Side-Channel Attacks on Lattice-Based Cryptography: Attacks and Countermeasures

Olivier Bronchain

NewTPQC – Oxford –June 2024

Our contribution to literature

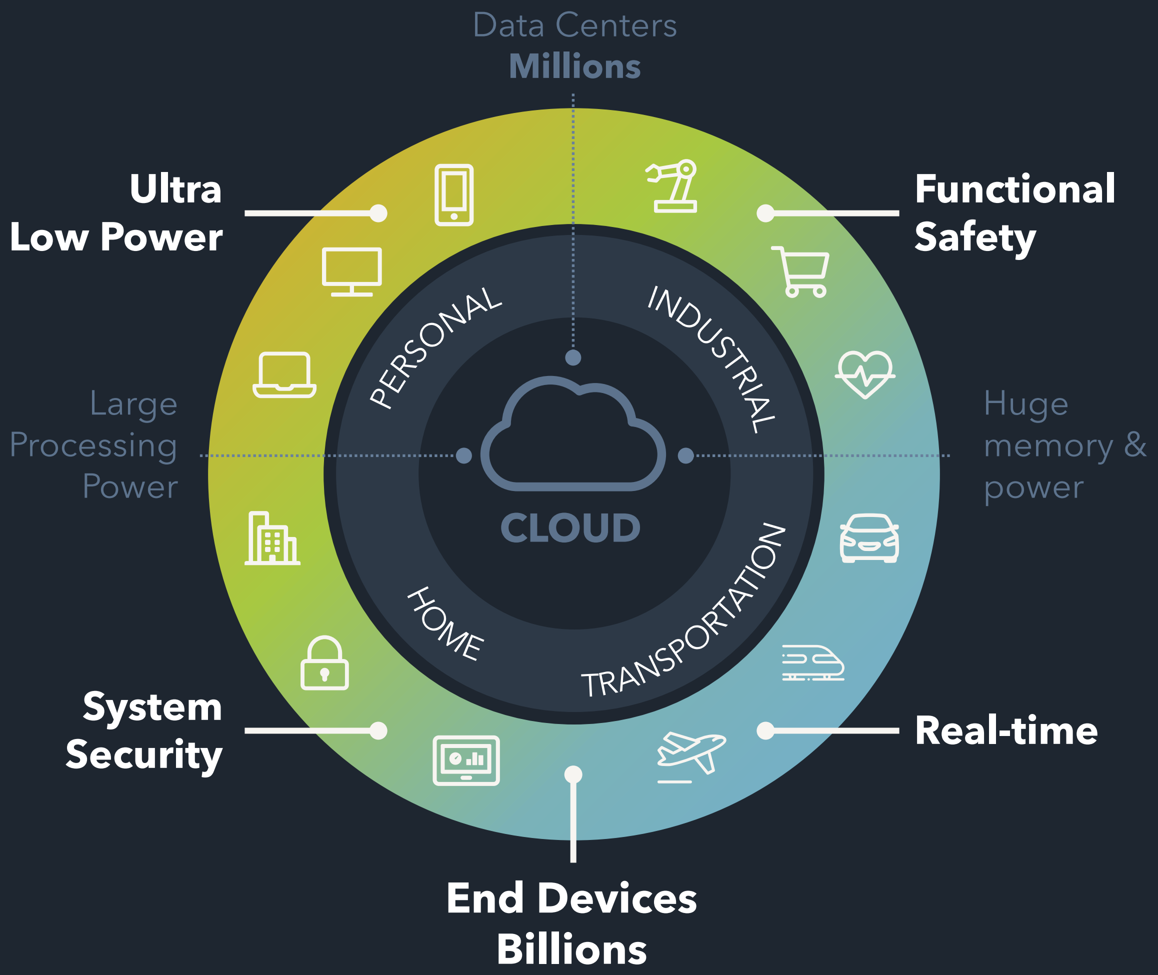
NXP’s Authors:

- Melissa Azouaoui
- Joppe Bos
- Olivier Bronchain
- Christine Cloostermans
- Frank Custers
- Mohamed ElGhamrawy
- Marc Gourjon
- Joost Renes
- Tobias Schneider
- Markus Schönauer
- Amber Sprenkels
- Denise Verbakel
- …

NXP’s Publications:

