NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

浙江大学 1897 ZHEJIANG UNIVERSITY

Fiddling the Twiddle Constants: Fault Injection Analysis of the Number Theoretic Transform

Prasanna Ravi[1] , Bolin Yang[3], Shivam Bhasin[1], Anupam Chattopadhyay[12], Fan Zhang[3]

[1]Temasek Labs, NTU, Singapore

[2]School of Computer Science and Engineering, NTU Singapore

[3]College of Information Science and Electronic Engineering, Zhejiang University, China

TCHES 2023, 11th September 2023

Outline

- Motivation
- FIA on Kyber
  - FIA on Key Generation
- FIA on Dilithium
  - FIA on Signing
  - FIA on Verification
- Conclusion

Outline

- ❑ **Motivation**
- ❑ FIA on Kyber
  - ❑ FIA on Key Generation
- ❑ FIA on Dilithium
  - ❑ FIA on Signing
  - ❑ FIA on Verification
- ❑ Conclusion

Motivation

- **Kyber and Dilithium** are Lattice-based schemes selected by NIST for PQC Standardization

- They share several common features:
  - Hardness based on **Module Learning With Error (MLWE)** Problem

  - Operate over **Similar Polynomial Rings,** leading to similar polynomial arithmetic operations

  - Share common **Building Blocks**:
    - Centered Binomial Sampler (CBD)
    - Number Theoretic Transform (NTT)

- Any implementation weakness in these building blocks will simultaneously affect both the schemes

- **NTT** operates over sensitive variables (secret key): attractive target for SCA and FIA
  - While NTT has been subjected to several types of SCA, so far no FIA has been performed

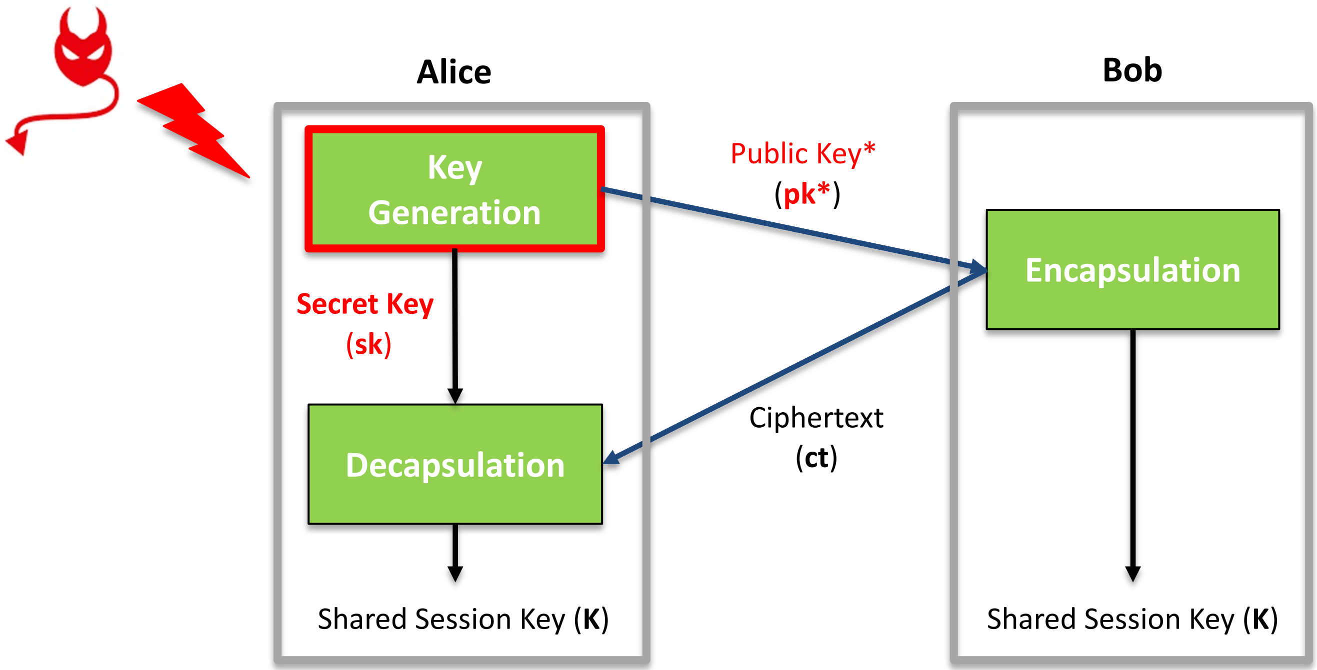
Our Work

- We proposed the first practical FIA on the NTT:
  - **New Fault Target:** **Single Point of Failure** in open-source NTT implementations for the ARM Cortex-M4 Microcontroller
  - Allows us to **zeroize all twiddle constants** of NTT using a **single fault**
    - Reduces the entropy of sensitive variables in Kyber and Dilithium
  - **Kyber**:
    - Key Recovery Attacks (Key Generation)
    - Message Recovery Attacks (Encapsulation)
  - **Dilithium:**
    - Signature Forgery Attacks (Signing)
    - Verification Bypass Attacks (Verification)
  - Experimentally validated using Electromagnetic Fault Injection (EMFI) with 100% success rate
  - Our attacks are able to bypass several fault injection countermeasures proposed for Kyber and Dilithium.

Outline

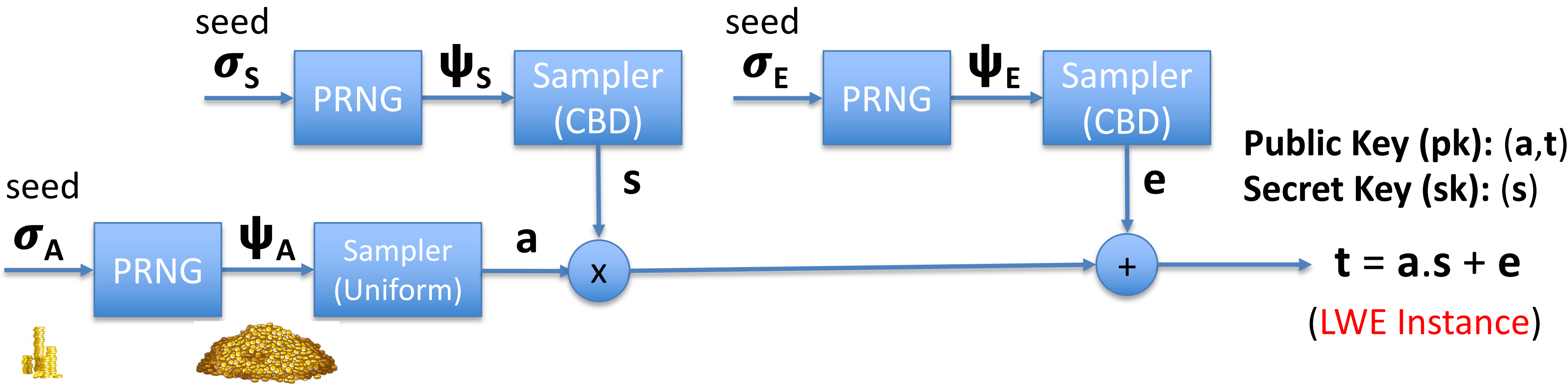
- ❑ Motivation
- ❑ FIA on Kyber
  - ❑ FIA on Key Generation
- ❑ FIA on Dilithium
  - ❑ FIA on Signing
  - ❑ FIA on Verification
- ❑ Conclusion

