NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

On Configurable SCA Countermeasures Against Single Trace Attacks for the NTT -

A Performance Evaluation Study over Kyber and Dilithium on the ARM Cortex-M4

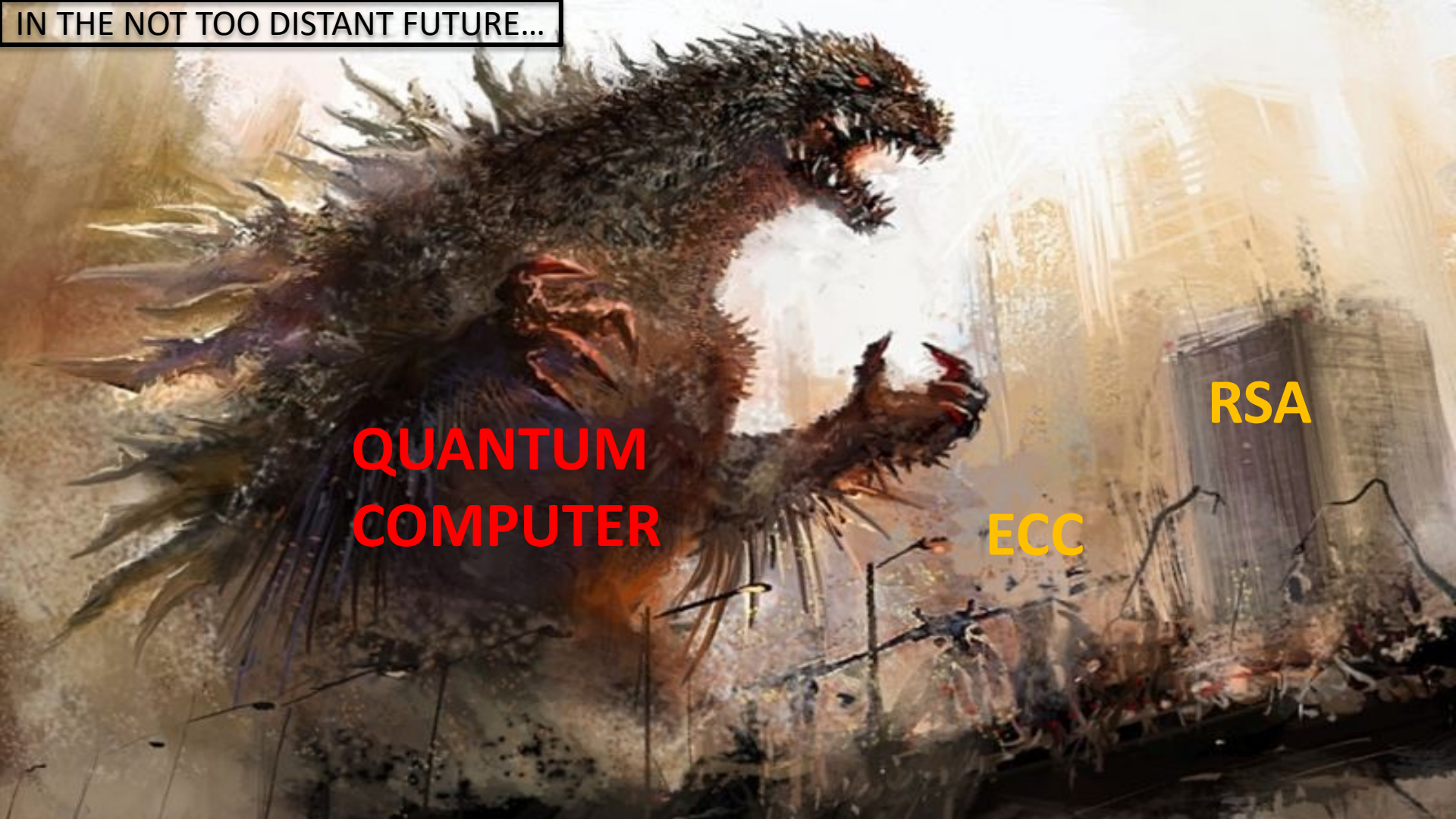
Prasanna Ravi[12], Romain Poussier[1], Shivam Bhasin[1] and Anupam Chattopadhyay[12]

Temasek Labs@NTU, Singapore

School of Physical and Mathematical Sciences, NTU Singapore

SPACE 2020,

December 19th, 2020


IN THE NOT TOO DISTANT FUTURE…
QUANTUM COMPUTER
ECC
RSA

Security in Quantum Era: NIST PQC Call

- **National Institute of Standards and Technology (NIST)** initiated a standardization process for post-quantum cryptography (**PQC**) in November 2017.
- The first round had **69** candidates, second round had **26** candidates and the process is currently in its **third** and **final** round.

Type	Signature	KEM/Encryption	**Finalist (Alternate)**
Lattice Based	**2**	**3 (2)**	**5 (2)**
Code-Based	-	1 (2)	1 (2)
Multivariate	1 (1)	-	1 (1)
Hash-Based	- (2)	-	- (2)
Isogeny based	-	- (1)	- (1)
Others	-	-	- (0)
Total	3 (3)	4 (5)	7 (8)

Security in Quantum Era: NIST PQC Call

- **Selection Criteria for Standardization Process:**
  - Theoretical Post-Quantum Security Guarantees
  - Implementation Performance (Speed, Area, latency, Power) on various HW/SW platforms
  - **Resistance against Side-Channel Attacks (SCA) and Fault Injection Attacks (FIA)**

- Several works done on side-channel attacks [1,2,3] as well as protection techniques [4,5].

- Most works on protected implementations focus on DPA style attacks, but very little is studied about protection against single trace attacks [6].

- Primas et al. ([2] in CHES-2017) proposed a single trace attack on the Number Theoretic Transform (NTT) over Ring-LWE encryption scheme (improved by Pessl et al. [3] in Latincrypt 2019).

- In this work, we focus on developing **countermeasures** for **NTT** against single trace attacks.

This Work

- We propose novel **masking** and **shuffling** countermeasures for the NTT.
- **Twiddle-Factor Randomization**: Exploitation of special arithmetic properties of twiddle factors
- We adopt a **bottom-up** approach to construct generic masked NTT with configurable SCA resistance.
- We also propose novel variants of the shuffling countermeasure for NTT with varying granularity.
- Experimental evaluation done on **Kyber** and **Dilithium** (Finalists) from the open source *pqm4* library on the ARM Cortex-M4 microcontroller.
- **Performance Overhead** (Clock Cycles):
  - **Kyber**: **7-78 %** across all procedures (KeyGen, Encaps, Decaps)
  - **Dilithium**: **12-197 %** for KeyGen and **32-490 %** for Sign

Preliminaries

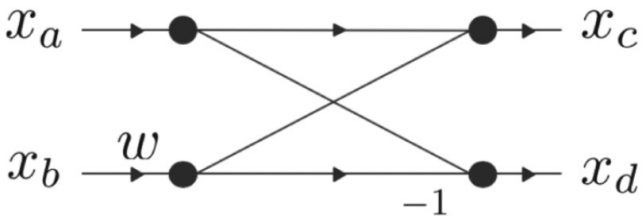
Number Theoretic Transform (NTT)

- An important kernel used for efficient polynomial multiplication in several lattice-based schemes.
- Used in schemes such as Kyber, Dilithium and NewHope which are designed with NTT-friendly parameters.
- Multiplication of two polynomials (i.e) $\mathbf{z} = \mathbf{x} \times \mathbf{y} \in R_q$ in the polynomial ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ is done as follows:

$$\mathbf{z} = \mathrm{INTT}(\mathrm{NTT}(\mathbf{x}) \circ \mathrm{NTT}(\mathbf{y}))$$

