Higher-Order Masked Ciphertext Comparison for Lattice-Based Cryptography

Jan-Pieter D'Anvers[1] Daniel Heinz[2,3] Peter Pessl[3] **Michiel Van Beirendonck**[1] Ingrid Verbauwhede[1]

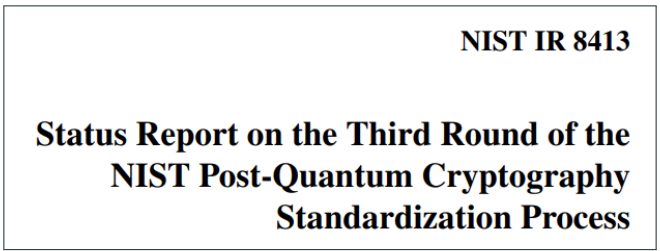
[1]imec-COSIC KU Leuven

[2]Research Institute CODE

[3]Infineon Technologies

Lattice-based Cryptography

NIST selected Kyber as PQC Key Encapsulation Mechanism (KEM) standard.


NIST IR 8413

Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process

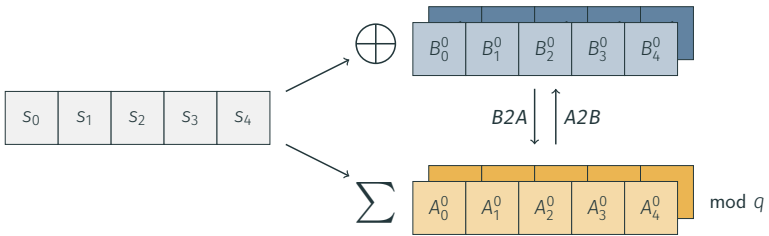
Higher-order Masked Ciphertext Comparison for Lattice-based Cryptography

Higher-order Masking

Implementations require protections again Side-Channel Attacks (SCA)

Higher-order Masking

Implementations require protections again Side-Channel Attacks (SCA)



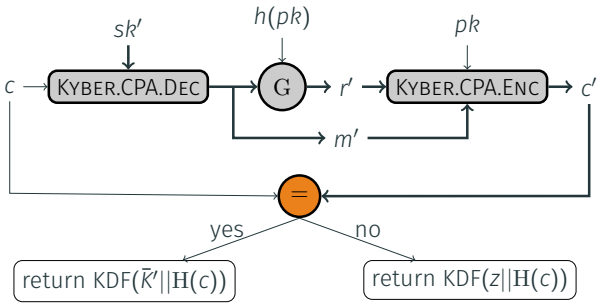
B2A, A2B are key building blocks in masking Lattice-based schemes

Ciphertext Comparison

Kyber uses the FO Transform from IND-CPA PKE to IND-CCA KEM

Ciphertext Comparison

Kyber uses the FO Transform from IND-CPA PKE to IND-CCA KEM



Side-channel Attacks on the FO Transform

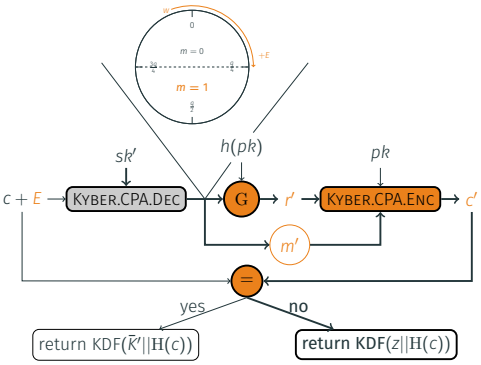
- The FO transform is a target for SCA that builds plaintext-checking oracles

