# Safe-Error Analysis of Post-Quantum **Cryptography Mechanisms**

Luk Bettale, Simon Montoya and Guénaël Renault





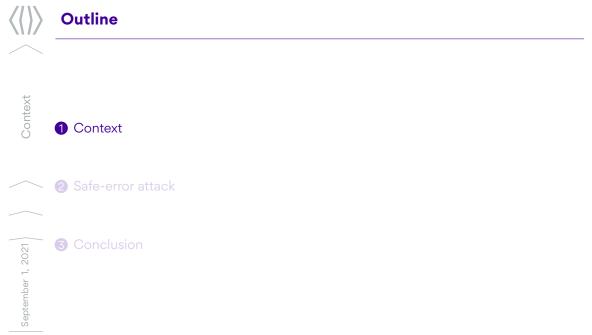








#### Conclusion





- NIST launched in 2016 a call for PQ safe crypto.
  - Key Exchange Mechanism (KEM).
  - Signature.
- Algorithms for a future standardization.
- Here we focus on embedded devices.

Context



Context

- Context
- NIST launched in 2016 a call for PQ safe crypto.
  - Key Exchange Mechanism (KEM).
  - Signature.
- Algorithms for a future standardization.
- Here we focus on embedded devices.

#### **Embedded devices**

- Less RAM and power consumption.
- Lattice-based schemes seems suitable for embedded devices.



- NIST launched in 2016 a call for PQ safe crypto.
  - Key Exchange Mechanism (KEM).
  - Signature.
- Algorithms for a future standardization.
- Here we focus on embedded devices.

#### **Embedded devices**

- Less RAM and power consumption.
- Lattice-based schemes seems suitable for embedded devices.
- In threat of physical attacks:
  - Side-channel.
  - Fault injection.
- Fault injection for PQC has not been much investigated.



# Context

#### Safe-error attack

- Safe-error attack (SEA) is a way to use fault injection.
  - Specific fault may or not lead to a faulty output.
  - The faulty or not output gives information.
- Very efficient against constant time implementation.



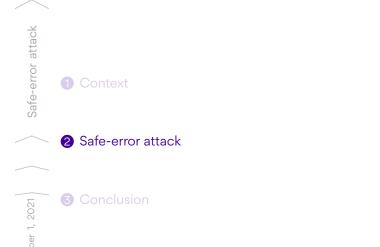
# Context

#### Safe-error attack

- Safe-error attack (SEA) is a way to use fault injection.
  - Specific fault may or not lead to a faulty output.
  - The faulty or not output gives information.
- Very efficient against constant time implementation.
- In our context the attacker can:
  - Set the fault to a target operation.
  - Skip an instruction or function call.
  - Set a variable to O.
- Our work focus on NTRU, Saber, Dilithium and Kyber.



# Outline





#### Tool for security analysis

- NIST PQC mentioned 5 security categories: 1 to 5.
- However, candidates under/over estimates these categories.

#### Tool for security analysis

- NIST PQC mentioned 5 security categories: 1 to 5.
- However, candidates under/over estimates these categories.
- Then we use the toolkit LWE with side information (L. Ducas, H. Gong and M. Rossi).
- Allow to determine the security lost due to side-channel information.
- The security estimation:  $bikz \beta$ .
  - Correspond to the BKZ- $\beta$  to solve DBDD instance.
  - **•** No conversion between  $\beta$  and bits.



# Safe-error attack

## High-level attack

- Lattice-based finalists secret distribution  $\Rightarrow$  numerous null coeffs.
- Our goal: retrieve the null coefficients.
- Focus on the sign or decrypt algorithms.

# High-level attack

- Lattice-based finalists secret distribution  $\Rightarrow$  numerous null coeffs.
- Our goal: retrieve the null coefficients.
- Focus on the sign or decrypt algorithms.
- The attack procedure:
  - I Find a function where each secret coefficient is manipulated.
  - 2 Fault the operation.
  - If the output is unchanged: coeff = 0.
  - 4 Else: coeff  $\neq$  0.

September 1, 2021



# NTRU

## Focus on poly mult.

■ Our goal: retrieve O-coeffs of *f*.