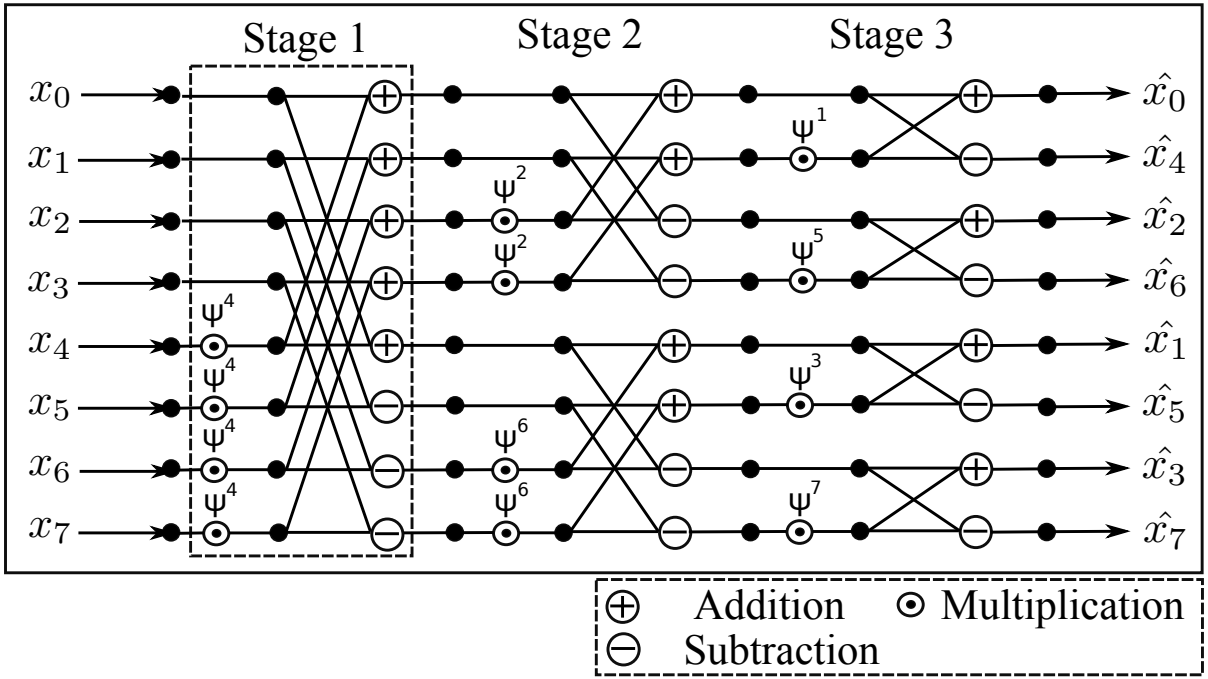
- Computation of NTT is similar to the Fast Fourier Transform (FFT) and uses the well known ***butterfly*** network.

Simple two-input NTT (n = 2)



$$x_c = x_a + x_b \cdot w$$
$$x_d = x_a - x_b \cdot w$$

Number Theoretic Transform (NTT)



Eight input NTT (n = 8)

- **For an n-input NTT:**
  - log(n) stages
  - n/2 butterflies in each stage
  - n/2 log(n) additions
  - n/2 log(n) subtractions
  - n/2 log(n) multiplications
  - Powers of n^{th} or $2n^{th}$ roots of unity (**twiddle-factors**).

Attacking NTT/INTT for key recovery

- Given ciphertext $\mathbf{ct} = (\mathbf{u}, \mathbf{v}) \in R_q$, decryption procedure computes

$$\begin{aligned}\mathbf{m}' &= (\mathbf{v} - \mathbf{u} \times \mathbf{s}) \\ &= \mathsf{NTT}(\mathbf{v}) - \mathsf{NTT}(\mathbf{u}) \circ \mathsf{NTT}(\mathbf{s}) \\ &= \mathsf{INTT}(\mathbf{v}' - \mathbf{u}' \circ \mathbf{s}') \\ &= \boxed{\mathsf{INTT}(\mathbf{k})}\end{aligned}$$

- Use side-channel information to recover input to the INTT. Subsequently,

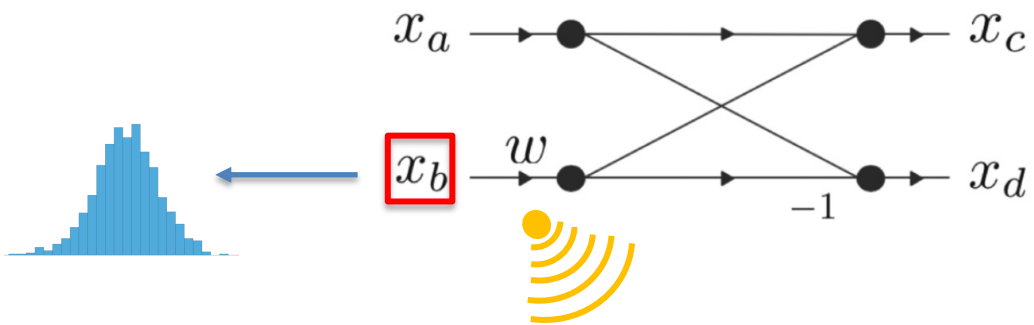
$$\mathbf{s} = \mathsf{INTT}(\mathbf{k} - \mathbf{v}') \times \mathbf{u}^{-1}$$

Attack Flow

- **Step 1:** Single trace Template Attack (TA) on various intermediate values of INTT operation
- **Step 2:** Combine leakage using Belief Propagation (BP) to recover input of INTT.
- **Step 3:** Secret Key Recovery using lattice-decoding algorithm.

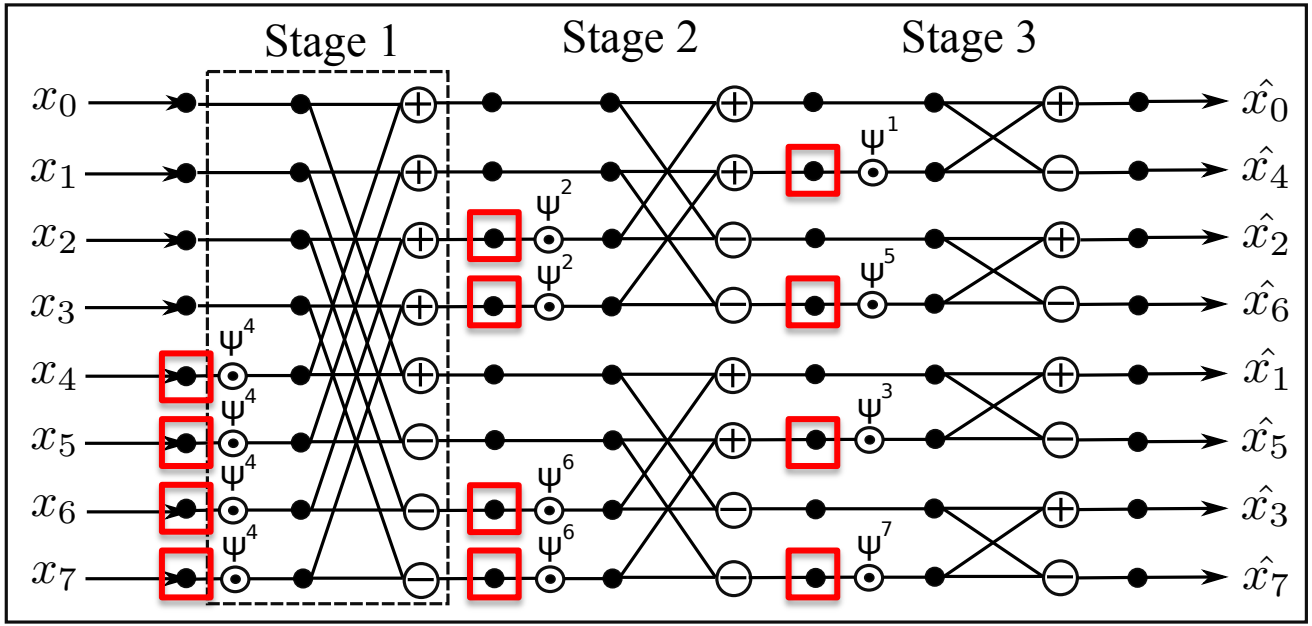
Step 1: Single Trace Template Attack

- Illustration using simple 2-input butterfly:
- **Target Operation**: Modular Multiplication (Primas et al. [2])



