X2X: Efficient A2B & B2A Conversions for $d + 1$ Shares in Hardware

with Application to Lattice-based PQC

CASCADE '25

Q. Norga, S. Kundu, JP. D'Anvers, I. Verbauwhede

COSIC, KU Leuven

April 3, 2025

Outline

1 Introduction to PQC & Masking

2 Algorithmic Improvements

3 Implementation & Evaluation

4 Conclusion

Outline

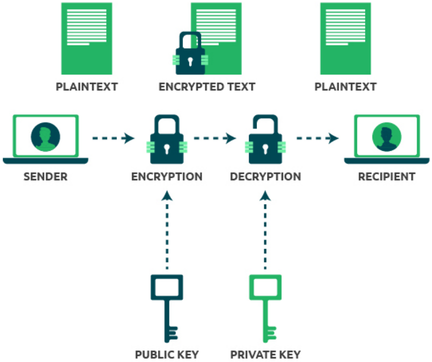
1 Introduction to PQC & Masking

2 Algorithmic Improvements

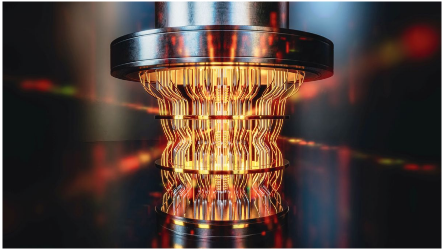
3 Implementation & Evaluation

4 Conclusion

Post-Quantum Cryptography



SOURCE: ClickSSL



SOURCE: ORF, Getty

Lattice-based PQC

ML-KEM & ML-DSA

Performance, security and bandwidth

FIPS 203

Federal Information Processing Standards Publication

Module-Lattice-Based Key-Encapsulation Mechanism Standard

Category: Computer Security   Subcategory: Cryptography

FIPS 204

Federal Information Processing Standards Publication

Module-Lattice-Based Digital Signature Standard

Category: Computer Security   Subcategory: Cryptography

Lattice-based PQC

ML-KEM & ML-DSA

Performance, security and bandwidth

Real-world deployment:

(Protection against) **Physical attacks**

FIPS 203

Federal Information Processing Standards Publication

Module-Lattice-Based Key-Encapsulation Mechanism Standard

Category: Computer Security　　Subcategory: Cryptography

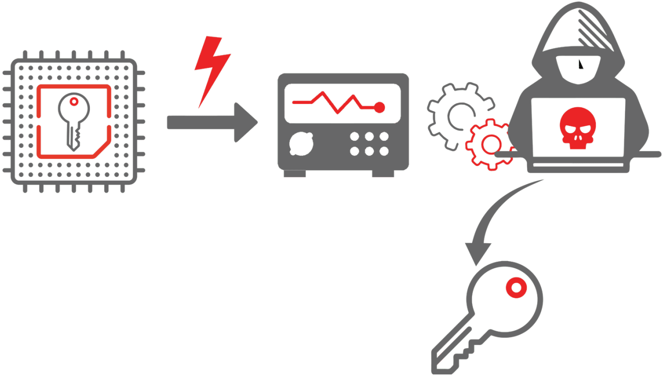
FIPS 204

Federal Information Processing Standards Publication

Module-Lattice-Based Digital Signature Standard

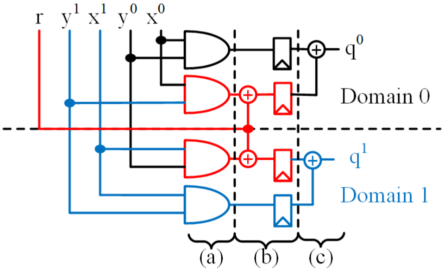
Category: Computer Security　　Subcategory: Cryptography