Alg	orithm 1 Polynomial Multiplication
Inp	ut: a, c, f
Ou	tput: a
1:	for <i>k</i> = 0 to <i>n</i> do
2:	$a[k] \leftarrow 0$
3:	<b>for</b> <i>i</i> = 1 to <i>n</i> <b>do</b>
4:	$a[k] \leftarrow a[k] + c[k+i] \times f[n-i]$
5:	end for
6:	for <i>i</i> = 0 to <i>k</i> + 1 <b>do</b>
7:	$a[k] \leftarrow a[k] + c[k - i]  imes f[i]$
8:	end for
9:	end for
10:	return a

2021

September 1,



# NTRU

- The secret poly f has coefficients in  $\{-1, 0, 1\}$  (uniform).
- Fault injection during a poly mult with *f*.



# NTRU

- The secret poly f has coefficients in  $\{-1, 0, 1\}$  (uniform).
- Fault injection during a poly mult with *f*.
- We suppose that the secret coeffs are well distributed: n/3 are 0.

	Classical	Attacked
NTRU HPS 1	Dim = 1018	Dim = 680
n = 509, q = 2048	$\beta = 172.15$	$\beta = 95.53$
NTRU HPS 2	Dim = 1354	Dim = 904
<i>n</i> = 677, <i>q</i> = 2048	$\beta = 249.95$	$\beta = 146.20$
NTRU HPS 3	Dim = 1642	Dim = 1096
n = 821q, q = 4096	$\beta = 308.42$	$\beta = 183.35$
NTRU HRSS	Dim = 1402	Dim = 936
<i>n</i> = 701, <i>q</i> = 8192	$\beta = 236.30$	$\beta = 135.96$

■ In average SEA: 42% security loss.



## Saber

- The secret poly *s* has coefficients in  $\{-\sigma, \ldots, \sigma\}$  (binomial).
- Fault injection during conversion byte to poly.



## Saber

- The secret poly *s* has coefficients in  $\{-\sigma, \ldots, \sigma\}$  (binomial).
- Fault injection during conversion byte to poly.
- We suppose that the secret coeffs are well distributed.

	Classical	Attacked
Light Saber	Dim = 1025	Dim = 900
$n, m = 512, \sigma = 5$	$\beta = 404.38$	$\beta = 292.05$
Saber	Dim = 1537	Dim = 1328
$n, m = 768, \sigma = 4$	$\beta = 648.72$	$\beta = 462.57$
Fire Saber	Dim = 2049	Dim = 1729
$n, m = 1024, \sigma = 3$	$\beta = 892.21$	$\beta = 613.26$

■ In average SEA: 30% security loss.



# Dilithium

• The secret poly  $s_1, s_2$  have coefficients in  $\{-\sigma, \ldots, \sigma\}$  (binomial).

Fault injection during conversion byte to poly.



# Dilithium

- The secret poly  $s_1, s_2$  have coefficients in  $\{-\sigma, \ldots, \sigma\}$  (binomial).
- Fault injection during conversion byte to poly.
- We suppose that the secret coeffs are well distributed.

	Classical	Attacked
Dilithium 1	Dim = 2049	Dim = 1281
(n,m) = (1024, 1024)	$\beta = 348.84$	$\beta = 192.84$
$\sigma = 2$		
Dilithium 2	Dim = 2817	Dim = 2049
(n,m) = (1280, 1536)	$\beta = 499.65$	$\beta = 340.06$
$\sigma = 4$		
Dilithium 3	Dim = 3841	Dim = 2401
(n,m) = (1792, 2048)	$\beta = 717.52$	$\beta = 411.13$
$\sigma = 2$		

■ In average SEA: 40% security loss.







#### **3** Conclusion



# Conclusion

#### **Countermeasures**

- Mask the secret distribution (as Kyber with NTT representation).
- Shuffling.



#### Countermeasures

- Mask the secret distribution (as Kyber with NTT representation).
- Shuffling.

#### Conclusion

- Determine the security impact of SEA against lattice-based crypto.
- Decrease significantly the theoretical security.
- Without additional knowledge  $\Rightarrow$  difficult to retrieve the entire secret key.
- However, SEA + others side-channel leakage could be devastating.

Conclusion