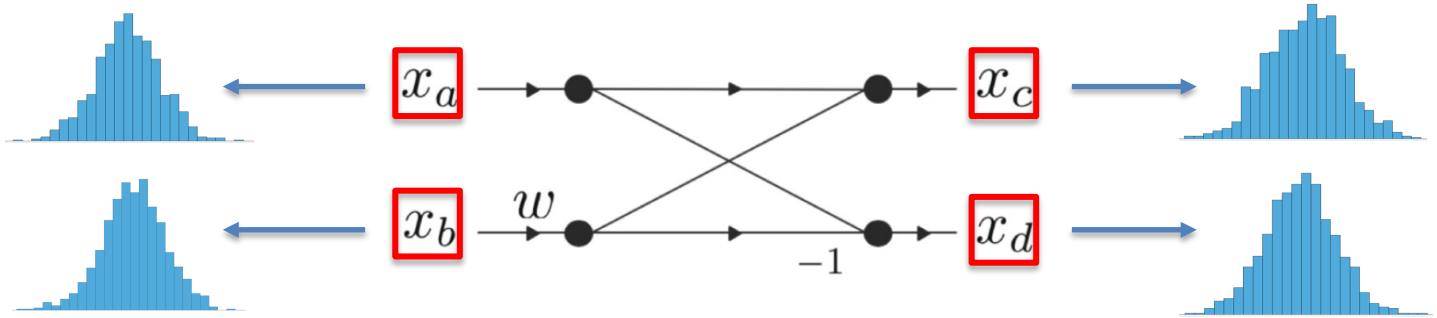
Step 1: Single Trace Template Attack

- Illustration using simple 2-input butterfly:
- **Target Operation**: Modular Multiplication (Primas et al. [2])



Step 1: Single Trace Template Attack

- Illustration using simple 2-input butterfly:
- **Target Operation**: Modular Multiplication (Primas et al. [2])
- Loads and Stores of input and outputs were also templated by Pessl et al. [3].

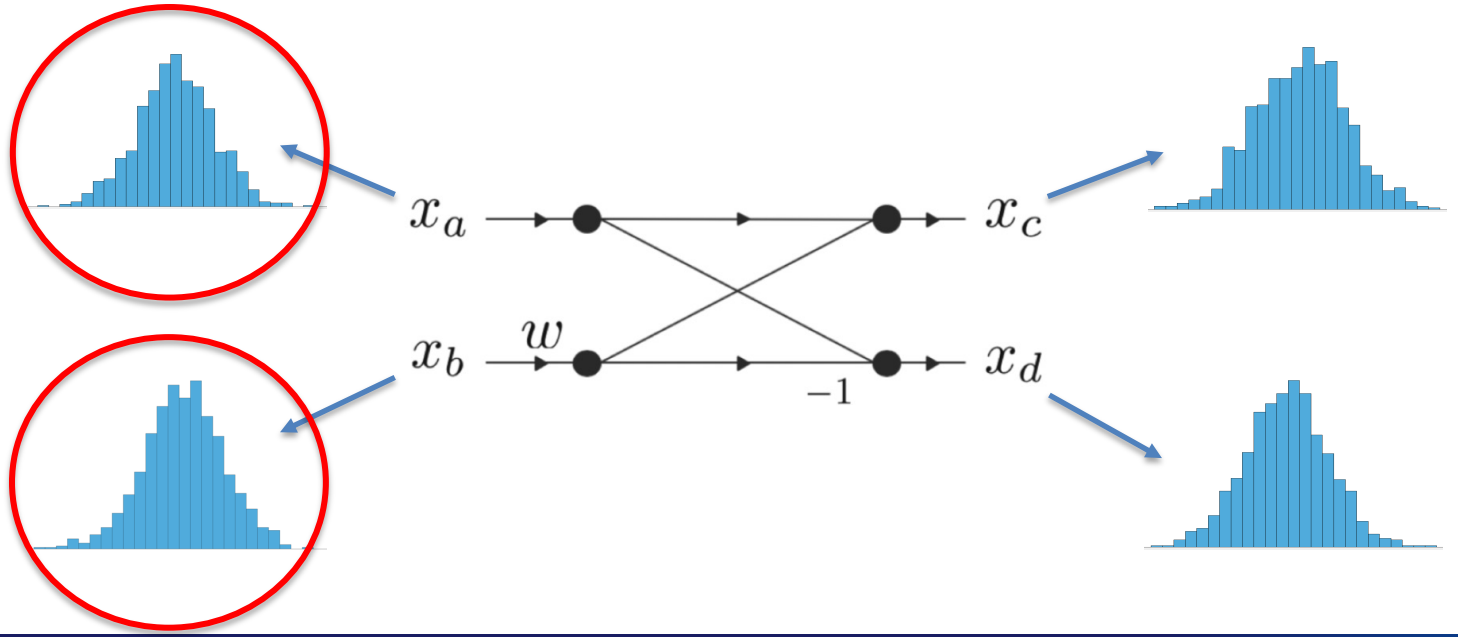


Step 2: Combining Leakage using BP

- It is an iterative algorithm typically used in algebraic style side-channel attacks [8,9].
- Combines leakage information (probability distributions) from different intermediates to arrive at a unique (close to unique) solution.