Side-Channel Attacks



SOURCE: Secure-iC

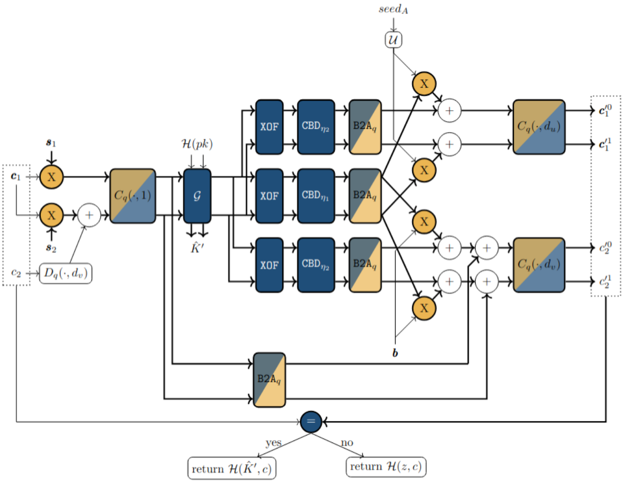
Masking



RAND & REG

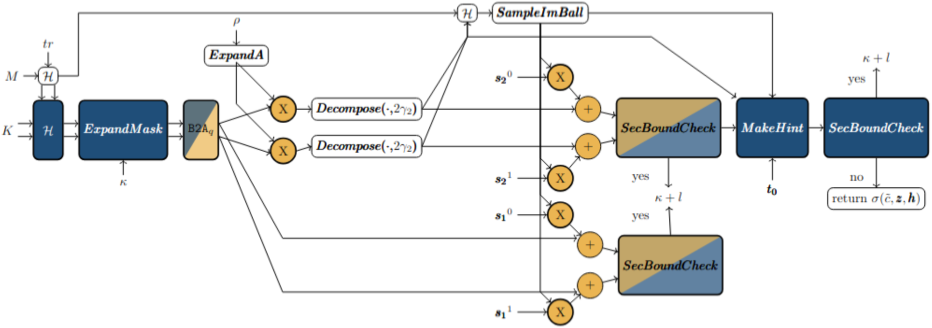
COSIC

Masking ML-KEM.Decaps



COSIC

Masking ML-DSA.Sign



COSIC

Masking Lattice-based PQC

Masking Lattice-based PQC requires a mix of arithmetic and Boolean sharing.

- Polynomial arithmetic (e.g., PolMult): $x = \sum_{i=0}^{d} x^{\{i\}}$
- Bitwise arithmetic (e.g., Hashing): $x = \bigoplus_{i=0}^{d} x^{\{i\}}$

Need **A2B** and **B2A**!

This Work: **X2X**

Full `ML-KEM.Decaps` or `ML-DSA.Sign` requires:

- **ANY** protection order d
- **ANY** modulus p or q
- **ANY** operation (`A2B` or `B2A`)

- Low cost (randomness, area)
- High performance (throughput)

Outline

1 Introduction to PQC & Masking

2 Algorithmic Improvements

3 Implementation & Evaluation

4 Conclusion

Secure Addition: SecADD

$$s^{\{0:d\}} = x^{\{0:d\}} + y^{\{0:d\}} \bmod q = \bigoplus_{i=0}^{d} x^{\{i\}} + \bigoplus_{i=0}^{d} y^{\{i\}} \bmod q$$

- "Arithmetic addition on Boolean shares"

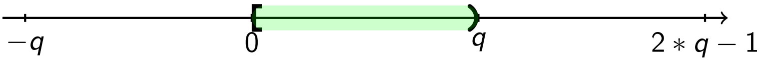
Secure Addition: SecADD

$$s^{\{0:d\}} = x^{\{0:d\}} + y^{\{0:d\}} \mod q = \bigoplus_{i=0}^{d} x^{\{i\}} + \bigoplus_{i=0}^{d} y^{\{i\}} \mod q$$

- "Arithmetic addition on Boolean shares"

$\mathtt{SecADD}_q$: Typical Approach

$$\bigoplus_{i=0}^{d} x^{\{i\}}, \quad \bigoplus_{i=0}^{d} y^{\{i\}}$$

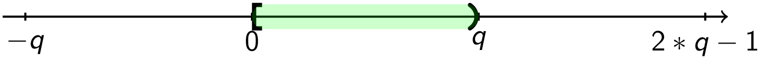


$\mathtt{SecADD}_q$: Typical Approach

Step 1: $s^{\{0:d\}} = \bigoplus_{i=0}^{d} x^{\{i\}} + \bigoplus_{i=0}^{d} y^{\{i\}}$

$\texttt{SecADD}_q$: Typical Approach

$$\textit{Step 2:} \quad s^{\{0:d\}} = \bigoplus_{i=0}^{d} x^{\{i\}} + \bigoplus_{i=0}^{d} y^{\{i\}} \mod q$$



- `SecMUX` [1] or $2 \times$ `SecADD` [2]