- *’’Exploiting Small-Norm Polynomial Multiplication with Physical Attacks: Application to CRYSTALS-Dilithium’’,* Bronchain et al, TCHES 2024.
- ’’From MLWE to RLWE: A Differential Fault Attack on Randomized & Deterministic Dilithium’’, ElGhamrawy et al., TCHES 2023
- ‘’Protecting Dilithium Against Leakage Revisited Sensitivity Analysis and Improved Implementations’’, Azouaoui et al., TCHES 2023.
- ‘’Enabling FrodoKem for embedded devices’’, Bos et al., TCHES 2023
- “Post-Quantum Secure Over-the-Air Update of Automotive Systems”, Bos et al., Escar 2023.
- “Post-Quantum Authenticated Encryption against Chosen-Ciphertext Side-Channel Attacks”, Azouaoui et al., TCHES 2022
- ‘’Post-Quantum Secure Boot on Vehicle Network Processors”, Bos et al., Escar 2022.
- “Dilithium for Memory Constrained Devices”, Bos et al, AfricaCrypt 2022.
- “Masking Kyber: First- and Higher-Order Implementations”, Bos et al., TCHES 2021
- …

IMPACT PQC ON OUR ECO-SYSTEM



Data collection, processing and decisions at the edge
Devices securely connected to the cloud

No Silver Bullet
If a crypto scheme was better, we would have standardized this already

Cryptographic Keys
Orders of magnitude larger.
Dilithium secret key up to 4.8KB
(ECC: 32 bytes, RSA: 384 bytes)

Performance
Varies: some faster, some significantly slower.
SHA-3 is a dominating component (~80%)

Memory
Orders of magnitude more.

Bandwidth & Power
Larger signatures (up to 4.6KB) → more bandwidth required → increase in power usage




INDUSTRIAL

- Fit-for-purpose Scalable Processors
- Functional Safety & Security
- Industrial Connectivity & Control
- Machine Learning & Vision
- Comprehensive Software

PQC ON EMBEDDED DEVICES

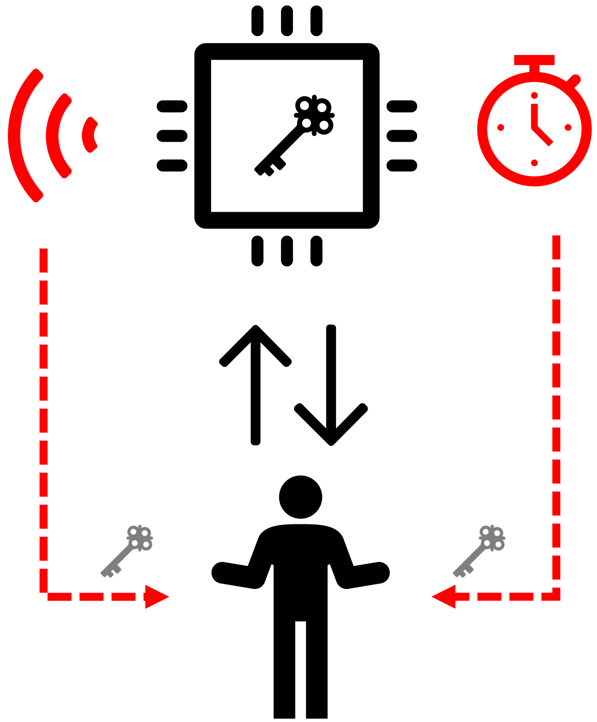
Pqm4: Post-quantum crypto library for the ARM Cortex-M4, STM32F4DISCOVERY
196 KiB of RAM and 1 MiB of Flash ROM

The fastest implementations in pqm4 require ≈ 49, ≈ 80 and ≈ 116 KiB memory for Dilithium-{2,3,5}.

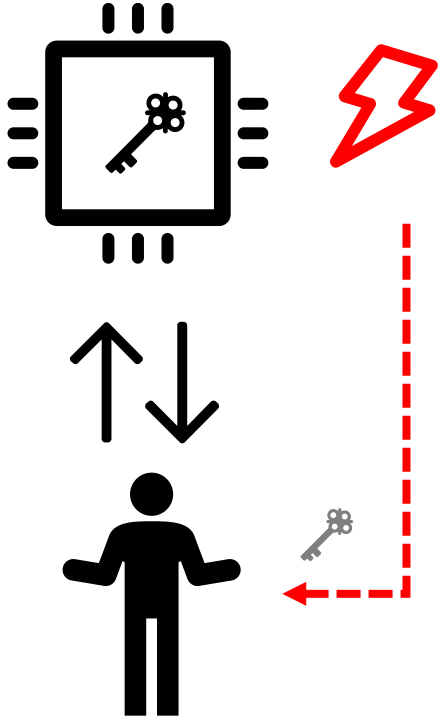
Low-power edge computing: NXP LPC800 Series