FIA on Kyber KeyGen

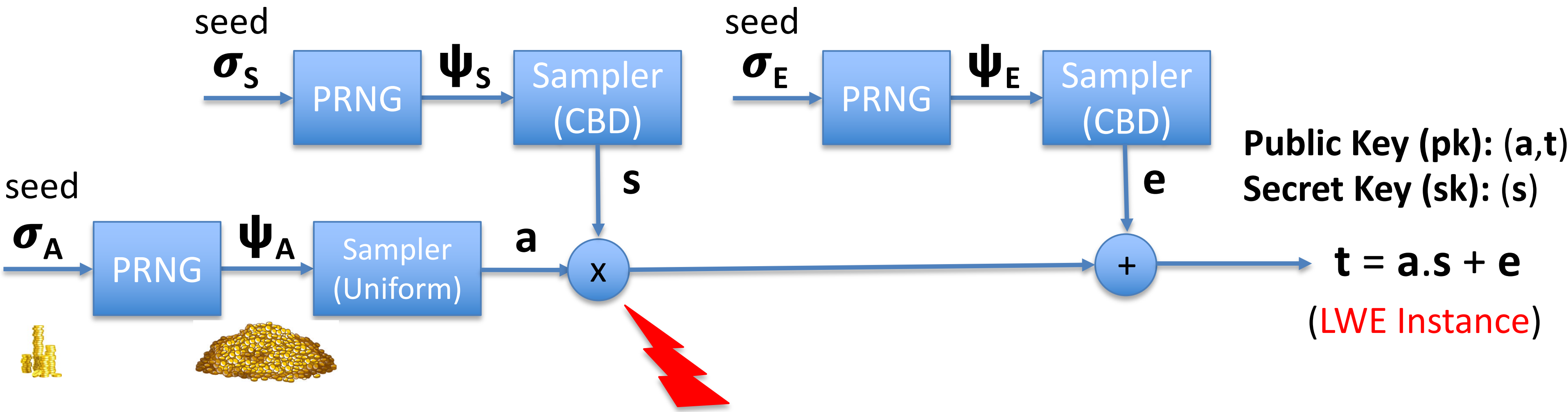


- ❑ Single execution to target Key Generation: Key Recovery Attack
  - ❑ Recover Secret key from Faulty but valid Public Key

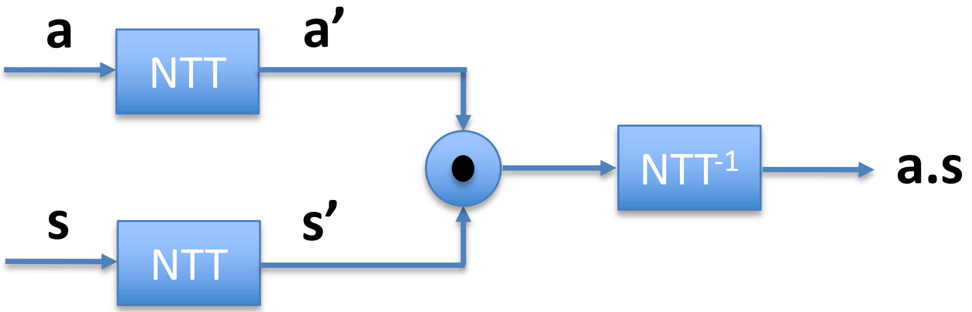
Kyber KeyGen



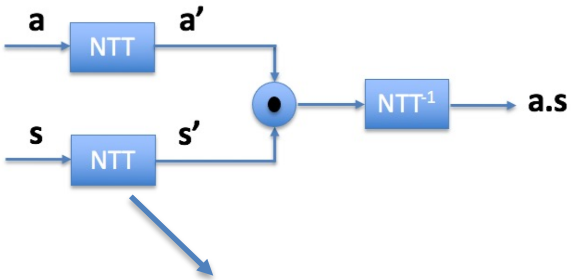
Kyber KeyGen

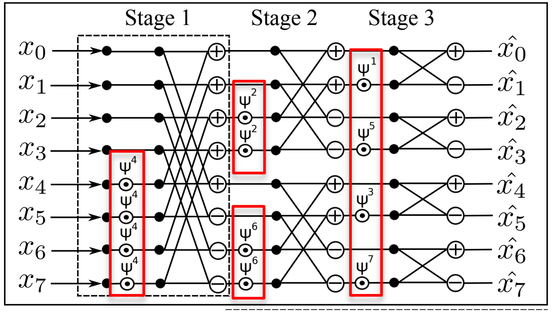


Polynomial multiplication is done using Number Theoretic Transform (NTT)