A2B

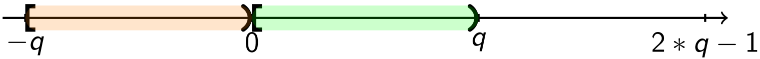
- A2B $\approx$ SecADD(SecADD($\cdots$))

$$B^{\{0:d\}} = z^{\{0\}} + z^{\{1\}} + \ldots + z^{\{d\}}$$

- $\uparrow d \quad \rightarrow \quad \uparrow \#$ SecADD

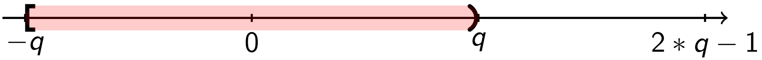
$\texttt{SecADDChain}_q$

Step 0: $\bigoplus_{i=0}^{d} x^{\{i\}}, \quad \bigoplus_{i=0}^{d} y^{\{i\}} - q$



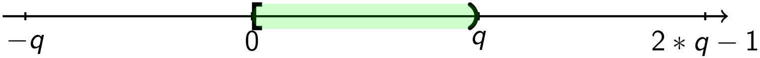
$\texttt{SecADDChain}_q$

Step 1: $s^{\{0:d\}} = \bigoplus_{i=0}^{d} x^{\{i\}} + \bigoplus_{i=0}^{d} y'^{\{i\}}$



$\texttt{SecADDChain}_q$

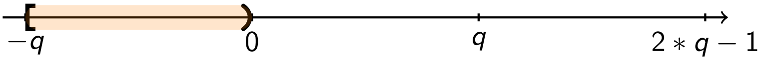
Step 2: $s^{\{0:d\}} = \bigoplus_{i=0}^{d} x^{\{i\}} + \bigoplus_{i=0}^{d} y^{\{i\}} \mod q$



- **1** × SecADD

$\texttt{SecADDChain}_q$

Step 2: $s'^{\{0:d\}} = s^{\{0:d\}} - q$



- **1** $\times$ `SecADD`
- Interleave 2 options

B2A

- $\texttt{B2A} \approx \texttt{A2B}$ & $\texttt{SecADD}^d$

🏭 : $R^0, \quad R^1 \quad \cdots \quad R^{d-1}$

B2A

- B2A ≈ A2B & $\texttt{SecADD}^d$

🏭 : $R^0, \quad R^1 \quad \cdots \quad R^{d-1}$

$$B^{\{0:d\}} = R^{\{0\}} + R^{\{1\}} + \ldots + 0$$

B2A

- B2A ≈ A2B & $\texttt{SecADD}^d$

🏭 : $R^0, \quad R^1 \quad \cdots \quad R^{d-1}$

$$B^{\{0:d\}} = R^{\{0\}} + R^{\{1\}} + \ldots + 0$$

$$z^{\{0:d\}} = B^{\{0:d\}} + x^{\{0:d\}}$$

B2X2A & X2B

- B2X2A $\approx$ X2B

 $: R^0, \quad R^1 \quad \cdots \quad R^{d-1}$

B2X2A & X2B

- B2X2A $\approx$ X2B

🏭 : $R^0, \quad R^1 \quad \cdots \quad R^{d-1}$

$z^{\{0:d\}} = R^{\{0\}} + R^{\{1\}} + \ldots + x^{\{0:d\}}$

B2X2A & X2B

- B2X2A ≈ X2B

$$: R^0, \quad R^1 \quad \cdots \quad R^{d-1}$$

$$z^{\{0:d\}} = R^{\{0\}} + R^{\{1\}} + \ldots + x^{\{0:d\}}$$

- X2B ≈ SecADD'(SecADD'(⋯))
- Pre- and post-processing: see full paper!

Operation Cost: $\texttt{SecADDChain}_q^d$ & B2X2A

		# SecADD					# SecMUX				
	Order	1	2	3	d	Total	1	2	3	d	Total
[1]	1	4	-	-	-	4	2	-	-	-	2
	2	2	4	-	-	6	1	2	-	-	3
	3	4	-	4	-	8	2	-	2	-	4
	d	-	-	-	4	$2(d+1)$	-	-	-	2	$d+1$
[3]	1	2	-	-	-	2	-	-	-	-	-
[2]	1	2	-	-	-	2	-	-	-	-	-
	2	2	5	-	-	7	-	-	-	-	-
	3	4	0	6	-	10	-	-	-	-	-
	d	-	-	-	5 or 6[a]	$3d$ or $3d+1$[a]	-	-	-	-	-
B2X2A	1	2	-	-	-	2	-	-	-	-	-
	2	2	2	-	-	**4**	-	-	-	-	-
	3	2	0	4	-	**6**	-	-	-	-	-
	d	-	-	-	$2 \cdot \lceil \log_2(d) \rceil$	**2d**	-	-	-	-	-

Table: Detailed $B2A_q$ Operation Cost Comparison ($d+1$ shares, k-bit words).

