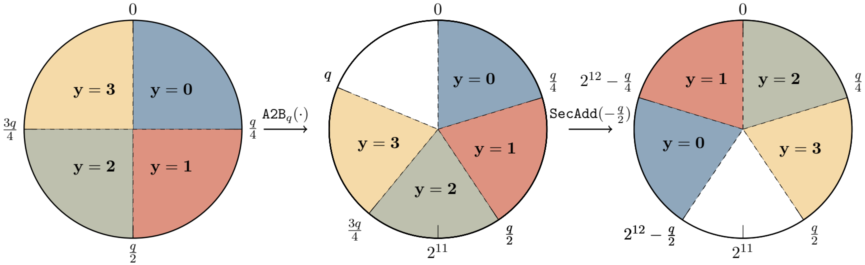
- SecSearch [BGR$^+$21] $\equiv$ MSB(SecConstAdd$(x, -\frac{q}{2})$) [FBR$^+$21]



$\texttt{MaskedCompress}_q(x, 2)$? - Kyber

[OSPG18]: $\texttt{Transform}_{2^k}$ and $\texttt{A2B}_{2^k}$
[FBR$^+$21]: $\texttt{A2B}_q$ and $\texttt{SecAdd}(-\frac{q}{2})$
[BGR$^+$21]: $\texttt{A2B}_q$ and $\texttt{BitSliceSecSearch}$

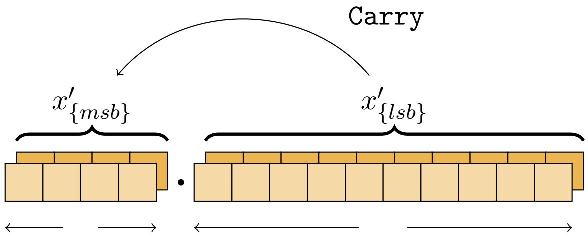
- $(2^{12} - \frac{q}{4})$ and $\frac{q}{4}$ no longer spaced at bit-intervals



$\texttt{MaskedCompress}_q(x, d)$ - Kyber

$$y = \texttt{Compress}'_q(x,d) = \lfloor x' \rceil \bmod 2^d \ , \ x' = (2^d/q) \cdot x$$

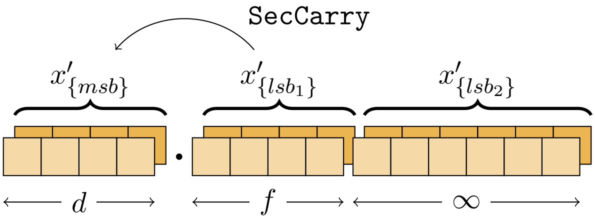
- $x' =$



$\texttt{MaskedCompress}_q(x, d)$ - Kyber

$$y = \texttt{Compress}'_q(x, d) = \lfloor x' \rceil \bmod 2^d \;,\; x' = (2^d/q) \cdot x$$

- $x' =$

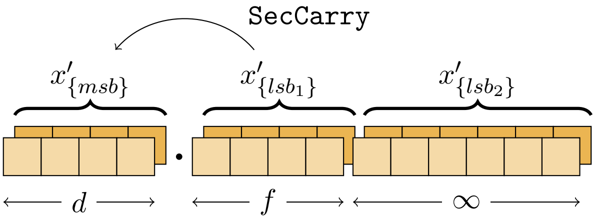


- Only need f fractional bits $x'_{\{lsb_1\}}$ to determine carry† [FBR+21]

† f increases (logarithmically) with the number of shares

$\texttt{MaskedCompress}_q(x, d)$ - Kyber

$$y = \texttt{Compress}'_q(x, d) = \lfloor x' \rceil \bmod 2^d \ , \ x' = (2^d/q) \cdot x$$

- $x' =$ 

- Only need f fractional bits $x'_{\{lsb_1\}}$ to determine carry† [FBR+21]
  - Since $x'_{\{lsb\}} = (2^d \cdot x \bmod q)/q$ takes only q discrete values

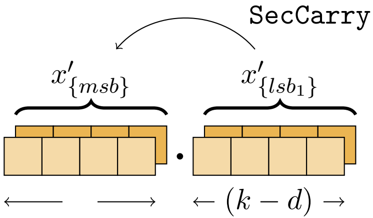
† f increases (logarithmically) with the number of shares