NTT Fault Vulnerability



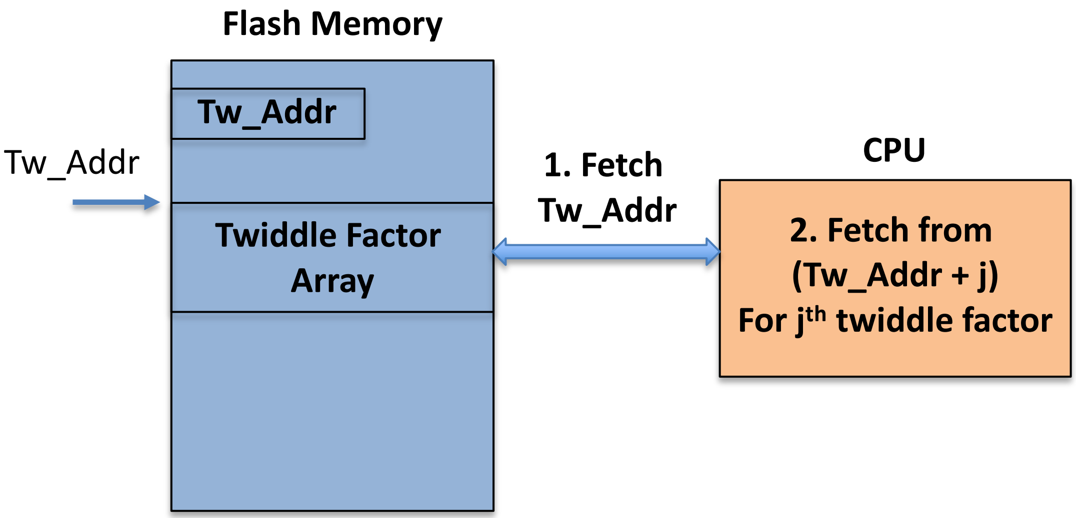


In MCU, Twiddle Constants are stored in Flash Memory as part of Firmware Binary

Manipulation of Twiddle Constants

Bare metal Software Implementation

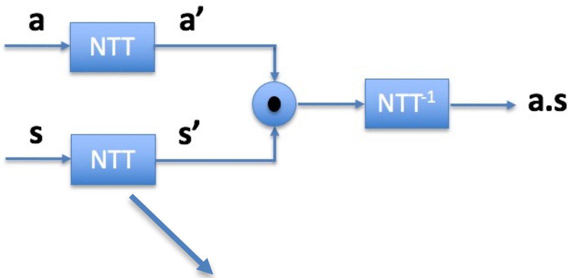
Implementation Style used in all open-source optimized implementations of Kyber and Dilithium for ARM Cortex-M4 Processor [BKS19, ABCG20, AHKS22, GKOS18, GKS21]

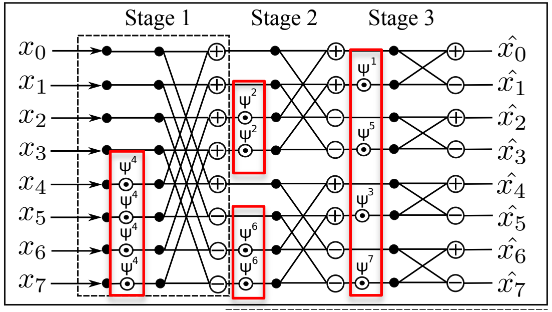


Main Observation: Tw_Addr is used as **base-address** to calculate address for all constants

Fault Vulnerability: Can an attacker fault the base address?

NTT Fault Vulnerability



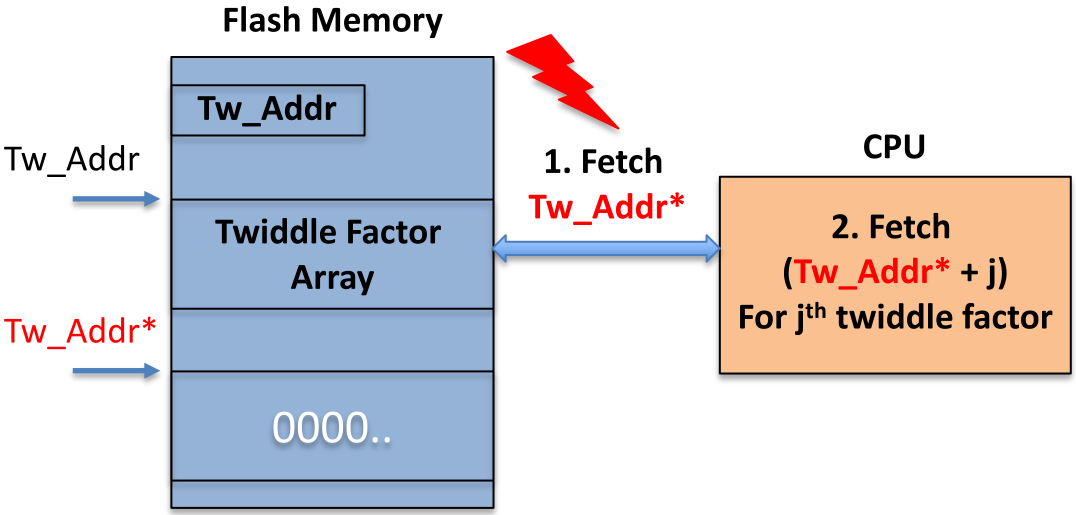


In MCU, Twiddle Constants are stored in Flash Memory as part of Firmware Binary

Manipulation of Twiddle Constants

Bare metal Software Implementation

Implementation Style used in all open-source optimized implementations of Kyber and Dilithium for ARM Cortex-M4 Processor [BKS19, ABCG20, AHKS22, GKOS18, GKS21]



Fault Model: Bit Set, Reset Faults on data transferred from flash memory [MBD+19]

Observation: Can zeroize the entire twiddle factor array in a single fault

25% of random memory locations yield **zeros** on ARM Cortex-M4 processor

[MBD+19] Menu, Alexandre, Shivam Bhasin, Jean-Max Dutertre, Jean-Baptiste Rigaud, and Jean-Luc Danger. "Precise spatio-temporal electromagnetic fault injections on data transfers." In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 1-8. IEEE, 2019.

NTT Fault Vulnerability: Zeroization of Twiddle Constants

Algorithm 3 Assembly Optimized NTT of Kyber in *pqm4* library [KRSS19] (Simplified)

```
ldr r1, [pc, #4]                          ▷ Loading twiddle-ptr from address (pc+4) to register r1

                 ***Start of NTT Assembly Routine***

n ← 16
while n > 0 do                            ▷ First stage (Stage 1,2,3)
    load poly
    ldrh twiddle, [twiddle-ptr]           ▷ Loading twiddle from twiddle-ptr
    doublebutterfly (poly, twiddle)
    ldr twiddle, [twiddle-ptr, #2]        ▷ Loading twiddle from (twiddle-ptr+2)
    doublebutterfly (poly, twiddle)
    ···
    n − −
end while
add twiddle-ptr, #14                      ▷ Incrementing twiddle-ptr by 14 for next stage
n ← 8
while n > 0 do                            ▷ Second stage (Stage 4,5,6)
    m ← 2
    while m > 0 do
        load poly
        ldrh twiddle, [twiddle-ptr]       ▷ Loading twiddle from twiddle-ptr
        doublebutterfly (poly, twiddle)
        ldr twiddle, [twiddle-ptr, #2]    ▷ Loading twiddle from (twiddle-ptr+2)
        doublebutterfly (poly, twiddle)
        ···
        m − −
    end while
    add twiddle-ptr, #14                  ▷ Incrementing twiddle-ptr by 14 for next stage
    n − −
    ···                                   ▷ Last stage (Stage 7)
end while
```