- 8 to 60 MHz Cortex-M0+ core
- $\{4, 8, 16\}$ KiB of SRAM
- $\{16, 32\}$ KiB Flash

Embedded implementation attacks



Side-Channel Attacks (SCA)



Fault Attacks (FA)

Embedded cryptography and implementation attacks

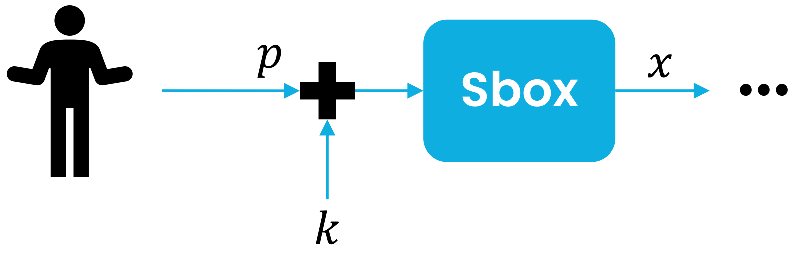
Attacks		Countermeasures
Deep understanding in both academia and industry from decades of research.	**Current Asymmetric Cryptography** ECDSA RSA ECC	Practically secure and certified implementations.
Active research area resulting in increasingly powerful attacks.	**Post-Quantum Cryptography** Dilithium Kyber Falcon SPHINCS+ XMSS	Early stage of academic research. Limited industrial results.

01

Side-channel attacks

Background

Introduction to side-channel attack on a toy block-cipher

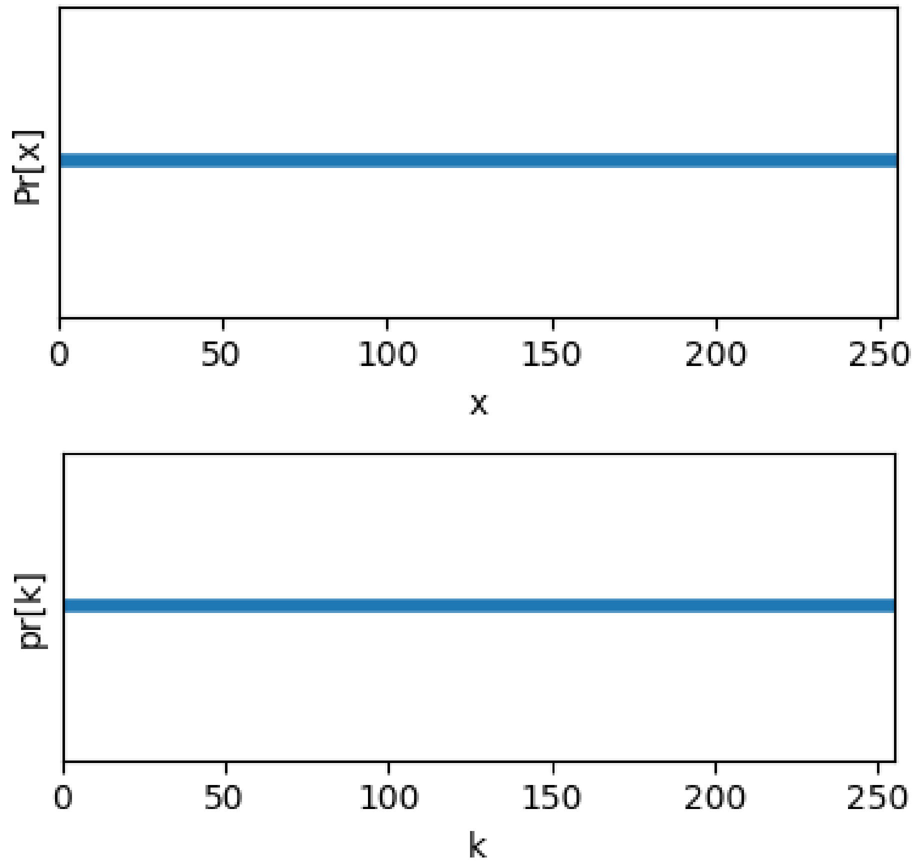


<u>Let's take an toy block-cipher:</u>

- 8-bit plaintext p.
- 8-bit key k.
- 8-bit Sbox output x.