Side-channel Attacks on the FO Transform

- The FO transform is a target for SCA that builds plaintext-checking oracles

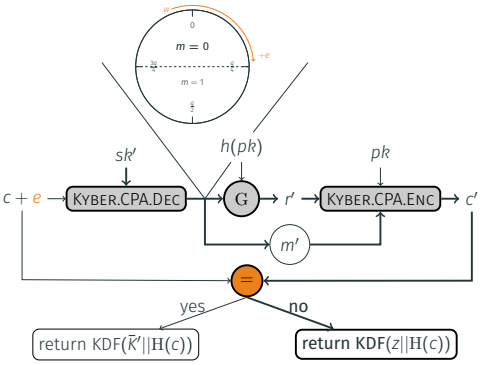
Side-channel Attacks on the FO Transform

- The FO transform is a target for SCA that builds plaintext-checking oracles



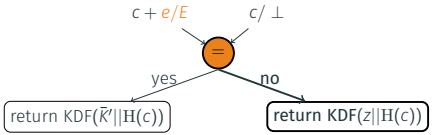
Side-channel Attacks on the FO Transform

- The FO transform is a target for SCA that builds plaintext-checking oracles



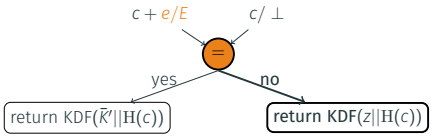
Side-channel Attacks on the FO Transform

- The FO transform is a target for SCA that builds plaintext-checking oracles



Side-channel Attacks on the FO Transform

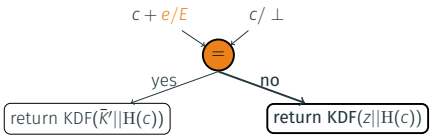
- The FO transform is a target for SCA that builds plaintext-checking oracles



- [BDH+21] Two masked comparisons broken, with fixes proposed:
  1. First-order Hash-based Comparison
  2. Higher-order Reduce-Comparisons
     - Not applicable to Kyber, Saber due to ciphertext compression

Side-channel Attacks on the FO Transform

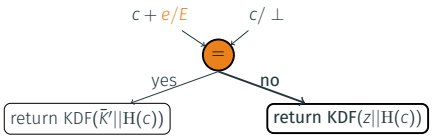
- The FO transform is a target for SCA that builds plaintext-checking oracles



- [BDH+21] Two masked comparisons broken, with fixes proposed:
  1. First-order Hash-based Comparison
  2. Higher-order Reduce-Comparisons
     - Not applicable to Kyber, Saber due to ciphertext compression
- [BGR+21] Decompressed-Comparison : $>$ 50% of masked KYBER.CCA.DEC execution time

Side-channel Attacks on the FO Transform

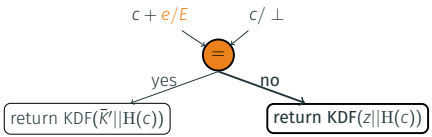
- The FO transform is a target for SCA that builds plaintext-checking oracles



- [BDH+21] Two masked comparisons broken, with fixes proposed:
  1. First-order Hash-based Comparison $\rightarrow$ **1. we show SCA collision attack**
  2. Higher-order Reduce-Comparisons
     - Not applicable to Kyber, Saber due to ciphertext compression
- [BGR+21] Decompressed-Comparison : $>$ 50% of masked KYBER.CCA.DEC execution time

Side-channel Attacks on the FO Transform

- The FO transform is a target for SCA that builds plaintext-checking oracles



- [BDH+21] Two masked comparisons broken, with fixes proposed:
  1. First-order Hash-based Comparison → **1. we show SCA collision attack**
  2. Higher-order Reduce-Comparisons → **2. we develop into a new method**
     - Not applicable to Kyber, Saber due to ciphertext compression
- [BGR+21] Decompressed-Comparison : > 50% of masked KYBER.CCA.DEC execution time