Load Twiddle Pointer from Flash

Load from **Twiddle Pointer***

Load from **Twiddle Pointer***

Increment **Twiddle Pointer***

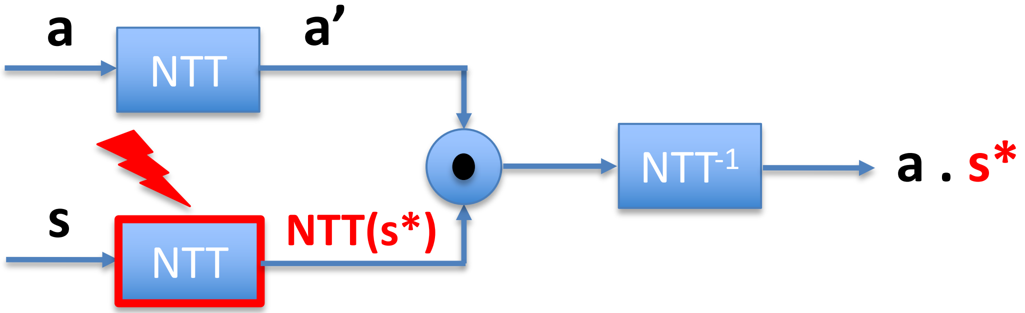
Load from **Twiddle Pointer***

Load from **Twiddle Pointer***

Increment **Twiddle Pointer***

Corrupts all twiddle constants

NTT Fault Vulnerability: Zeroization of Twiddle Constants



Valid Secret, but with Low Entropy

Faulty secret s*

s0	s1	0	0	0	0	0	0

Entropy reduces by half in every NTT layer

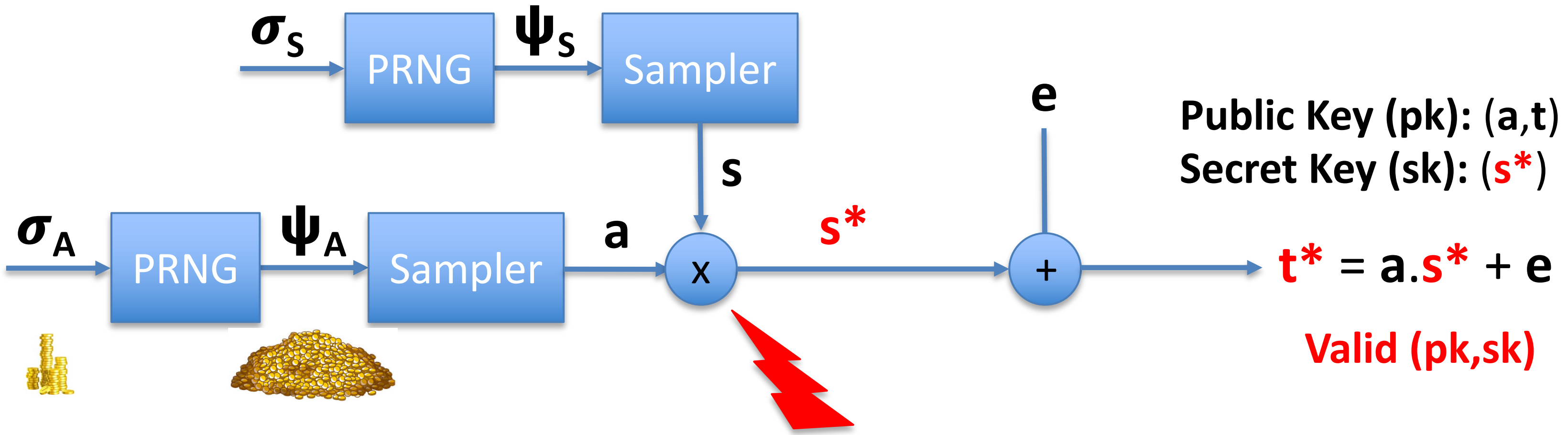
s0	s1	s2	s3	s4	s5	s6	s7

s0	s1	s0	s1	s0	s1	s0	s1

- Secret **s** has **k** polynomials
  - **k** NTTs
- But, we experimentally observed that fault on one NTT is sufficient
- Maybe faulty twiddle pointer is cached and reused for **k** NTTs

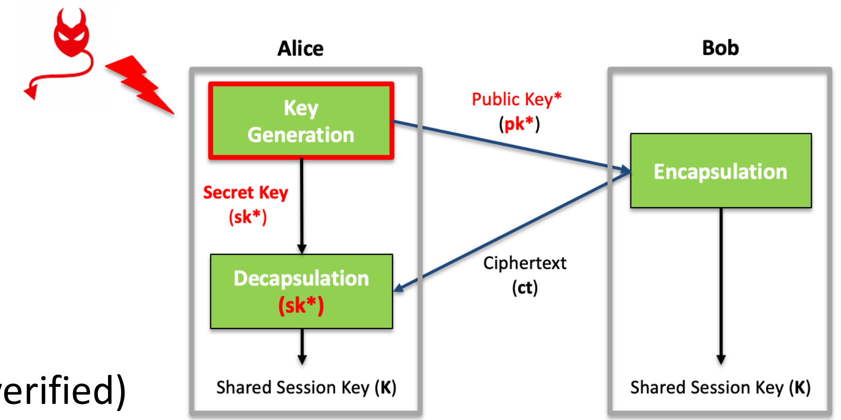
- Kyber uses Incomplete NTT
  - 7 layers (256 point NTT)
  - Two non zero coeff. at NTT output
- Dilithium uses complete NTT
  - 8 layers (256 point NTT)
  - One non-zero coeff. At NTT output

FIA on Kyber KeyGen: Zeroization of Twiddle Constants



- Same Secret (s*) in NTT domain is used for **Decaps**
  - To avoid extra NTT/INTT conversions
  - Originally sampled secret **s** is forgotten!!!
  - Memoryless property of Kyber