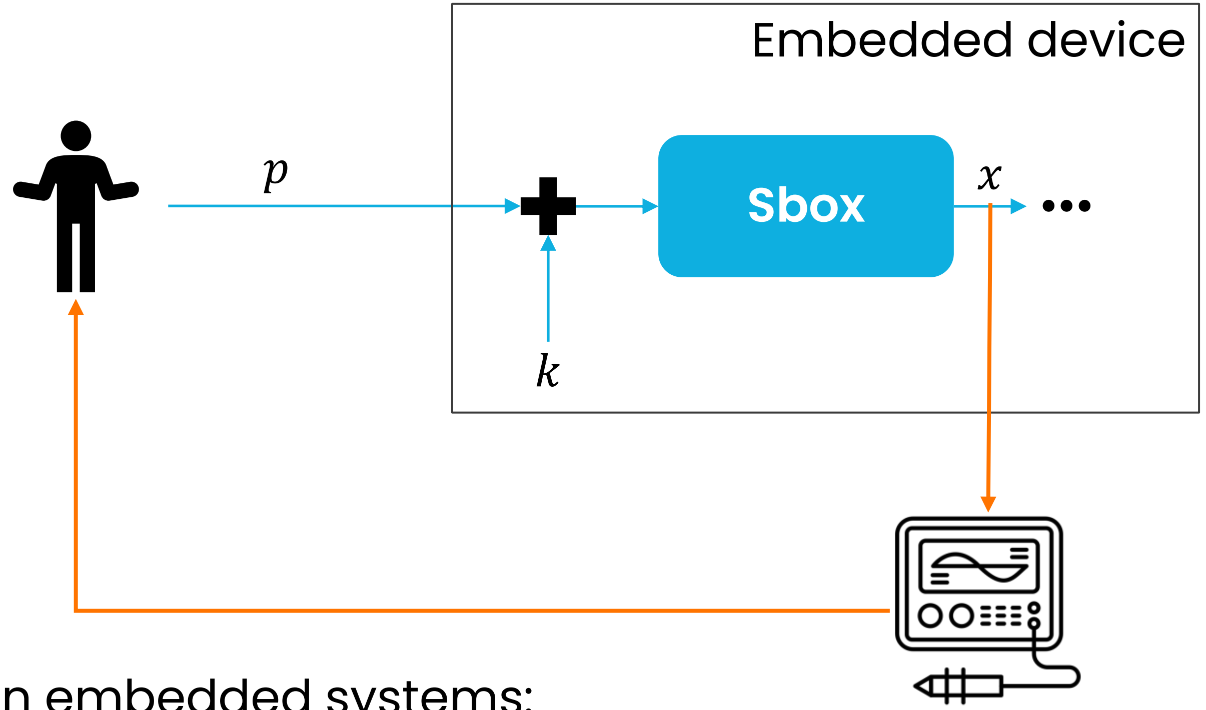
<u>In a black-box setting:</u>

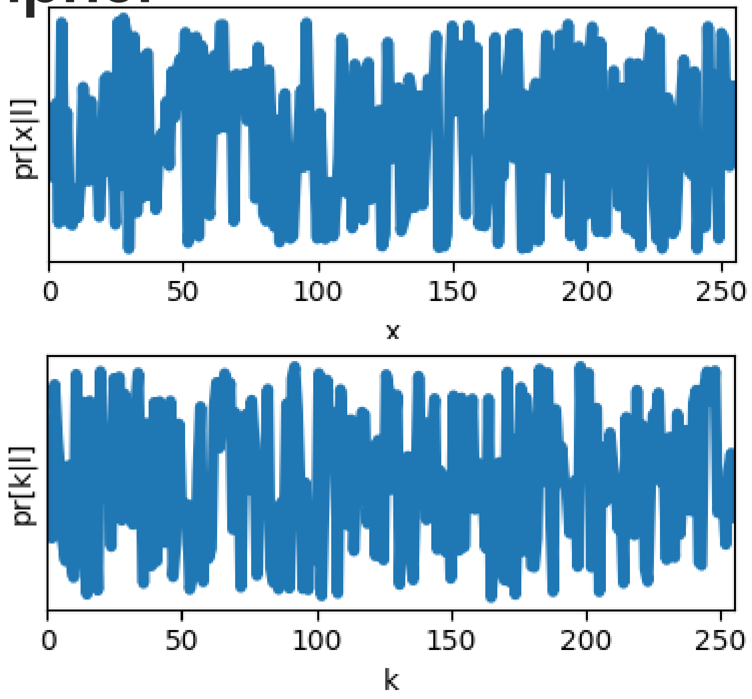
- k is uniform as sampled uniformly.
- x is uniform because of XOR with secret key.