Hash-based Masked Comparison

- Masked Ciphertext Comparison

$$c \stackrel{?}{=} \mathbf{c}'^{(0)} + \mathbf{c}'^{(1)}$$

Hash-based Masked Comparison

- Masked Ciphertext Comparison

$$c \stackrel{?}{=} \mathbf{c}'^{(0)} + \mathbf{c}'^{(1)}$$

- Extra hashing step prevents sensitive leakage

$$H(c - \mathbf{c}'^{(0)}) \stackrel{?}{=} H(\mathbf{c}'^{(1)})$$

SCA on Hash-based Masked Comparison

- Trace contains 2 hash invocations:
  1. $H(c - \mathbf{c}'^{(0)})$
  2. $H(\mathbf{c}'^{(1)})$

SCA on Hash-based Masked Comparison

- Trace contains 2 hash invocations:
  1. $H(c - \mathbf{c'}^{(0)})$
  2. $H(\mathbf{c'}^{(1)})$
- We want to distinguish with 2^{nd}-order SCA between:
  1. $c - \mathbf{c'}^{(0)} = \mathbf{c'}^{(1)} + e$
  2. $c - \mathbf{c'}^{(0)} = \perp_1$, $\mathbf{c'}^{(2)} = \perp_2$

SCA on Hash-based Masked Comparison

- Trace contains 2 hash invocations:
  1. $H(c - \mathbf{c'}^{(0)})$
  2. $H(\mathbf{c'}^{(1)})$
- We want to distinguish with 2^{nd}-order SCA between:
  1. $c - \mathbf{c'}^{(0)} = \mathbf{c'}^{(1)} + e$
  2. $c - \mathbf{c'}^{(0)} = \perp_1$, $\mathbf{c'}^{(2)} = \perp_2$
- Keccak incrementally hashes these inputs in *5 blocks*

SCA on Hash-based Masked Comparison

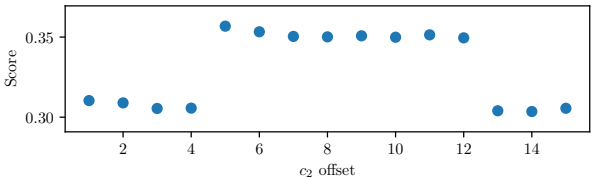
- Trace contains 2 hash invocations:
  1. $H(c - \mathbf{c'}^{(0)})$
  2. $H(\mathbf{c'}^{(1)})$
- We want to distinguish with 2^{nd}-order SCA between:
  1. $c - \mathbf{c'}^{(0)} = \mathbf{c'}^{(1)} + e$
  2. $c - \mathbf{c'}^{(0)} = \perp_1$, $\mathbf{c'}^{(2)} = \perp_2$
- Keccak incrementally hashes these inputs in *5 blocks*
- Horizontal collision attack
  1. Place the error e into the final block
  2. Compute "*collision score*" for first block

SCA on Hash-based Masked Comparison

- Trace contains 2 hash invocations:
  1. $H(c - \mathbf{c'}^{(0)})$
  2. $H(\mathbf{c'}^{(1)})$
- We want to distinguish with 2^{nd}-order SCA between:
  1. $c - \mathbf{c'}^{(0)} = \mathbf{c'}^{(1)} + e \rightarrow$ **first block identical**
  2. $c - \mathbf{c'}^{(0)} = \perp_1$, $\mathbf{c'}^{(2)} = \perp_2 \rightarrow$ **first block completely different**
- Keccak incrementally hashes these inputs in *5 blocks*
- Horizontal collision attack
  1. Place the error e into the final block
  2. Compute "*collision score*" for first block

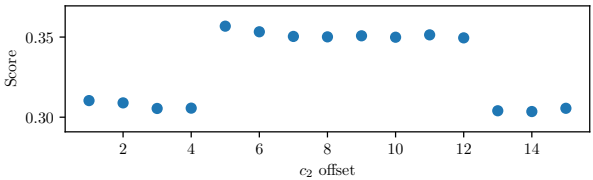
SCA on Hash-based Masked Comparison

- We want to distinguish with 2^{nd}-order SCA between:
  1. $c - \mathbf{c}'^{(0)} = \mathbf{c}'^{(1)} + e \rightarrow$ **first block identical**
  2. $c - \mathbf{c}'^{(0)} = \perp_1$, $\mathbf{c}'^{(2)} = \perp_2 \rightarrow$ **first block completely different**
- Horizontal collision attack
  1. Place the error e into the final block
  2. Compute "*collision score*" for first block