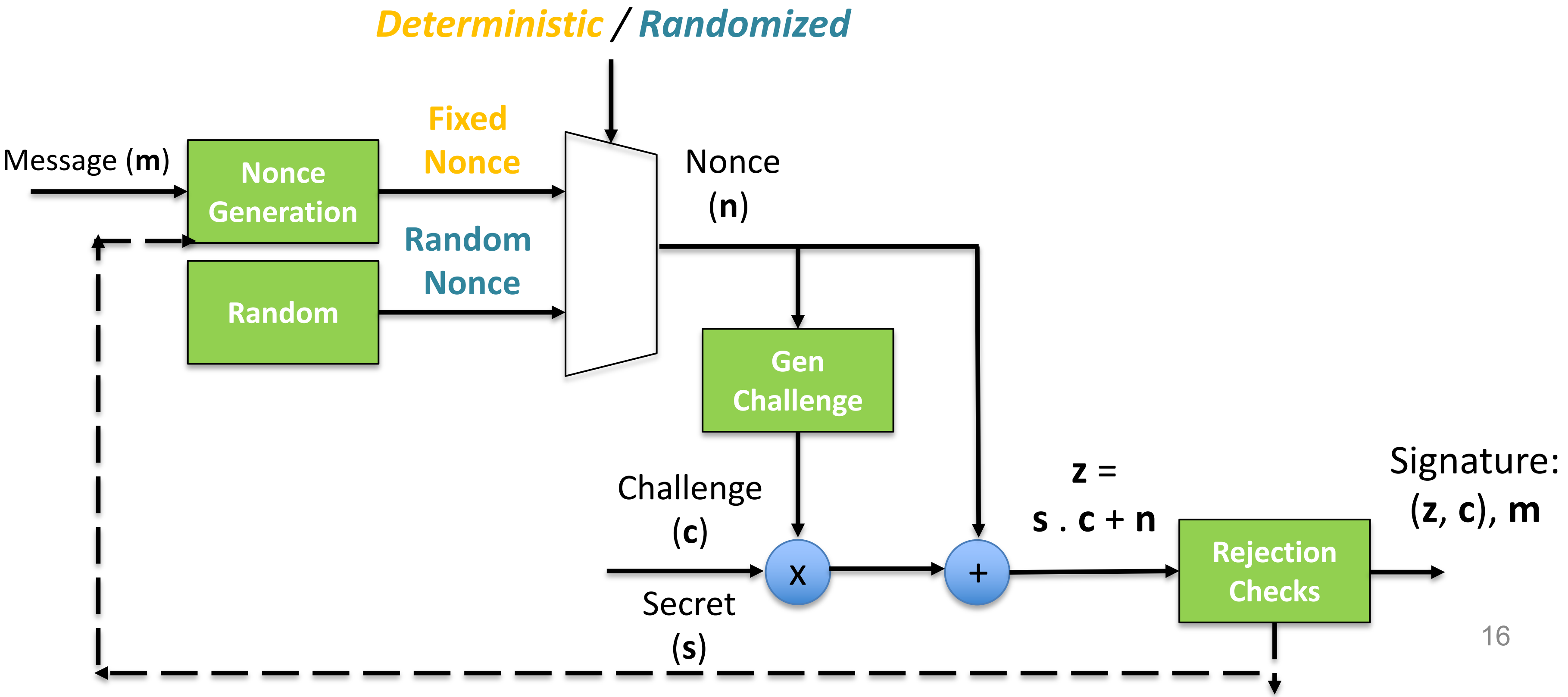
- Attack also applies to masked implementations
  - Repeat Same Fault on All Shares (Experimentally verified)



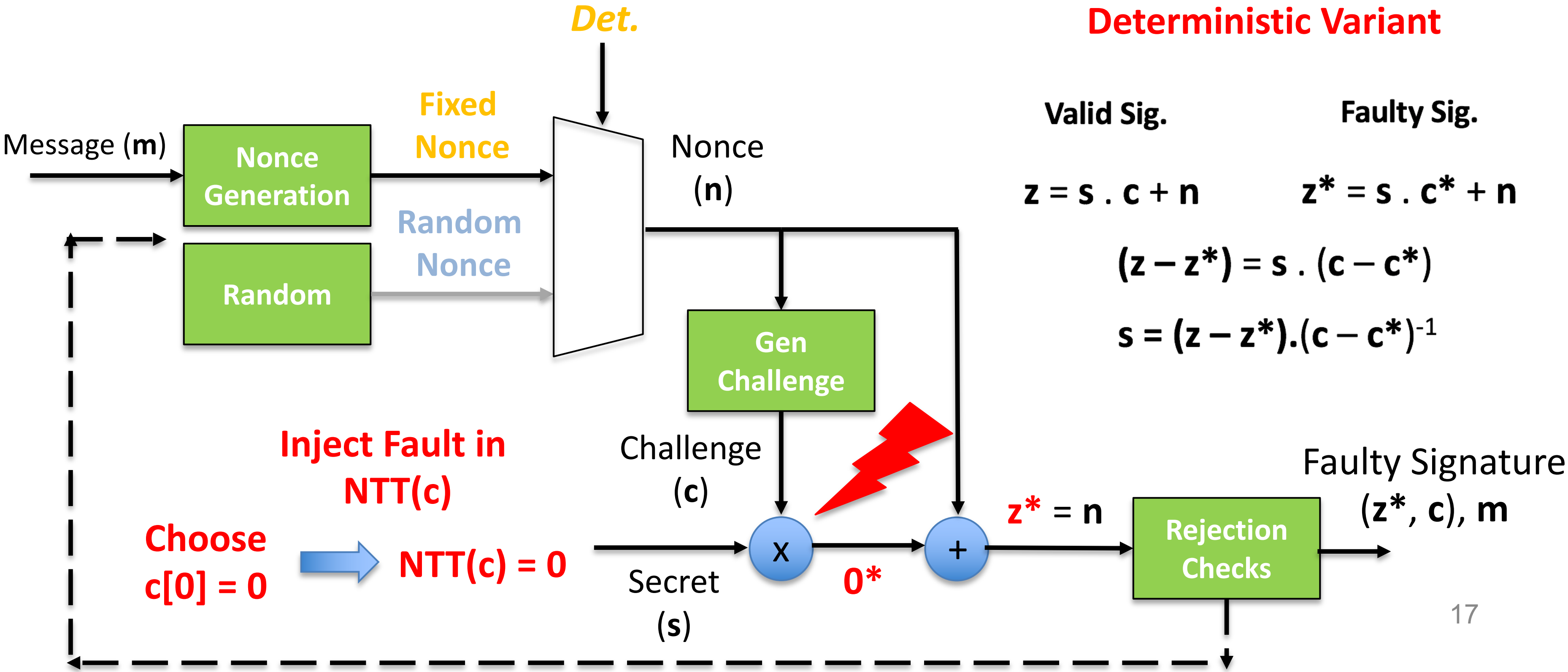
Outline

- FIA on Kyber:
  - FIA on Key Generation
- **FIA on Dilithium**
  - **FIA on Signing**
  - FIA on Verification
- Conclusion