Prob. Distribution of variables

Introduction to side-channel attack on a toy block-cipher



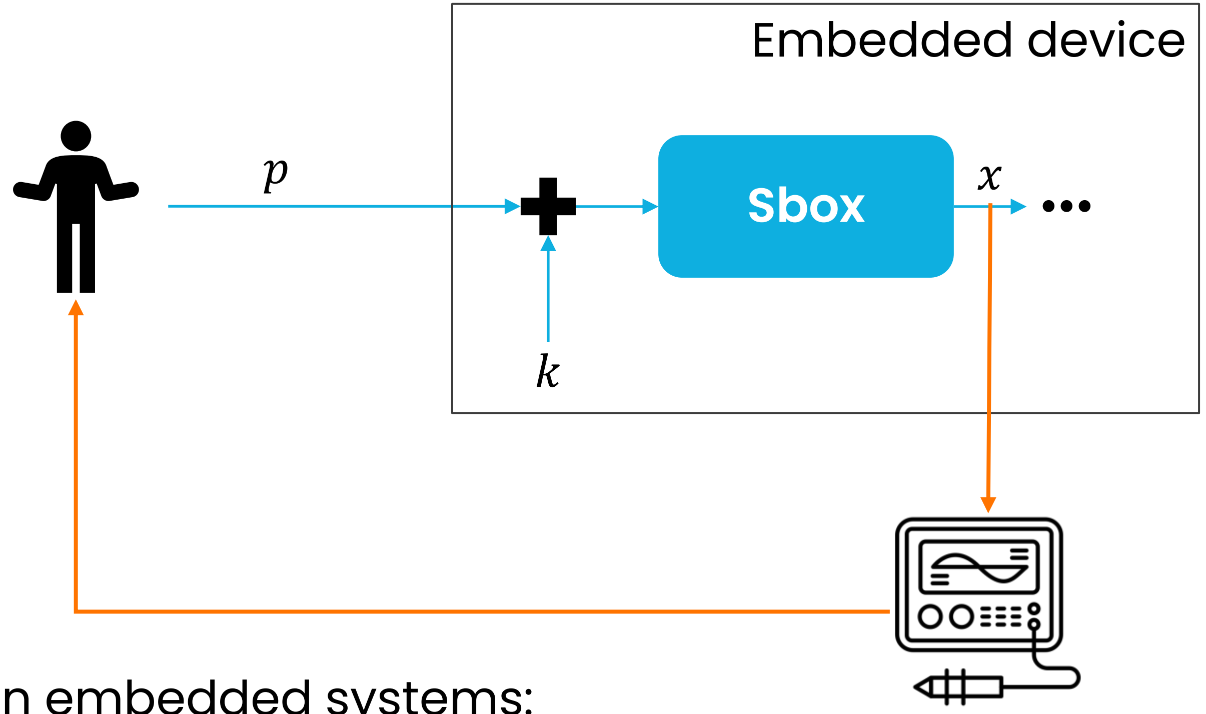


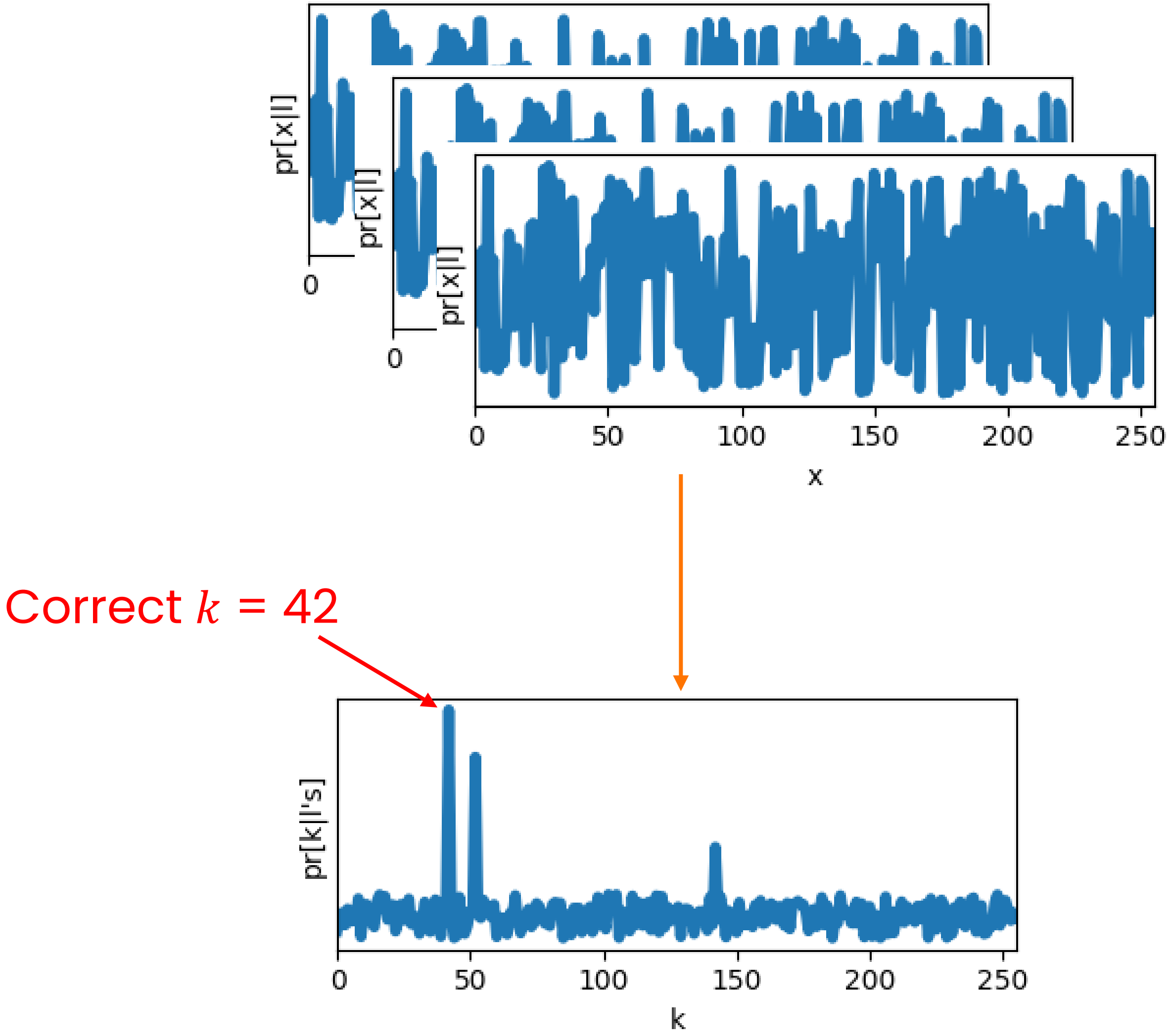
Prob. Distribution of variables

<u>In embedded systems:</u>

1. Adversary sends a plaintext p.
2. x generates some leakages such as power or EM.
3. The adversary records these leakages.
4. Adversary samples the posterior distribution of x.
5. Adversary derives the posterior distribution of k.