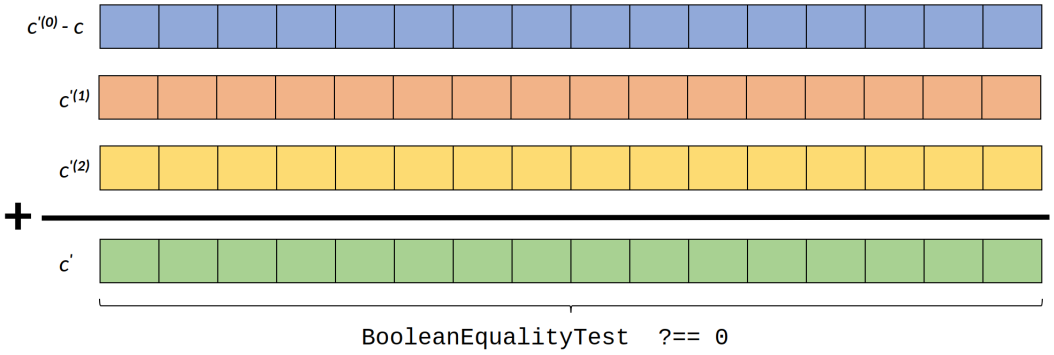
SCA on Hash-based Masked Comparison

- We want to distinguish with 2nd-order SCA between:
  1. $c - \mathbf{c}'^{(0)} = \mathbf{c}'^{(1)} + e \rightarrow$ **first block identical**
  2. $c - \mathbf{c}'^{(0)} = \perp_1$, $\mathbf{c}'^{(2)} = \perp_2 \rightarrow$ **first block completely different**
- Horizontal collision attack
  1. Place the error e into the final block
  2. Compute "*collision score*" for first block

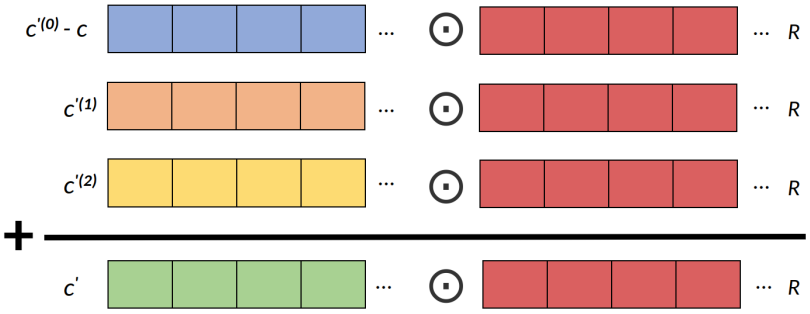


- Single trace for plaintext-checking, 6,000 traces to recover a Kyber512 key
  - **Higher-order Masked Comparison is very necessary**