FIA on Dilithium Signing: Background



FIA on Dilithium Signing: Deterministic Variant



DFA Attack on Deterministic Variant

Valid Sig.	Faulty Sig.
$z = s \cdot c + n$	$z^* = s \cdot c^* + n$

$$(z - z^*) = s \cdot (c - c^*)$$

$$s = (z - z^*) \cdot (c - c^*)^{-1}$$

FIA on Dilithium Signing: Randomized Variant

Impl. Variant: z is computed in the **NTT Domain**

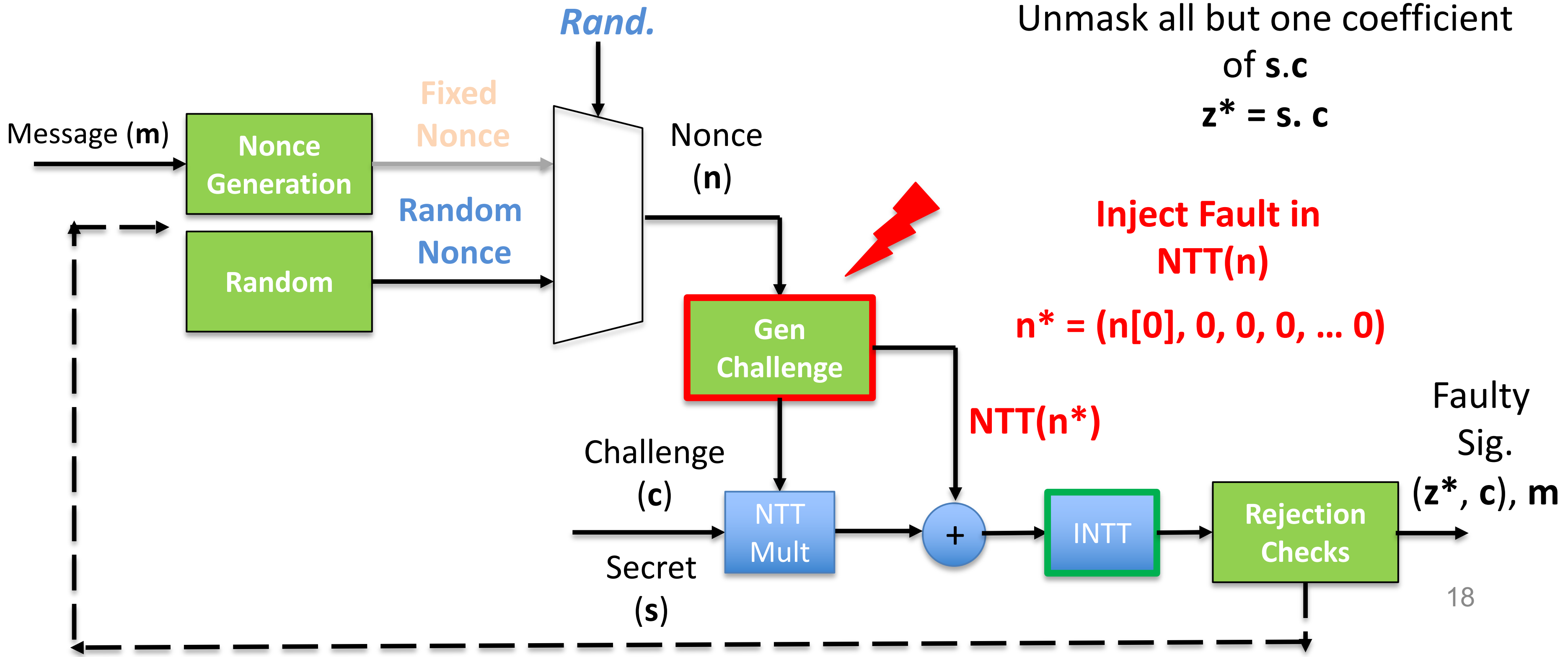
$$\mathbf{z} = INTT(\ (NTT(\mathbf{n}) + NTT(\mathbf{s}).\ NTT(\mathbf{c})\ )$$

Unmask all but one coefficient of **s.c**

z* = s. c

Inject Fault in NTT(n)

n* = (n[0], 0, 0, 0, … 0)



Outline

- ❑ FIA on Kyber:
  - ❑ FIA on Key Generation
- ❑ **FIA on Dilithium**
  - ❑ FIA on Signing
  - ❑ **FIA on Verification**
- ❑ Conclusion

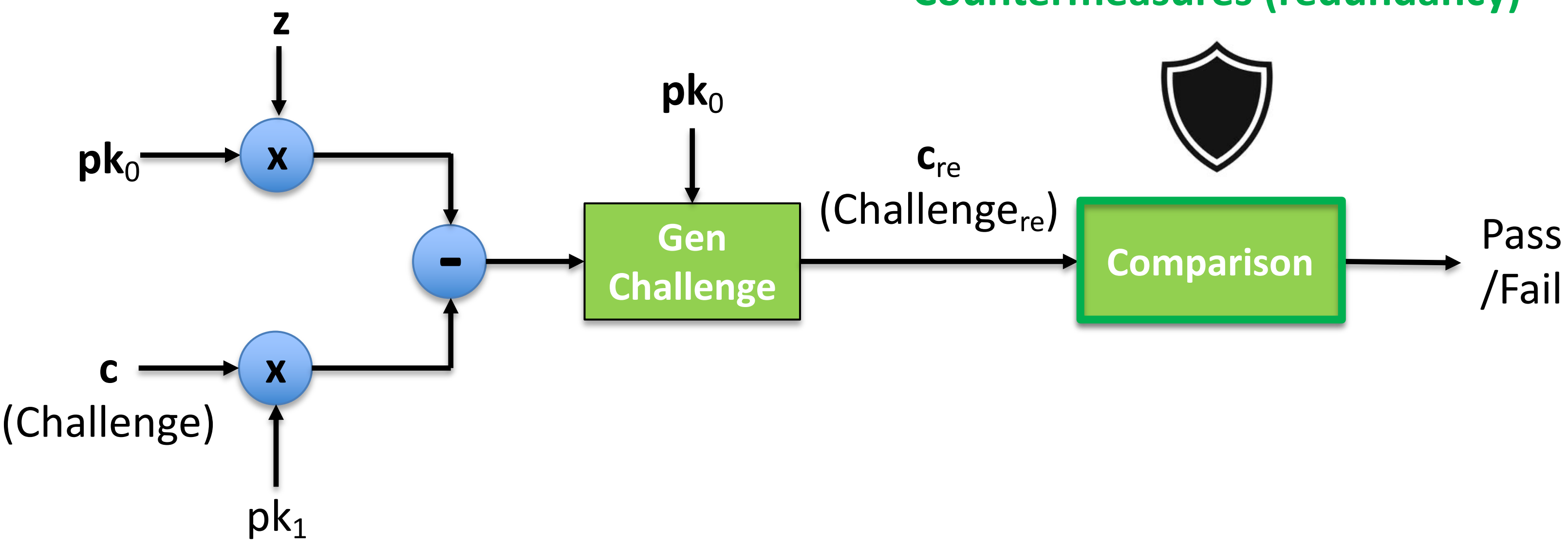
FIA on Verification Procedure

Signature: (z, c) , m

Public Key: (pk_0, pk_1)