Introduction to standard template attack DPA.





Prob. Distribution of variables

<u>In embedded systems:</u>

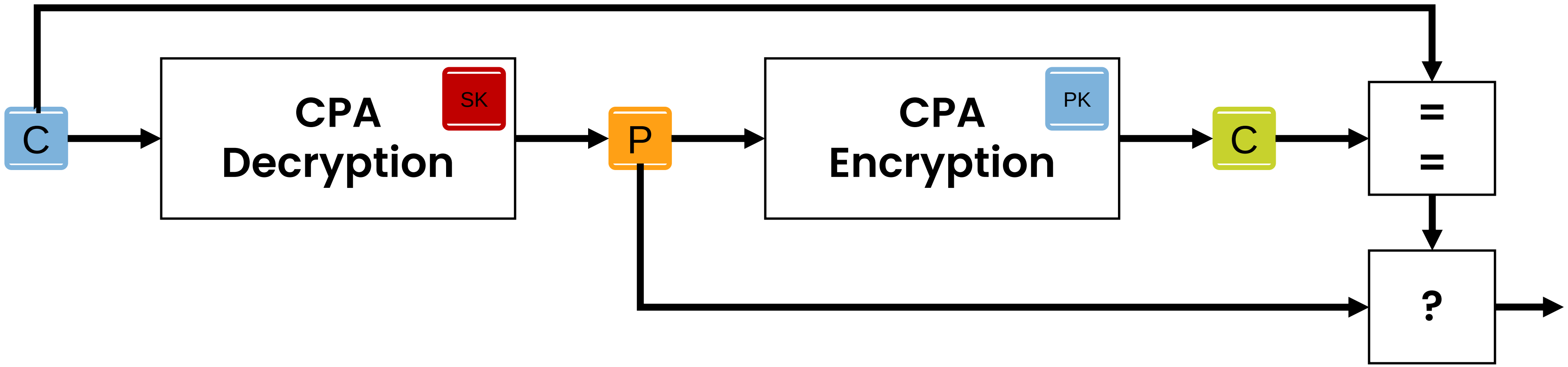
1. Adversary sends a plaintext p.
2. x generates some leakages such as power or EM.
3. The adversary records these leakages.
4. Adversary samples the posterior distribution of x.
5. Adversary derives the posterior distribution of k.
6. Repeat the process to obtain the correct k.