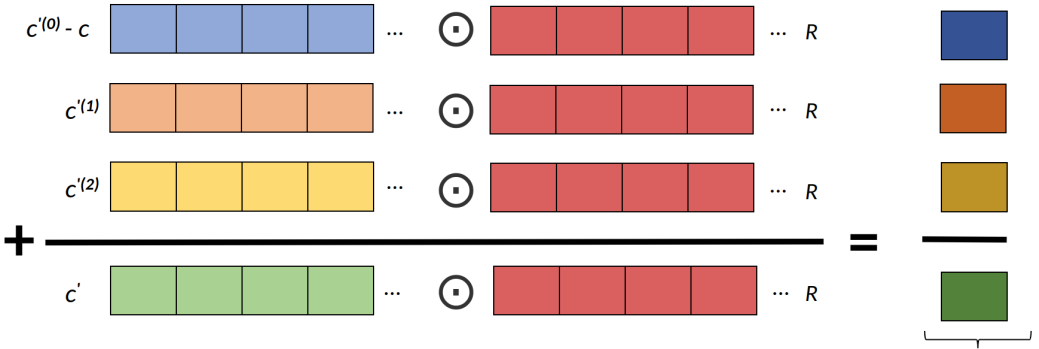
Reduce-Comparisons



Reduce-Comparisons



Reduce-Comparisons



`BooleanEqualityTest ?== 0`

Reduce-Comparisons

Collision probability P_{col} that $e_i \cdot R_i = 0 \bmod q$

1. Prime modulus q: $P_{col} = 1/q$
2. Power-of-two-modulus $q = 2^k$: $P_{col} = 1/2$

Repeat the check L times for probability P_{col}^L

Reduce-Comparisons

Collision probability P_{col} that $e_i \cdot R_i = 0 \bmod q$

1. Prime modulus q: $P_{col} = 1/q$
2. Power-of-two-modulus $q = 2^k$: $P_{col} = 1/2$

Repeat the check L times for probability P_{col}^L

Saber, but also Kyber is in Category 2 due to ciphertext compression

- $\texttt{Compress}_q(x) = \lfloor (2^k/q) \cdot x \rceil \bmod 2^k$

Random-sum Method

Collision probability $P_{col} = 1/2$ that $e_i \cdot R_i = 0 \mod 2^k$

- $e_i \leftarrow \mathbb{Z}_{2^k}$, $R_i \xleftarrow{\$} \mathbb{Z}_{2^k}$

Random-sum Method

Collision probability $P_{col} = 1/2$ that $e_i \cdot R_i = 0 \mod 2^k$

- $e_i \leftarrow \mathbb{Z}_{2^k}$, $R_i \xleftarrow{\$} \mathbb{Z}_{2^k}$

Simple key idea (from MPC literature) : enlarge the masking modulus

Collision probability $P_{col} = 1/2^s$ that $e_i \cdot R_i = 0 \mod 2^{k+s}$

- $e_i \leftarrow \mathbb{Z}_{2^k}$, $R_i \xleftarrow{\$} \mathbb{Z}_{2^s}$

Random-sum Method

Collision probability $P_{col} = 1/2$ that $e_i \cdot R_i = 0 \mod 2^k$

- $e_i \leftarrow \mathbb{Z}_{2^k}$, $R_i \xleftarrow{\$} \mathbb{Z}_{2^k}$

Simple key idea (from MPC literature) : enlarge the masking modulus

Collision probability $P_{col} = 1/2^s$ that $e_i \cdot R_i = 0 \mod 2^{k+s}$

- $e_i \leftarrow \mathbb{Z}_{2^k}$, $R_i \xleftarrow{\$} \mathbb{Z}_{2^s}$

More involved implementation :

1. Higher-order MaskedCompress [FVBBR$^+$21]
2. A combination of B2A, A2B to enlarge the masking modulus

Random-sum Method

Input : $\mathbf{c}'$, c. **Output** : $Compress(\mathrm{c}') \stackrel{?}{=} c$

Step 0 : Pre-processing

Random-sum Method

Input : $\mathbf{c}'$, c. **Output** : $Compress(\mathrm{c}') \stackrel{?}{=} c$

Step 0 : Pre-processing

- $c'^{(0)} \leftarrow c'^{(0)} - c$

Random-sum Method

Input : $\mathbf{c}'$, c. **Output** : $Compress(\mathbf{c}') \stackrel{?}{=} c$

Step 0 : Pre-processing

- $c'^{(0)} \leftarrow c'^{(0)} - c$
- $c_i'^{(j)} \leftarrow \lfloor (2^{k+f_{bits}}/q) \cdot c_i'^{(j)} \rceil \bmod 2^{k+f_{bits}}$
  - Higher-order correctness proof for f_{bits} in the paper