[a] For *complete* or *incomplete* tree-structure.

COSIC

Outline

1 Introduction to PQC & Masking

2 Algorithmic Improvements

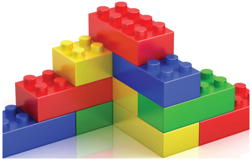
3 Implementation & Evaluation

4 Conclusion

Masking Techniques

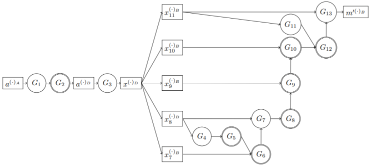
Approach 1: **Universal Composability**

- Masked Gadgets
- (Over)conservative RND & REG



Approach 2: **Manual Masking**

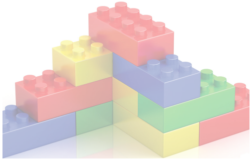
- Masked Gates
- Error-prone



Masking Techniques

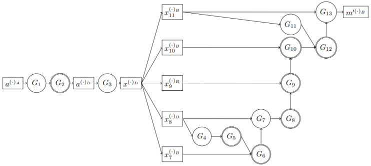
Approach 1: Universal Composability

- Masked Gadgets
- (Over)conservative RND & REG



Approach 2: Manual Masking

- Masked Gates
- Error-prone



COSIC

Masking Techniques: Cost Comparison

Masking Technique	RND [bits]	Latency [cycles]	Verification
HPC1 (PINI)	228	18	**Low**
DOM (t–NI) + `SecREF` (t–SNI)	176	11	*High*
DOM (t–NI)	**114**	**9**	*High*

Table: Comparison of first-order masking techniques of a Brent-Kung `SecADD` ($k = 13$).

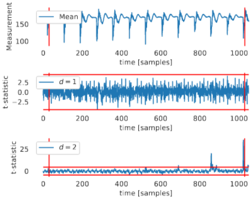
- Half-cycle datapath: see full paper!

Performance Comparison

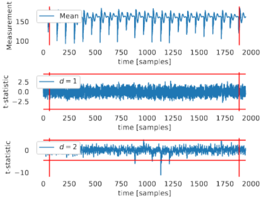
Table 4: Mask Conversion Hardware Implementation: Performance Comparison.

Design	Mask. Tech.	Device	k	d	Util. [LUT/FF]	Freq. [MHz]	OP	mod	Rand.[a] [bits]	Lat. [cycles]	TP [coeff/cycle]
[SMG15]	TI	Spartan-6	32	1	937/1,330	62	SecADD	2^k	32	6	0.167
				2	4,223/5,509	63			128	12	0.083
[FVBBR+21]	TI	Artix-7	32	1	2,464/1,323	454	SecADD	2^k	-	6	-
[BG22]	PINI (HPC)	-[c]	32	1	-	-	SecADD	2^k	122	10	1
				2	-	-			366	10	1
[CGM+23]	PINI (HPC)	Spartan-6	32	1	1,588/4,317	173	SecADD	2^k	74	18	1
				2	1,666/7,122	158			222	18	1
[CGTV15][b]	PINI (HPC)	Artix-7	32	2	13,064/17,952	351	A2B	2^k	1,280	24	1
[BC22][b]	PINI (HPC)	Artix-7	32	2	2,234/20,423	512	A2B	2^k	124	124	0.008
[LZP+24]	PINI (HPC)	Artix-7	32	2	11,196/14,550	370	A2B	2^k	1,056	14	1
This Work (Full-cycle)	**DOM**	Kintex-7[d]	13	1	1,150/3,335	176	A2B	2^k	**140**	10	**2**
								3329	**255**	20	**1**
							B2A	2^k	**140**	11	**2**
								3329	**255**	21	**1**
				2	3,128/16,774	144	A2B	2^k	534	20	**2**
								3329	**993**	40	**1**
							B2A	2^k	**534**	21	**2**
								3329	**993**	41	**1**
This Work (Half-cycle)	**DOM**	Kintex-7[d]	**13**	1	**1,133/2,170**	139	A2B	2^k	**140**	**5**	**2**
								3329	**255**	**10**	**1**
							B2A	2^k	**140**	**5**	**2**
								3329	**255**	**10**	**1**
				2	**3,105/9,376**	130	A2B	2^k	534	**10**	**2**
								3329	**993**	**20**	**1**
							B2A	2^k	**534**	**10**	**2**
								3329	**993**	**20**	**1**