01

Side-channel attacks

Kyber

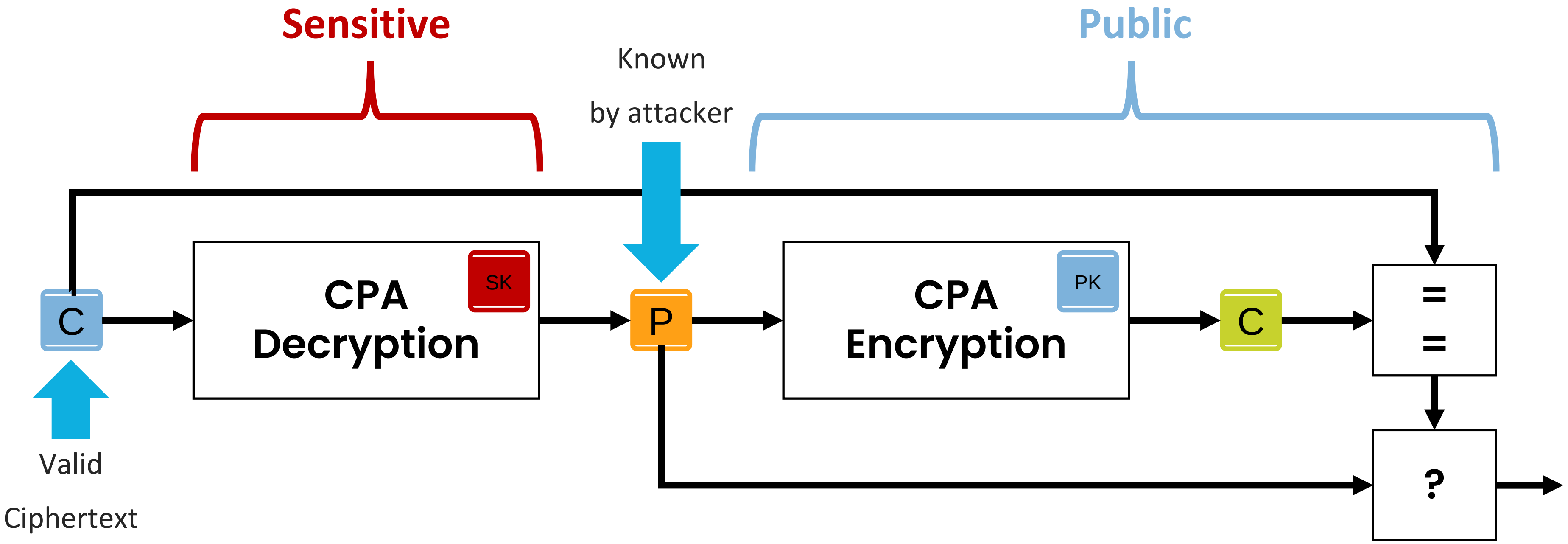
Kyber Overview: Fujisaki Okamoto transform