Random-sum Method

Input : $\mathbf{c}'$, c. **Output** : $Compress(\mathbf{c}') \stackrel{?}{=} c$

Step 0 : Pre-processing

- $c'^{(0)} \leftarrow c'^{(0)} - c$
- $c_i'^{(j)} \leftarrow \lfloor (2^{k+f_{bits}}/q) \cdot c_i'^{(j)} \rceil \bmod 2^{k+f_{bits}}$
  - Higher-order correctness proof for f_{bits} in the paper

Step 1 : Convert to Boolean masking

- $\mathbf{c}' \leftarrow A2B(\mathbf{c}') \gg f_{bits}$

Random-sum Method

Input : $\mathbf{c}'$, c. **Output** : $Compress(\mathbf{c}') \stackrel{?}{=} c$

Step 0 : Pre-processing

- $c'^{(0)} \leftarrow c'^{(0)} - c$
- $c_i'^{(j)} \leftarrow \lfloor (2^{k+f_{bits}}/q) \cdot c_i'^{(j)} \rceil \bmod 2^{k+f_{bits}}$
  - Higher-order correctness proof for f_{bits} in the paper

Step 1 : Convert to Boolean masking

- $\mathbf{c}' \leftarrow A2B(\mathbf{c}') \gg f_{bits}$

Step 2 : Convert to arithmetic masking modulo 2^{k+s}

- $\mathbf{c}' \leftarrow B2A_{2^k \rightarrow 2^{k+s}}(\mathbf{c}')$

Random-sum Method

Input : $\mathbf{c}'$, c. **Output** : $Compress(\mathbf{c}') \stackrel{?}{=} c$

Step 0 : Pre-processing

- $c'^{(0)} \leftarrow c'^{(0)} - c$
- $c_i'^{(j)} \leftarrow \lfloor (2^{k+f_{bits}}/q) \cdot c_i'^{(j)} \rceil \bmod 2^{k+f_{bits}}$
  - Higher-order correctness proof for f_{bits} in the paper

Step 1 : Convert to Boolean masking

- $\mathbf{c}' \leftarrow A2B(\mathbf{c}') \gg f_{bits}$

Step 2 : Convert to arithmetic masking modulo 2^{k+s}

- $\mathbf{c}' \leftarrow B2A_{2^k \rightarrow 2^{k+s}}(\mathbf{c}')$

Step 3 : ReduceComparisons

Random-sum Method

Input : $\mathbf{c}'$, c. **Output** : $Compress(\mathbf{c}') \stackrel{?}{=} c$

Step 0 : Pre-processing

- $c'^{(0)} \leftarrow c'^{(0)} - c$
- $c_i'^{(j)} \leftarrow \lfloor (2^{k+f_{bits}}/q) \cdot c_i'^{(j)} \rceil \bmod 2^{k+f_{bits}}$
  - Higher-order correctness proof for f_{bits} in the paper

Step 1 : Convert to Boolean masking

- $\mathbf{c}' \leftarrow A2B(\mathbf{c}') \gg f_{bits}$

Step 2 : Convert to arithmetic masking modulo 2^{k+s}

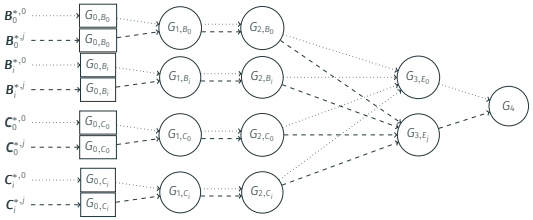
- $\mathbf{c}' \leftarrow B2A_{2^k \rightarrow 2^{k+s}}(\mathbf{c}')$