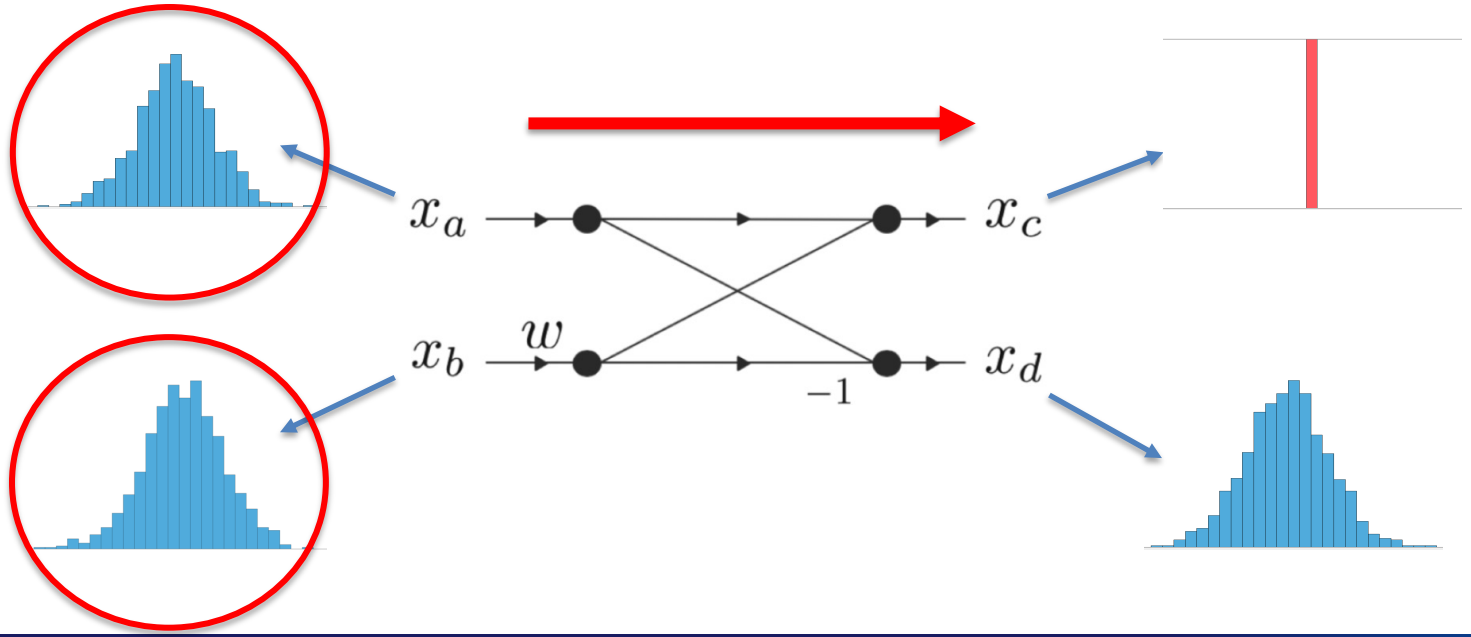
Step 2: Combining Leakage using BP

- It is an iterative algorithm typically used in algebraic style side-channel attacks [8,9].
- Combines leakage information (probability distributions) from different intermediates to arrive at a unique (close to unique) solution.



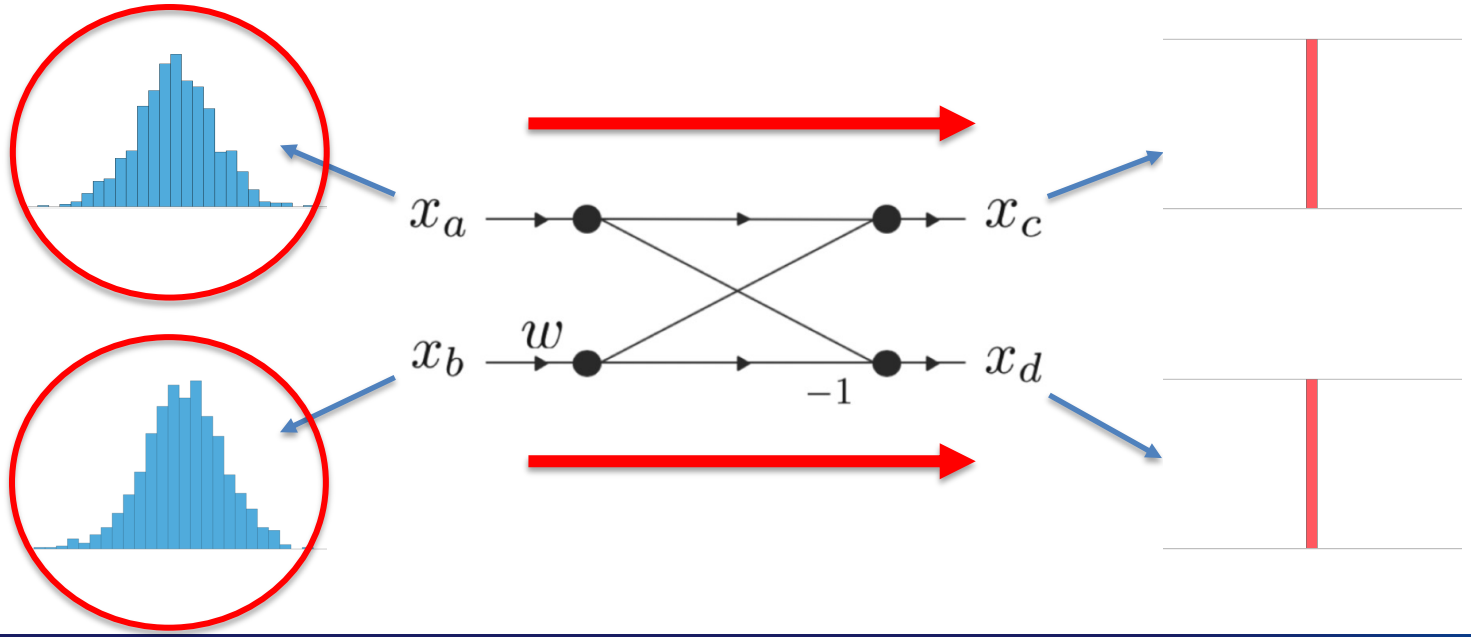
Step 2: Combining Leakage using BP

- It is an iterative algorithm typically used in algebraic style side-channel attacks [8,9].
- Combines leakage information (probability distributions) from different intermediates to arrive at a unique (close to unique) solution.



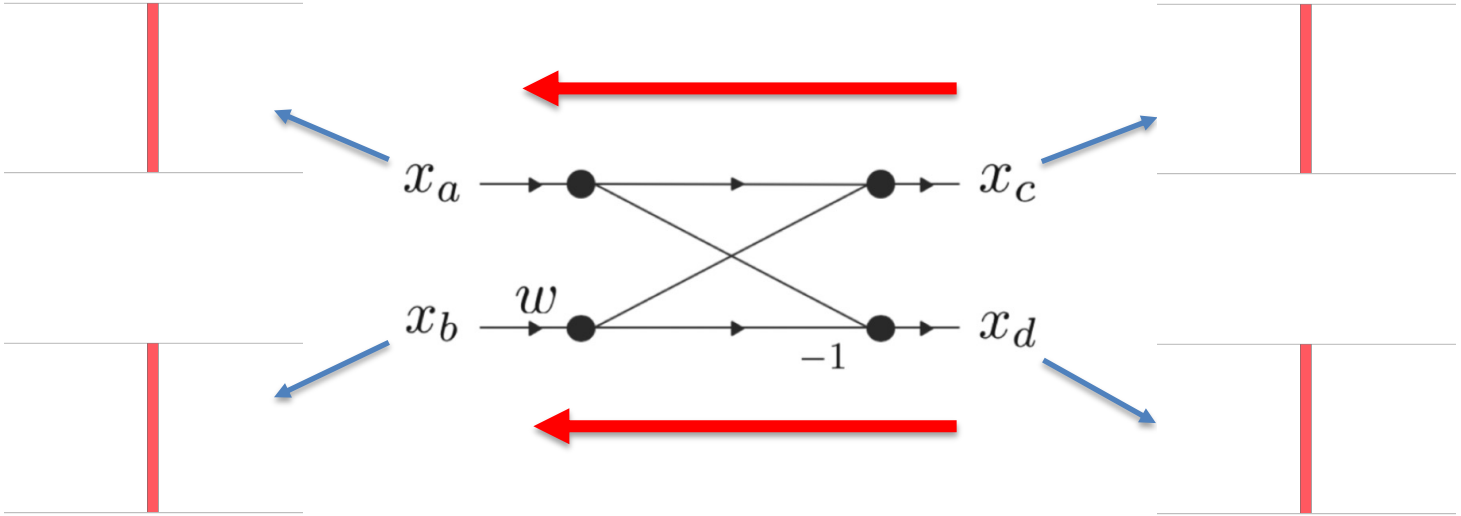
Step 2: Combining Leakage using BP

- It is an iterative algorithm typically used in algebraic style side-channel attacks [8,9].
- Combines leakage information (probability distributions) from different intermediates to arrive at a unique (close to unique) solution.