Most Obvious Target

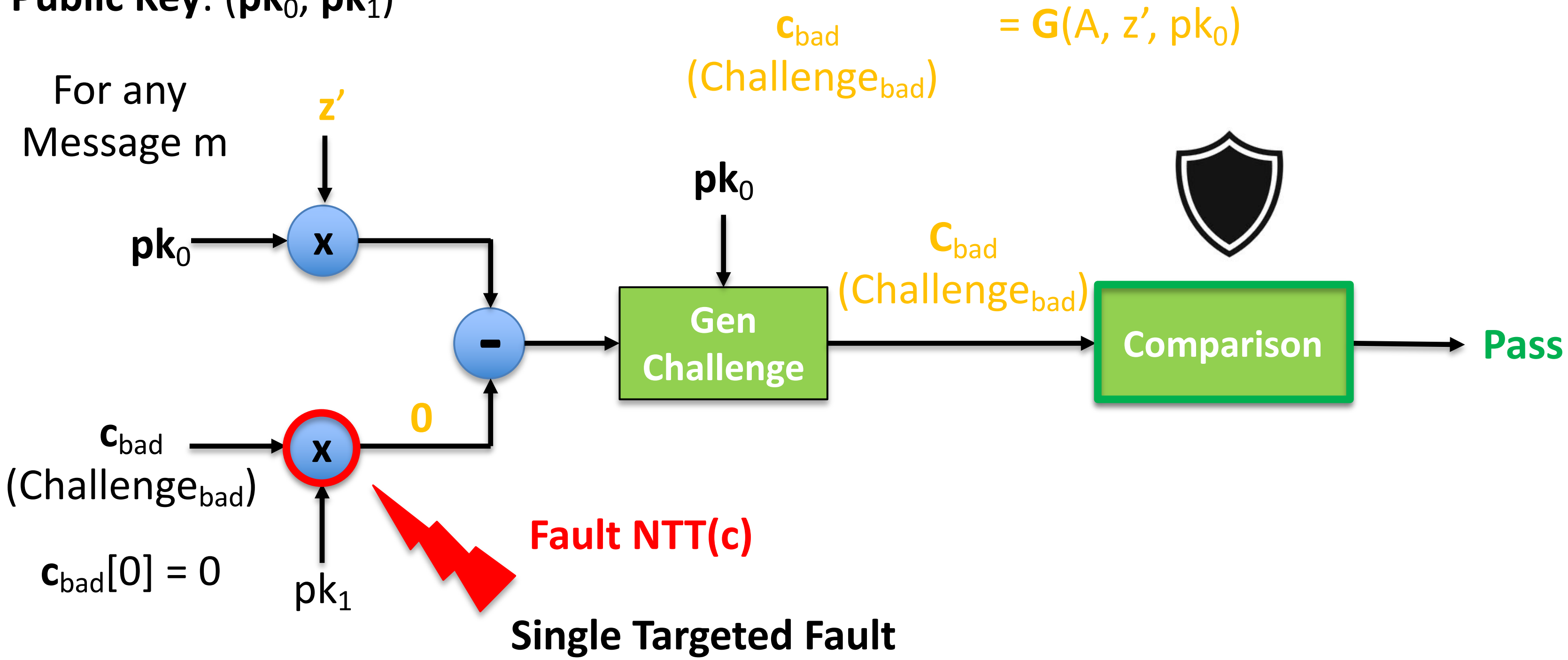
Can be fortified with known Countermeasures (redundancy)



FIA on Verification Procedure

Signature: (**z', c_{bad}**) , **m**

Public Key: (**pk_0, pk_1**)



Experimental Evaluation

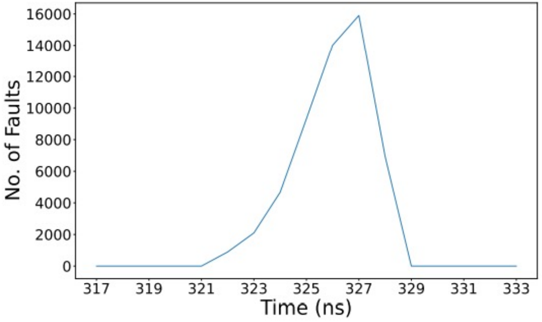
- Experimental validation was done using Electromagnetic Fault Injection (EMFI)
- **Target**:
  - Optimized implementations of Kyber and Dilithium from the *pqm4* library [KRSS19] on the ARM Cortex-M4 MCU
- We were able to achieve 100% fault repeatability using several fault parameters
  - 25% of random memory locations in the memory space fetch zero twiddle factor arrays
  - Very repeatable fault can be achieved when targeting data transfer from flash memory [MBD+19]
- Our attack is orthogonal to fault countermeasures against prior FIA on Kyber and Dilithium

[KRSS19] Kannwischer, Matthias J., Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. "pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4." (2019).

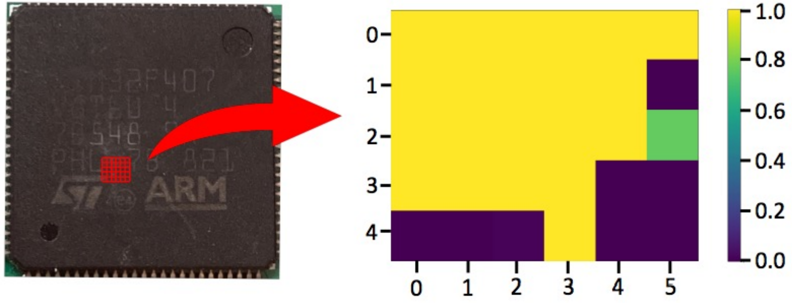
[MBD+19] Menu, Alexandre, Shivam Bhasin, Jean-Max Dutertre, Jean-Baptiste Rigaud, and Jean-Luc Danger. "Precise spatio-temporal electromagnetic fault injections on data transfers." In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 1-8. IEEE, 2019.

Experimental Evaluation (Kyber)