KU LEUVEN

$\texttt{MaskedCompress}(x, d)$

Saber

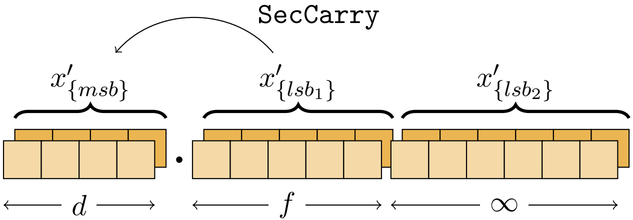
- $x' = (2^d/2^k) \cdot x =$



, $(k-d) \in \{3, 6, 9\}$

Kyber

- $x' = (2^d/q) \cdot x =$



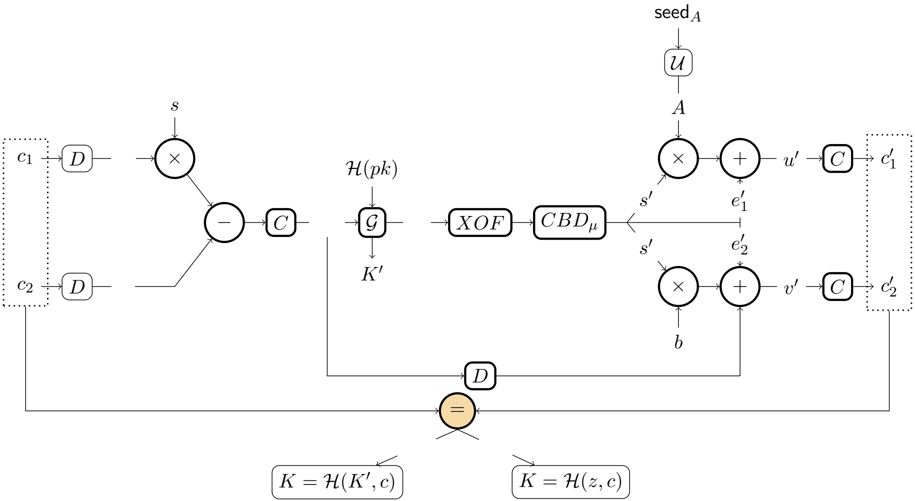
, $f = 13$

$\texttt{MaskedCompress}(x, d)$

	ARM Cortex-M4 cycles ($n = 2$)	PolyMaskedCompress table-A2B	PolyMaskedCompress SecAdd-A2B†
Saber	$(k-d) = 3$	$3 \times 14.5k$	3×374k
	$(k-d) = 6$	$17k$	594k
	$(k-d) = 9$	$19k$	814k
		79.5k(1.00x)	**2530k(1.00x)**
Kyber	$f = 13$	$5 \times 24k$	5×1098k
		120k(1.51x)	**5490k(2.17x)**

† unoptimized reference implementation

MaskedComparison



KU LEUVEN

`MaskedComparison`

$n = 2$:

- `HashComparison` [OSPG18] with fix [BDK+21, BDH+21].

$n > 2$:

- `MaskedSum` [BPO+20] with `ReduceComparisons` fix [BDH+21]
- `DecompressedComparison` [BGR+21]

MaskedComparison

$n = 2$:

- HashComparison [OSPG18] with fix [BDK+21, BDH+21].

$n > 2$:

- MaskedSum [BPO+20] with ReduceComparisons fix [BDH+21]
  - Not a full comparison
  - Doesn't work with compression
- DecompressedComparison [BGR+21]
  - One of the motivations: no existing MaskedCompress
- Interesting to consolidate approaches

Results

Algorithm	Device	Decapsulation unmasked	masked
Saber [BDK+21]	ARM M4	$1,123,280$	$2,833,348\ (\times 2.52)$
Kyber [BGR+21]	ARM M0+	$5,530,000$	$12,208,000\ (\times 2.21)^*$
Saber [FBR+21]	RISC-V	$347,323$	$914,925\ (\times 2.63)$
Kyber [FBR+21]	RISC-V	$338,746$	$1,402,650\ (\times 4.14)^\dagger$

Future work

- More efficient higher-order methods
- Saber on M0+, Kyber on M4 for a better comparison

*randomness sampling not included
†randomness sampling included: 167k cycles (17.5x Saber due to more random bits and rejection sampling)

Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini.
Strong non-interference and type-directed higher-order masking.
In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 116–129, New York, NY, USA, 2016. Association for Computing Machinery.

Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck.
Attacking and defending masked polynomial comparison for lattice-based cryptography.
Cryptology ePrint Archive, Report 2021/104, 2021.

Michiel Van Beirendonck, Jan-Pieter D'anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede.
A side-channel-resistant implementation of saber.
J. Emerg. Technol. Comput. Syst., 17(2), April 2021.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
Building power analysis resistant implementations of Keccak.
In *Second SHA-3 candidate conference*, volume 142, 2010.

Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal.
Masking kyber: First- and higher-order implementations.
Cryptology ePrint Archive, Report 2021/483, 2021.

Florian Bache, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu.
High-speed masking for polynomial comparison in lattice-based KEMs.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3):483–507, Jun. 2020.

Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala.
Secure conversion between Boolean and arithmetic masking of any order.
In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 188–205. Springer, 2014.

Chi-Ming Marvin Chung, Vincent Hwang, Matthias J Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang.
NTT multiplication for NTT-unfriendly rings.
IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 159–188, 2021.

Blandine Debraize.
Efficient and provably secure methods for switching from arithmetic to Boolean masking.
In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 107–121. Springer, 2012.

Tim Fritzmann, Michiel Van Beirendonck, Debapriya Basu Roy, Patrick Karl, Thomas Schamberger, Ingrid Verbauwhede, and Georg Sigl.
Masked accelerators and instruction set extensions for post-quantum cryptography.
Cryptology ePrint Archive, Report 2021/479, 2021.

Louis Goubin.
A sound method for switching between Boolean and arithmetic masking.
In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 3–15. Springer, 2001.

Jose Maria Bermudo Mera, Angshuman Karmakar, and Ingrid Verbauwhede.
Time-memory trade-off in Toom-Cook multiplication: an application to module-lattice based cryptography.
IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 222–244, 2020.

Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu.
Practical CCA2-secure and masked ring-LWE implementation.
IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 142–174, 2018.

Tobias Schneider, Clara Paglialonga, Tobias Oder, and Tim Güneysu.
Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto.
In *IACR International Workshop on Public Key Cryptography*, pages 534–564. Springer, 2019.

Michiel Van Beirendonck, Jan-Pieter D'Anvers, and Ingrid Verbauwhede.
Analysis and comparison of table-based arithmetic to Boolean masking.
IACR Cryptol. ePrint Arch., 2021:67, 2021.