Step 2: Combining Leakage using BP

- It is an iterative algorithm typically used in algebraic style side-channel attacks [8,9].
- Combines leakage information (probability distributions) from different intermediates to arrive at a unique (close to unique) solution.



Masked NTT

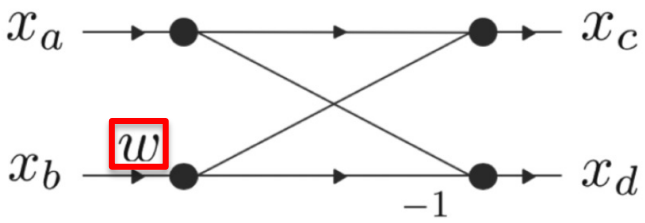
Masking Countermeasures for the NTT

- **Motivation**: Analysis of SASCA over block ciphers such as AES [8,9].

$$Y = \mathrm{SBOX}(X \oplus K)$$

- There is a significant degradation in the attack success rate when the plaintext is unknown.

- **Key Observation**: Twiddle factors are the only known constants within the NTT operation.



- **Key Idea**: **Randomize** the twiddle factors such that there are no known constants used in the NTT.

- The target NTT is fully computed using randomized values.

Twiddle Factor Masking

- Illustration using simple 2-input BF:

$$c = a + b \cdot w_x$$
$$d = a - b \cdot w_x$$

- We now compute:

$$\begin{aligned} c' &= c \cdot w_y \\ &= (a + b \cdot w_x) \cdot w_y \\ &= a \cdot w_y + b \cdot w_x \cdot w_y \\ &= a \cdot w_y + b \cdot w_{(x+y)} \quad (\because w_{(x+y)} = \psi^x \cdot \psi^y = \psi^{(x+y)}) \\ d' &= a \cdot w_y - b \cdot w_{(x+y)} \end{aligned}$$

Product of two twiddle factors is another twiddle factor!!

Twiddle Factor Masking

- We consider masked inputs $(a', b') = (a \cdot w_i, b \cdot w_i)$

$$
\begin{aligned}
c' &= (a' + b' \cdot w_x) \cdot w_y \\
&= a \cdot w_i \cdot w_y + b \cdot w_i \cdot w_x \cdot w_y \\
&= a \cdot w_{(i+y)} + b \cdot w_{(i+x+y)} \\
&= c \cdot w_{(i+y)}
\end{aligned}
$$

- Inputs masked with w_i and outputs with $w_{(i+y)}$: **Same Input Same Output** (**SISO**) masked BF

- **Advantages**:
  - All twiddle factors (masks) are pre-computed.
  - Mask access using simple table look up (only generate index).
  - Only one extra multiplication compared to unmasked BF.

Twiddle Factor Masking

- We extend to inputs masked with different twiddle factors

$$(a', b') = (a \cdot w_i, b \cdot w_j)$$

- Computation of butterfly only possible by **re-masking** both inputs with same mask.

- We exploit the property that $\psi^{2n} = 1$ to **re-mask** without **unmasking**:

$$\begin{aligned} a'' &= a' \cdot w_{(2n-i+k)} \\ &= a \cdot w_i \cdot w_{(2n-i+k)} \\ &= a \cdot w_k \end{aligned}$$

Twiddle Factor Masking

- We extend the same to inputs masked with different twiddle factors

$$(a', b') = (a \cdot w_i, b \cdot w_j)$$

$$\begin{aligned} c' &= (a' \cdot w_{(2n-i)} + b' \cdot w_{(2n-j)} \cdot w_x) \cdot w_y \\ &= a' \cdot w_{(2n-i+y)} + b' \cdot w_{(2n-j+y+x)} \\ &= a \cdot w_i \cdot w_{(2n-i+y)} + b \cdot w_j \cdot w_{(2n-j+y+x)} \\ &= c \cdot w_y \\ d' &= a' \cdot w_{(2n-i+y)} - b' \cdot w_{(2n-j+y+x)} \end{aligned}$$

- We refer to it as **Different Input Same Output** (**DISO**) masked BF.

- Same computation cost as that of **SISO** masked BF.

Twiddle Factor Masking

- We can extend the same techniques to construct
  - Same Input Different Output (**SIDO**) masked BF
  - Different Input Different Output (**DIDO**) masked BF

- We observe that *DO (SIDO and DIDO) BFs are costlier than *SO (SISO and DISO) BFs.
  - Unmasked butterfly: **1** mult.
  - *SO butterfly : **2** mults.
  - *DO butterfly : **4** mults.

- We now use these four masked BFs for our generic masked NTT construction.

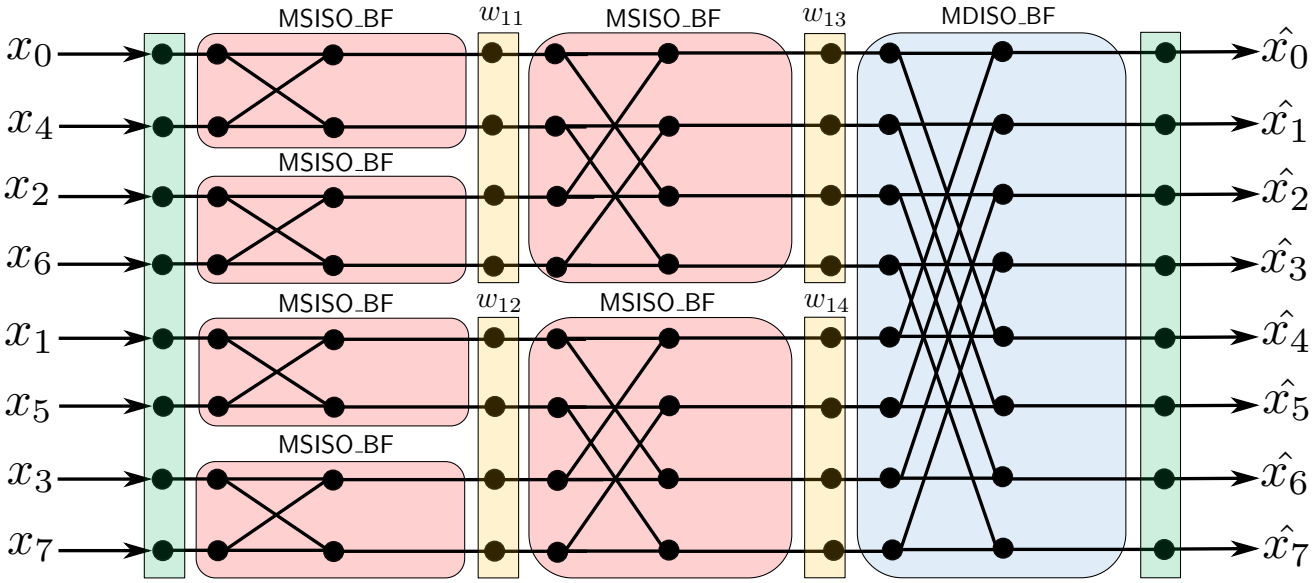
Building Generic Masked NTTs

- Requires to decide number of masks to be used for each stage (1…n)

- **Coarse-Masked NTT** : 1 mask
- **Fine-Masked NTT** : n masks
- **Generic-x-Masked NTT** : x masks (x = 2^k)
- **Generic-Random-Masked NTT**