Step 3 : ReduceComparisons

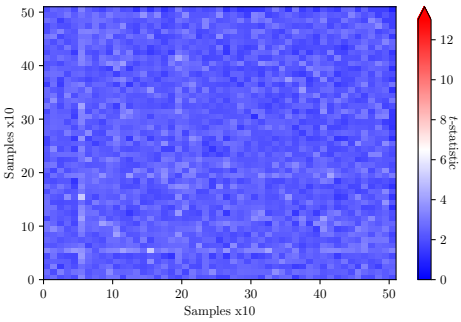
Step 4 : BooleanEqualityTest

Security Proof and Leakage Evaluation

t-SNI security proof



Bivariate 2^{nd}-order *t*-test evaluation



Performance Evaulation

Performance breakdown on ARM-Cortex M4 with collision probability $2^{-s} = 2^{-54}$

Scheme	Comparison	CPU [k]cycles		#random bytes	
		2^{nd}	3^{rd}	2^{nd}	3^{rd}
Saber	**MaskedComp**	**3,363**	**6,543**	**90,264**	**205,104**
Kyber768	**MaskedComp**	**5,253**	**9,712**	**111,256**	**251,184**
	|-Step 0	|-355	|-467	|-0	|-0
	|-Step 1	|-3,485	|-5,742	|-44,544	|-93,184
	|-Step 2	|-1,157	|-3,179	|-57,344	|-147,456
	|-Step 3	|-182	|-197	|-8,192	|-8,192
	|-Step 4	|-73	|-128	|-1,176	|-2,352
	[BGR+21]	22,017 (**4.2x**)	72,568 (**7.5x**)	902,126†	2,434,170†

Original Reduce-Comparisons : $e_i \cdot R_i = 0 \mod 2^k$

Our Reduce-Comparisons : $e_i \cdot R_i = 0 \mod 2^{k+s}$

- Need A2B, B2A to enlarge the masking modulus

Revisiting Higher-Order Masked Ciphertext Comparison for Lattice-Based Cryptography

Original Reduce-Comparisons : $e_i \cdot R_i = 0 \mod 2^k$

Our Reduce-Comparisons : $e_i \cdot R_i = 0 \mod 2^{k+s}$

- Need A2B, B2A to enlarge the masking modulus

Galois-field Reduce-Comparisons : $e_i \odot R_i = 0$

- After A2B, natively in "GF" format

Revisiting Higher-Order Masked Ciphertext Comparison for Lattice-Based Cryptography

Scheme	Comparison	CPU cycles 2^{nd}	CPU cycles 3^{rd}	#random bytes 2^{nd}	#random bytes 3^{rd}
Kyber768	This work (2.1x)	5,2M	9,7M	111K	251K
	GF (1.7x)	4,2M	6,7M	47K	95K
	Simple (1.2x)	3,1M	5,2M	48K	100K
	[CGMZ21] (1.3x)	3.3M	4.5M	80K	94K
	Streamlined Hybrid (1.0x)	2,5M	3,6M	44K	62K

`github.com/KULeuven-COSIC/Masked-Comparison`

`github.com/KULeuven-COSIC/Revisiting-Masked-Comparison`

References

- Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck.
  Attacking and defending masked polynomial comparison.
  IACR TCHES, 2021(3):334–359, 2021.
  https://tches.iacr.org/index.php/TCHES/article/view/8977.
- Joppe W. Bos, Marc Gourjon, Joost Renes, Tobias Schneider, and Christine van Vredendaal.
  Masking kyber: First- and higher-order implementations.
  IACR TCHES, 2021(4):173–214, 2021.
  https://tches.iacr.org/index.php/TCHES/article/view/9064.
- Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun.
  High-order polynomial comparison and masking lattice-based encryption.
  Cryptology ePrint Archive, Paper 2021/1615, 2021.
  https://eprint.iacr.org/2021/1615.
- 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.
  IACR TCHES, 2022(1):414–460, 2021.
  https://tches.iacr.org/index.php/TCHES/article/view/9303.