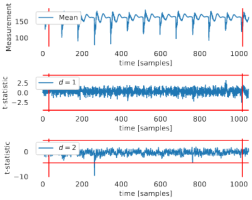
Security Evaluation: TVLA in Lab



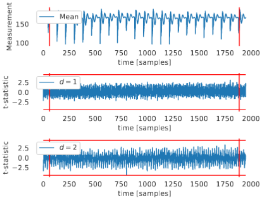
(a) A2B mod 2^{13}.



(b) A2B mod 3329.

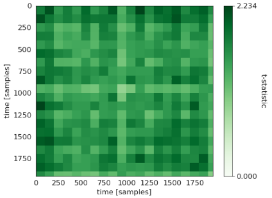


(c) B2A mod 2^{13}.

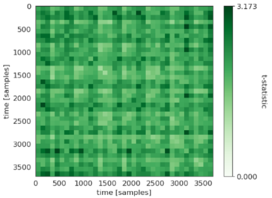


(d) B2A mod 3329.

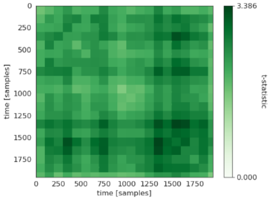
Security Evaluation: TVLA in Lab



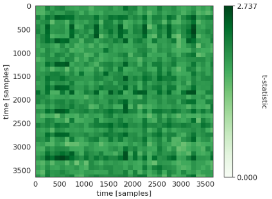
(a) A2B mod 2^{13}.



(b) A2B mod 3329.



(c) B2A mod 2^{13}.



(d) B2A mod 3329.

Outline

1 Introduction to PQC & Masking

2 Algorithmic Improvements

3 Implementation & Evaluation

4 Conclusion

COSIC

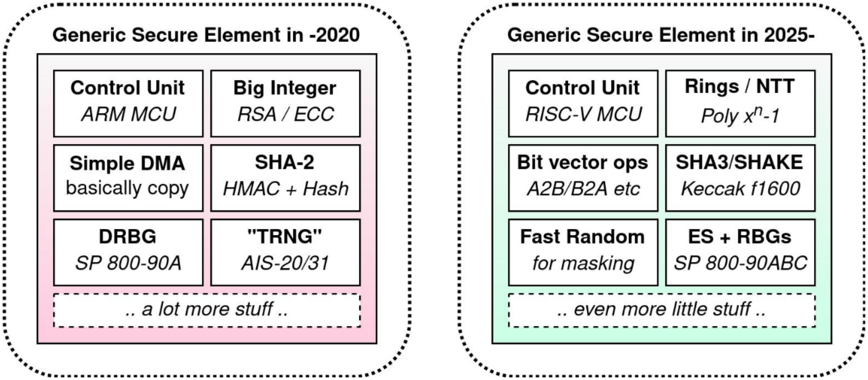
X2X: Summary

Full `ML-KEM.Decaps` or `ML-DSA.Sign` requires:

- **ANY** protection order d ✅
- **ANY** modulus p or q ✅
- **ANY** operation (`A2B` or `B2A`) ✅

- Low cost (randomness, area) ✅ (up to 62%, 45-60%)
- High performance (throughput, latency) ✅ (29-92%)

Future Work



SOURCE: PQShield

Thank you. Questions?

[1] Gilles Barthe et al. "Masking the GLP Lattice-Based Signature Scheme at Any Order". In: *Advances in Cryptology – EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Cham: Springer International Publishing, 2018, pp. 354–384. ISBN: 978-3-319-78375-8.

[2] Gaëtan Cassiers. "Composable and efficient masking schemes for side-channel secure implementations". PhD thesis. École polytechnique de Louvain and Université catholique de Louvain, 2022.

[3] Tim Fritzmann et al. "Masked Accelerators and Instruction Set Extensions for Post-Quantum Cryptography". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2022.1 (Nov. 2021), pp. 414–460. DOI: 10.46586/tches.v2022.i1.414-460. URL: https://tches.iacr.org/index.php/TCHES/article/view/9303.