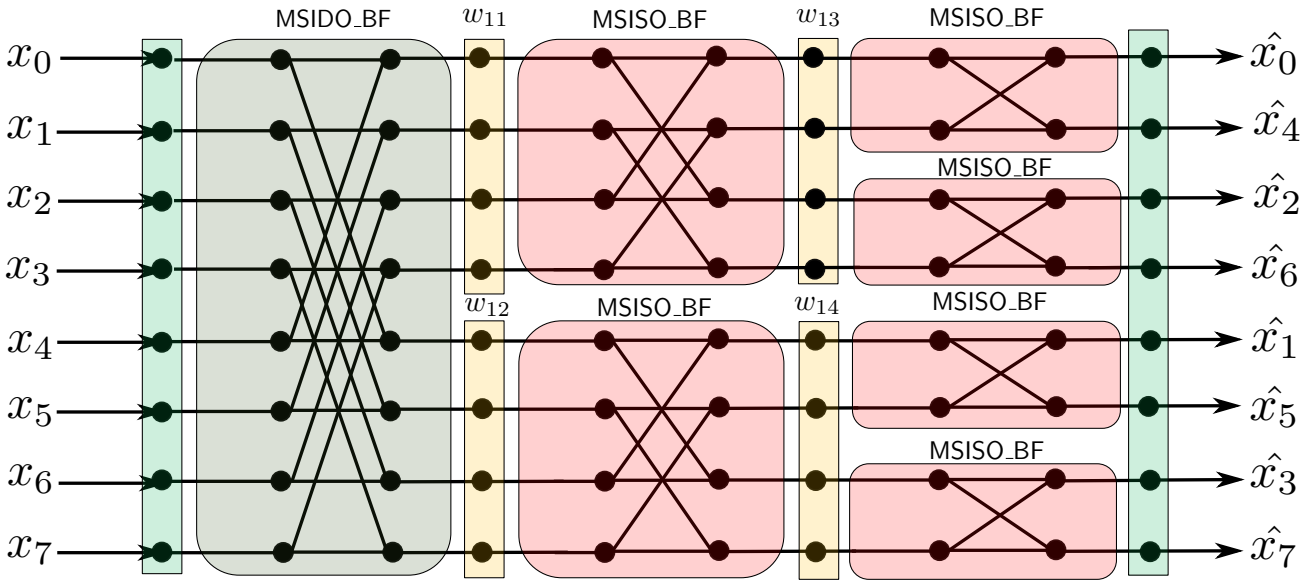
Building Generic Masked NTTs

- We illustrate using an example of NTT with **two (2)** masks per stage



Building Generic Masked NTTs

- We illustrate using an example of NTT with **two (2)** masks per stage

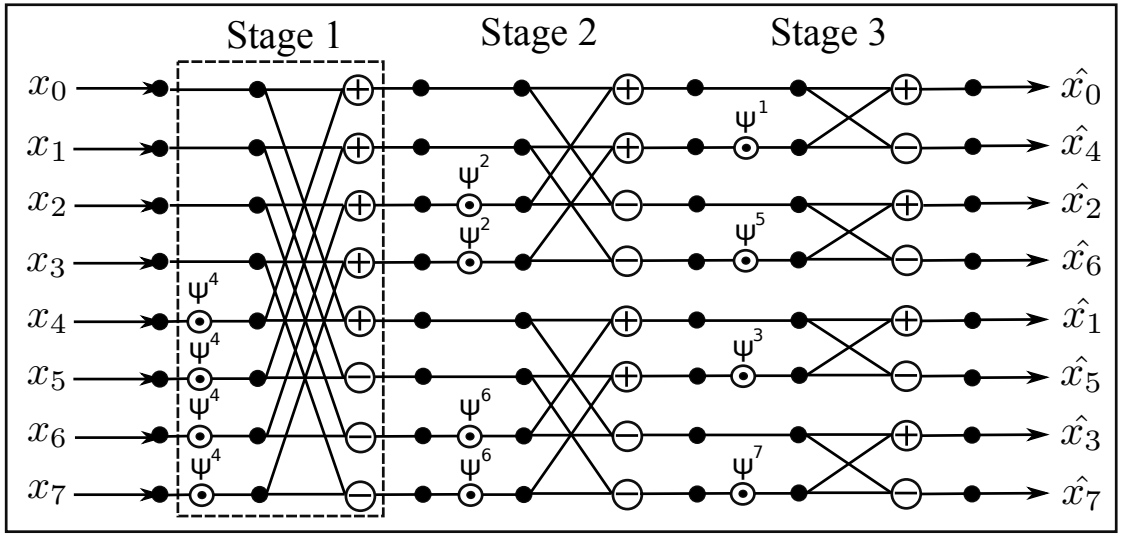


Shuffled NTT

Shuffling Countermeasures for the NTT

- Shuffling is typically a strong countermeasure against SASCA style attacks.
- Assigning leakage to corresponding nodes in factor graph is not possible (provided shuffling is not broken).
- We propose shuffling countermeasures for the NTT with different granularity.

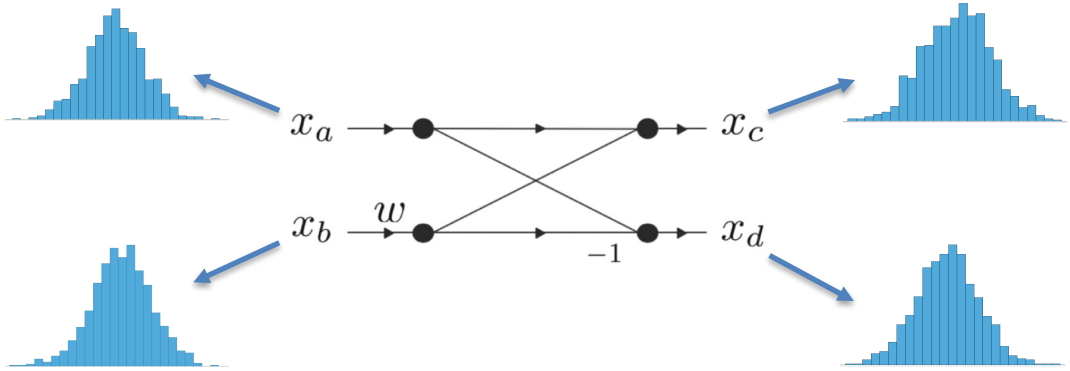
Shuffling Countermeasures for the NTT



- **Key Observation:** No dependency between butterfly operations of a given stage.
- **Coarse-Full-Shuffled** NTT: Shuffle all butterflies within a given stage (proposed in [2,3])
- **Coarse-In-Group-Shuffled** NTT: Shuffle all butterflies within a group.

Shuffling Countermeasures for the NTT

- Pessl et al. [3] template the loads and stores of operands of each BF (apart from multiplication).



- **Fine-Shuffled** NTT: Shuffle the order of load and stores within each BF.
- Weaker mitigation strategy since multiplication can still be attacked.
- However, number of leakage PoI reduces as well the required number of templates increases significantly (from 213 templates to 1 million templates) [2]

Shuffling Countermeasures for the NTT

- We adapted the arithmetic conditional swap technique for randomizing loads/stores.

Randomized loading of p[j] and p[j + len]

$$mask = -1 \cdot r \quad (r = 0/1 \text{ and } mask = 0\text{x}0000/0\text{xFFFF})$$
$$op1 = p[j + r \cdot len]$$
$$op2 = p[j + (1 - r) \cdot len]$$

→ **Randomized Loading**

Attacks over ECC implementations [6,7]

$$temp = (op1 \oplus op2)$$
$$temp = temp \,\&\, mask$$
$$op1 = op1 \oplus temp$$
$$op2 = op2 \oplus temp$$

→ **Arithmetic Conditional Swap**

- Replace **16-bitwide** operations with iterative **single bit operations** to significantly reduce leakage.
- Removes usage of the leaky mask variable.