Key Generation

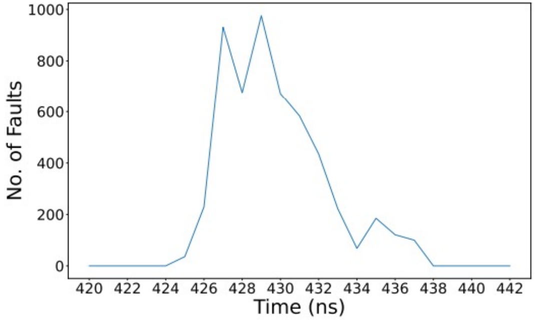


Number of Faults Versus Time

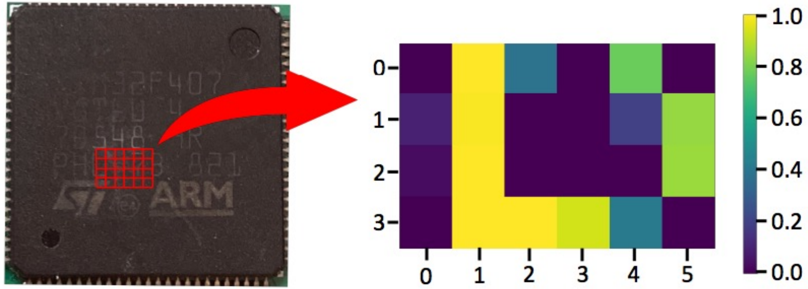


Fault Repeatability versus Location

Encapsulation



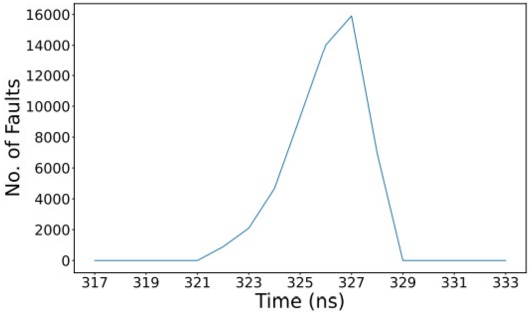
Number of Faults Versus Time



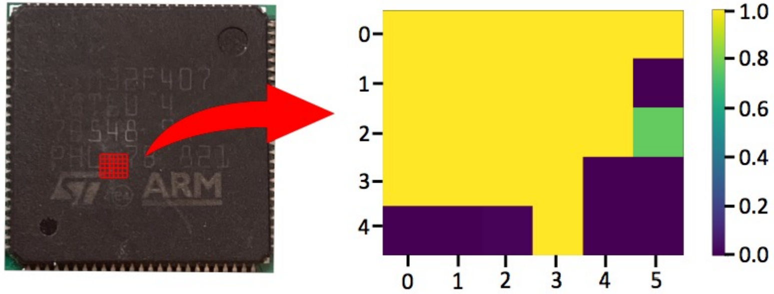
Fault Repeatability versus Location

Experimental Evaluation (Dilithium)

Signing (Deterministic)

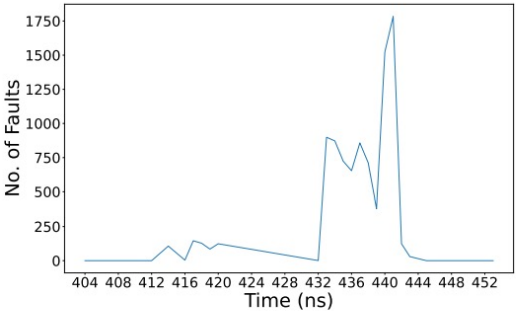


Number of Faults Versus Time

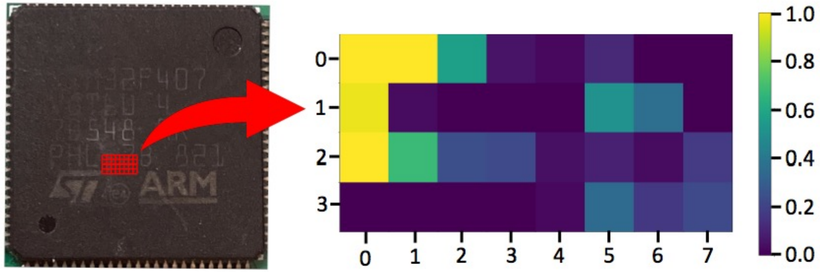


Fault Repeatability versus Location

Signing (Probabilistic)



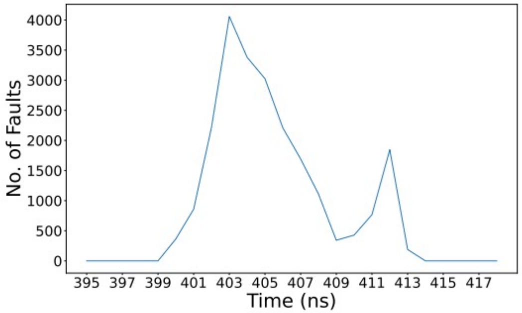
Number of Faults Versus Time



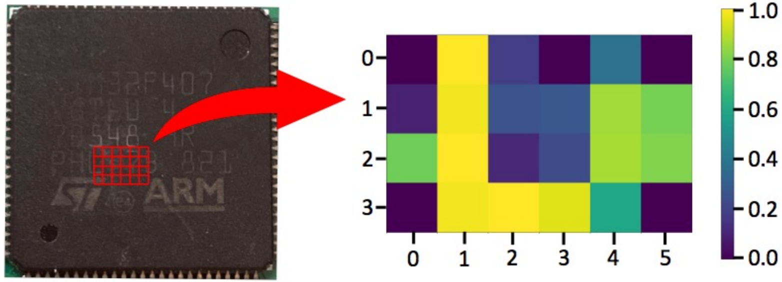
Fault Repeatability versus Location

Experimental Evaluation (Dilithium)

Verification



Number of Faults Versus Time

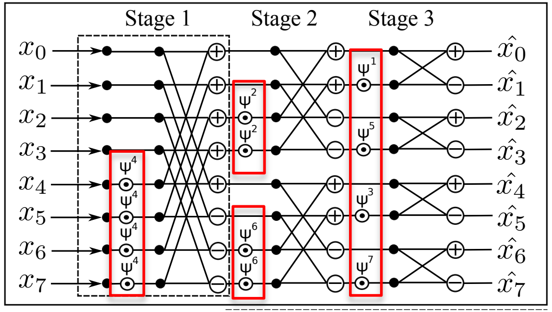


Fault Repeatability versus Location

Outline

- ❑ FIA on Kyber:
    - ❑ FIA on Key Generation
- ❑ FIA on Dilithium
    - ❑ FIA on Signing
    - ❑ FIA on Verification
- ❑ **Countermeasures**
- ❑ Conclusion

Countermeasures: Protect NTT against FIA



- **Countermeasure-1**: Sanity Check on Twiddle Constants
  - Check Arithmetic Properties of Twiddle Constants:
    - n^{th} root of unity
  - Check Entropy of Twiddle Constants

- **Countermeasure-2**: Sanity Check on NTT Outputs
  - Check Entropy of NTT Outputs

- **Countermeasure-3**: Do rely on single base address to access Twiddle Constant Array

Outline

- FIA on Kyber:
  - FIA on Key Generation
- FIA on Dilithium
  - FIA on Signing
  - FIA on Verification
- Countermeasures
- **Conclusion**

Conclusion

- In this work, we proposed the first practical FIA on the NTT:
  - **Single Point of Failure** in assembly-optimized NTT implementations for Kyber and Dilithium
  - Allows to **zeroize entire twiddle factor array with a single fault**
  - Practical Attacks on Kyber and Dilithium
  - Practical experimental validation using EMFI on implementations of Kyber and Dilithium in the pqm4 library with 100% success rate
  - Our attack is able to circumvent several fault countermeasures for Kyber and Dilithium
  - Dedicated countermeasures for the NTT implementation are necessary to defeat the attack

Thank you!

Prasanna Ravi,
Temasek Labs, NTU Singapore

E-mail: prasanna.ravi@ntu.edu.sg
GitHub: https://github.com/PRASANNA-RAVI