Implementation Details

- Performance Evaluation over implementations of **Kyber** and **Dilithium** on the ARM Cortex-M4.
- Target: STM32F4DISCOVERY board with the STM32F407 MCU (24 MHz).
- The countermeasures are implemented over C-based implementations of both NTT and INTT.
- Hardware TRNG used as sole randomness source, however NIST approved XOFs based on the SHA3 or AES in counter mode seeded using TRNG can also be used.
- **Kyber**:
  - modulus q = 3329, n = 256 (operates over 16-bits)
  - n^{th} root of unity in the ring (no. of masks = 256)
- **Dilithium**:
  - modulus q = 8380417, n = 256 (operates over 32-bits)
  - n^{th} and $2n^{th}$ root of unity in the ring (no. of masks = 512)

Experimental Results: NTT Perf. Evaluation

❑ **Masking Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)		
	Count	Overhead (%)	Rand.
Kyber			
Unprotected	31.0	-	
Coarse-Masked	44.6	43.7	0.2 (**0.4%**)
Generic-2-Masked	66.5	114.5	0.5 (**0.7%**)
Generic-4-Masked	72.1	132.7	1.0 (**1.4%**)
Fine-Masked	171.1	451.7	65.7 (**38.4%**)

Experimental Evaluation

Experimental Results: NTT Perf. Evaluation

❑ **Masking Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)		
	Count	Overhead (%)	Rand.
Kyber			
Unprotected	31.0	-	
Coarse-Masked	44.6	43.7	0.2 (**0.4%**)
Generic-2-Masked	66.5	114.5	0.5 (**0.7%**)
Generic-4-Masked	72.1	132.7	1.0 (**1.4%**)
Fine-Masked	171.1	451.7	65.7 (**38.4%**)

Experimental Results: NTT Perf. Evaluation

❑ **Masking Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)		
	Count	Overhead (%)	Rand.
Kyber			
Unprotected	31.0	-	
Coarse-Masked	44.6	43.7	0.2 (**0.4%**)
Generic-2-Masked	66.5	114.5	0.5 (**0.7%**)
Generic-4-Masked	72.1	132.7	1.0 (**1.4%**)
Fine-Masked	171.1	451.7	65.7 (**38.4%**)

Experimental Results: NTT Perf. Evaluation

❑ **Masking Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)		
	Count	Overhead (%)	Rand.
Kyber			
Unprotected	31.0	-	
Coarse-Masked	44.6	43.7	0.2 (**0.4%**)
Generic-2-Masked	66.5	114.5	0.5 (**0.7%**)
Generic-4-Masked	72.1	132.7	1.0 (**1.4%**)
Fine-Masked	171.1	451.7	65.7 (**38.4%**)

Experimental Results: NTT Perf. Evaluation

❑ **Masking Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)		
	Count	Overhead (%)	Rand.
Dilithium			
Unprotected	55.2	-	
Coarse-Masked	91.3	65.4	0.4 (**0.4%**)
Generic-2-Masked	125.5	127.3	0.5 (**0.4%**)
Generic-4-Masked	139	151.7	0.9 (**0.6%**)
Fine-Masked	297.3	438.2	70.2 (**23.6%**)

Experimental Results: NTT Perf. Evaluation

❑ **Masking Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)		
	Count	Overhead (%)	Rand.
Dilithium			
Unprotected	55.2	-	
Coarse-Masked	91.3	65.4	0.4 (**0.4%**)
Generic-2-Masked	125.5	127.3	0.5 (**0.4%**)
Generic-4-Masked	139	151.7	0.9 (**0.6%**)
Fine-Masked	297.3	438.2	70.2 (**23.6%**)

Experimental Results: NTT Perf. Evaluation

❑ **Masking Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)		
	Count	Overhead (%)	Rand.
Dilithium			
Unprotected	55.2	-	
Coarse-Masked	91.3	65.4	0.4 (**0.4%**)
Generic-2-Masked	125.5	127.3	0.5 (**0.4%**)
Generic-4-Masked	139	151.7	0.9 (**0.6%**)
Fine-Masked	297.3	438.2	70.2 (**23.6%**)

Experimental Results: NTT Perf. Evaluation

❑ **Shuffling Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)			
	Count	Overhead (%)	Shuffle.	Rand.
Kyber				
Unprotected	31.0	-	-	-
Coarse-Full-Shuffled	87.2	181.1	16.6 (**19%**)	38.4 (**44.1%**)
Coarse-In-Group-Shuffled	84.4	172.2	17.1 (**20.3%**)	32.4 (**38.4%**)
Basic-Fine-Shuffled	76.7	147.4	35.1 (**45.7%**)	9.5 (**12.4%**)
Bitwise-Fine-Shuffled	142.6	356	100.1 (**70.2%**)	9.5 (**6.7%**)

Experimental Results: NTT Perf. Evaluation

❑ **Shuffling Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)			
	Count	Overhead (%)	Shuffle.	Rand.
Kyber				
Unprotected	31.0	-	-	-
Coarse-Full-Shuffled	87.2	181.1	16.6 (**19%**)	38.4 (**44.1%**)
Coarse-In-Group-Shuffled	84.4	172.2	17.1 (**20.3%**)	32.4 (**38.4%**)
Basic-Fine-Shuffled	76.7	147.4	35.1 (**45.7%**)	9.5 (**12.4%**)
Bitwise-Fine-Shuffled	142.6	356	100.1 (**70.2%**)	9.5 (**6.7%**)

Experimental Results: NTT Perf. Evaluation

❑ **Shuffling Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)			
	Count	Overhead (%)	Shuffle.	Rand.
Kyber				
Unprotected	31.0	-	-	-
Coarse-Full-Shuffled	87.2	181.1	16.6 (**19%**)	**38.4 (44.1%)**
Coarse-In-Group-Shuffled	84.4	172.2	17.1 (**20.3%**)	32.4 (**38.4%**)
Basic-Fine-Shuffled	76.7	147.4	35.1 (**45.7%**)	9.5 (**12.4%**)
Bitwise-Fine-Shuffled	142.6	356	100.1 (**70.2%**)	9.5 (**6.7%**)

Experimental Results: NTT Perf. Evaluation

❑ **Shuffling Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)			
	Count	Overhead (%)	Shuffle.	Rand.
Kyber				
Unprotected	31.0	-	-	-
Coarse-Full-Shuffled	87.2	181.1	16.6 (**19%**)	38.4 (**44.1%**)
Coarse-In-Group-Shuffled	84.4	172.2	17.1 (**20.3%**)	32.4 (**38.4%**)
Basic-Fine-Shuffled	76.7	147.4	35.1 (**45.7%**)	9.5 (**12.4%**)
Bitwise-Fine-Shuffled	142.6	356	100.1 (**70.2%**)	9.5 (**6.7%**)

Experimental Results: NTT Perf. Evaluation

❑ **Shuffling Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)			
	Count	Overhead (%)	Shuffle.	Rand.
Dilithium				
Unprotected	55.2	-	-	-
Coarse-Full-Shuffled	120.2	117.5	18.9 (**15.7%**)	43.9 (**36.6%**)
Coarse-In-Group-Shuffled	114.6	107.5	19 (**16.6%**)	37.8 (**33%**)
Basic-Fine-Shuffled	109.7	98.7	42.2 (**38.5%**)	10.9 (**10%**)
Bitwise-Fine-Shuffled	630.9	1042.1	554.7 (**87.9%**)	10.9 (**1.7%**)

Experimental Results: NTT Perf. Evaluation

❑ **Shuffling Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)			
	Count	Overhead (%)	Shuffle.	Rand.
Dilithium				
Unprotected	55.2	-	-	-
Coarse-Full-Shuffled	120.2	117.5	18.9 (**15.7%**)	43.9 (**36.6%**)
Coarse-In-Group-Shuffled	114.6	107.5	19 (**16.6%**)	37.8 (**33%**)
Basic-Fine-Shuffled	109.7	98.7	42.2 (**38.5%**)	10.9 (**10%**)
Bitwise-Fine-Shuffled	630.9	1042.1	554.7 (**87.9%**)	10.9 (**1.7%**)

Experimental Results: NTT Perf. Evaluation

❑ **Shuffling Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)			
	Count	Overhead (%)	Shuffle.	Rand.
Dilithium				
Unprotected	55.2	-	-	-
Coarse-Full-Shuffled	120.2	117.5	18.9 (**15.7%**)	43.9 (**36.6%**)
Coarse-In-Group-Shuffled	114.6	107.5	19 (**16.6%**)	37.8 (**33%**)
Basic-Fine-Shuffled	109.7	98.7	42.2 (**38.5%**)	10.9 (**10%**)
Bitwise-Fine-Shuffled	630.9	1042.1	554.7 (**87.9%**)	10.9 (**1.7%**)

Experimental Results: NTT Perf. Evaluation

❑ **Shuffling Countermeasure:**

Countermeasures	KCycles ($\times 10^3$)			
	Count	Overhead (%)	Shuffle.	Rand.
Dilithium				
Unprotected	55.2	-	-	-
Coarse-Full-Shuffled	120.2	117.5	18.9 (**15.7%**)	43.9 (**36.6%**)
Coarse-In-Group-Shuffled	114.6	107.5	19 (**16.6%**)	37.8 (**33%**)
Basic-Fine-Shuffled	109.7	98.7	42.2 (**38.5%**)	10.9 (**10%**)
Bitwise-Fine-Shuffled	630.9	1042.1	554.7 (**87.9%**)	10.9 (**1.7%**)

Overall Evaluation: Kyber (Masked)

Masking Countermeasures	KCycles ($\times 10^3$)					
	KeyGen	Ohead (%)	Encaps	Ohead (%)	Decaps	Ohead (%)
Unprotected	1.178	-	1.301	-	1.358	-
Coarse-Masked	1.259	**6.9**	1.395	**7.2**	1.466	**7.9**
Generic-2-Masked	1.383	**17.5**	1.54	**18.3**	1.63	**20**
Generic-4-Masked	1.411	**19.8**	1.571	**20.7**	1.665	**22.5**
Generic-Random-Masked	1.507	**27.9**	1.676	**28.8**	1.764	**29.8**
Fine-Masked	1.979	**68**	2.229	**71.3**	2.413	**77.6**

Overall Evaluation: Kyber (Masked)

Masking Countermeasures	KCycles ($\times 10^3$)					
	KeyGen	Ohead (%)	Encaps	Ohead (%)	Decaps	Ohead (%)
Unprotected	1.178	-	1.301	-	1.358	-
Coarse-Masked	1.259	**6.9**	1.395	**7.2**	1.466	**7.9**
Generic-2-Masked	1.383	**17.5**	1.54	**18.3**	1.63	**20**
Generic-4-Masked	1.411	**19.8**	1.571	**20.7**	1.665	**22.5**
Generic-Random-Masked	1.507	**27.9**	1.676	**28.8**	1.764	**29.8**
Fine-Masked	1.979	**68**	2.229	**71.3**	2.413	**77.6**

Overall Evaluation: Kyber (Shuffled)

Shuffling Countermeasures	KCycles ($\times 10^3$)					
	KeyGen	Ohead (%)	Encaps	Ohead (%)	Decaps	Ohead (%)
Unprotected	1.178	-	1.301	-	1.358	-
Coarse-Full-Shuffled	1.534	**30.2**	1.72	**32.2**	1.841	**35.4**
Coarse-In-Group-Shuffled	1.49	**26.5**	1.664	**27.9**	1.772	**30.4**
Basic-Fine-Shuffled	1.468	**24.7**	1.643	**26.3**	1.752	**28.9**
Bitwise-Fine-Shuffled	1.878	**59.4**	2.123	**63.2**	2.303	**69.5**

Overall Evaluation: Kyber (Shuffled)

Shuffling Countermeasures	KCycles ($\times 10^3$)					
	KeyGen	Ohead (%)	Encaps	Ohead (%)	Decaps	Ohead (%)
Unprotected	1.178	-	1.301	-	1.358	-
Coarse-Full-Shuffled	1.534	**30.2**	1.72	**32.2**	1.841	**35.4**
Coarse-In-Group-Shuffled	1.49	**26.5**	1.664	**27.9**	1.772	**30.4**
Basic-Fine-Shuffled	1.468	**24.7**	1.643	**26.3**	1.752	**28.9**
Bitwise-Fine-Shuffled	1.878	**59.4**	2.123	**63.2**	2.303	**69.5**

Overall Evaluation: Dilithium (Masked)

Masking Countermeasures	KCycles ($\times 10^3$)			
	KeyGen	Ohead (%)	Encaps	Ohead (%)
Unprotected	2.626	-	15.144	-
Coarse-Masked	2.955	**12.5**	17.253	**32.9**
Generic-2-Masked	3.289	**25.2**	21.585	**66.3**
Generic-4-Masked	3.404	**29.6**	22.765	**75.3**
Generic-Random-Masked	3.594	**36.8**	23.082	**77.8**
Fine-Masked	4.781	**82.1**	39.752	**206.2**

Overall Evaluation: Dilithium (Masked)

Masking Countermeasures	KCycles ($\times 10^3$)			
	KeyGen	Ohead (%)	Encaps	Ohead (%)
Unprotected	2.626	-	15.144	-
Coarse-Masked	2.955	**12.5**	17.253	**32.9**
Generic-2-Masked	3.289	**25.2**	21.585	**66.3**
Generic-4-Masked	3.404	**29.6**	22.765	**75.3**
Generic-Random-Masked	3.594	**36.8**	23.082	**77.8**
Fine-Masked	4.781	**82.1**	39.752	**206.2**

Overall Evaluation: Dilithium (Shuffled)

Shuffling Countermeasures	KCycles ($\times 10^3$)			
	KeyGen	Ohead (%)	Encaps	Ohead (%)
Unprotected	1.178	-	15.144	-
Coarse-Full-Shuffled	3.206	**22.1**	20.581	**58.5**
Coarse-In-Group-Shuffled	3.159	**20.3**	20.257	**56**
Basic-Fine-Shuffled	3.101	**18.1**	20.865	**60.7**
Bitwise-Fine-Shuffled	7.802	**197.1**	76.614	**490.2**

Overall Evaluation: Dilithium (Shuffled)

Shuffling Countermeasures	KCycles ($\times 10^3$)			
	KeyGen	Ohead (%)	Encaps	Ohead (%)
Unprotected	1.178	-	15.144	-
Coarse-Full-Shuffled	3.206	**22.1**	20.581	**58.5**
Coarse-In-Group-Shuffled	3.159	**20.3**	20.257	**56**
Basic-Fine-Shuffled	3.101	**18.1**	20.865	**60.7**
Bitwise-Fine-Shuffled	7.802	**197.1**	76.614	**490.2**

Conclusion and Future Works

- We have presented concrete masking and shuffling countermeasures for the NTT.
- Generic Masked NTT construction with masked twiddle factors.
- Novel Shuffling countermeasures with varying granularity.
- Performance evaluation of protected Kyber and Dilithium on the ARM Cortex-M4 MCU.
- Performance Overhead (Clock Cycles):
  - **Kyber**: **7-78 %** across all procedures (KeyGen, Encaps, Decaps)
  - **Dilithium**: **12-197 %** for KeyGen and **32-490 %** for Sign
- Concrete security evaluation of the proposed countermeasures as part of future work.

Thank you!