Kyber Overview: the SCA Problem OF the FO-Transform

Attack 1: Chosen Plaintext

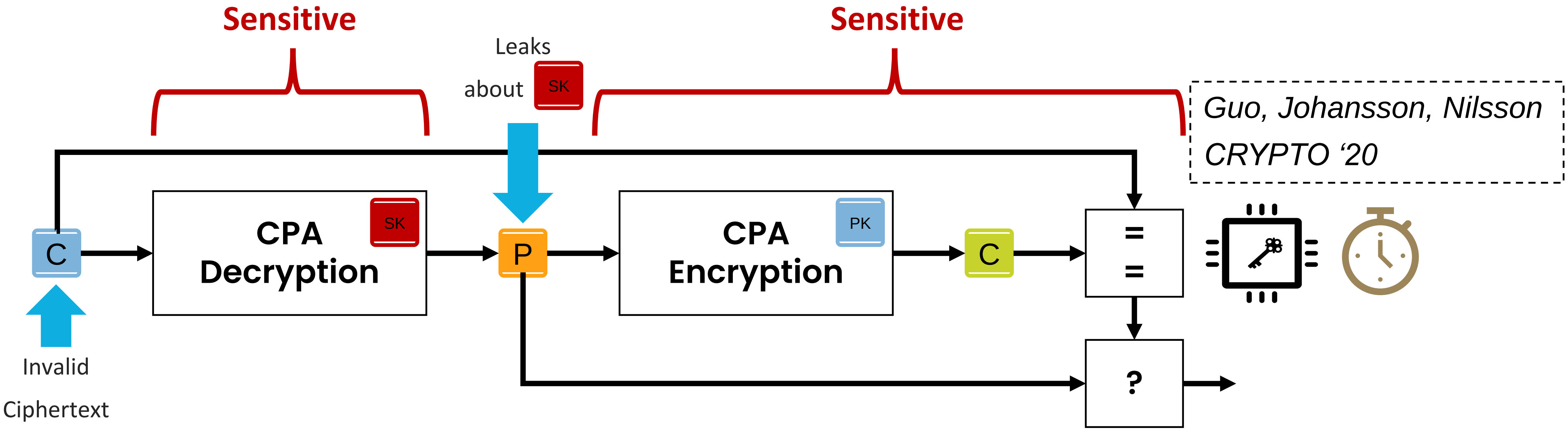
- Attacker inputs only valid ciphertexts
- Attack focuses on **CPA Decryption**, everything after (and including) P is public
- Only need to protect **CPA Decryption**



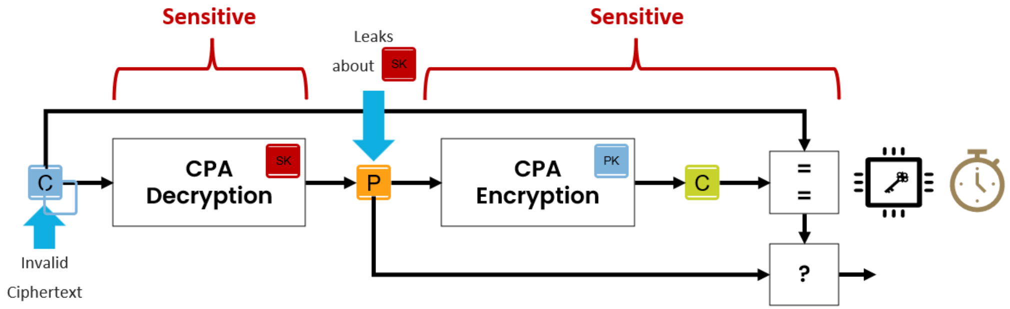
Kyber Overview: the SCA Problem OF the FO-Transform

Attack 2: Chosen Ciphertext

- Attacker inputs specially-crafted invalid ciphertexts
- Attack focuses on **CPA Decryption** + everything after (and including) P is potentially sensitive
- Potentially all (or most) modules need to be hardened



Kyber-Overview: the SCA Problem OF the FO-Transform



Attack 2: Chosen Ciphertext (example)

1. Attacker inputs specially-crafted invalid ciphertexts $\boxed{C}$ such that:
   - $\boxed{P}$ = b|000000 if $s_i \in \{0,1,2\}$, with $b = 1$
   - $\boxed{P}$ = b|000000 if $s_i \in \{-2,-1\}$, with $b = 0$
2. Attack gets the leakage from the all CPA-Encryption($\boxed{P}$) to recover b.
3. After recovering b, the attacker knows some about the coefficient s_i.
4. Repeat the same process for different subset for s_i and recover exact s_i value.
5. Repeat for each of the coefficients.

<u>Improvement tracks in the literature:</u>

- Recover information for more coefficients at once: b is a multiple bit target.
- Resilient to miss classification of b.
- Reduce the number of b to be recovered per coefficient.
- Combine with sieving not to have to recover all the coefficients.

Challenge of protecting Kyber against SCA

Block-cipher vs. KEMs:

- Block-cipher:
  - Only a small portion of the intermediates can efficiently be exploited:
  - Due to the size of the key guesses to perform and diffusion.
- KEMs:
  - A single bit must be distinguished hence the size of the key-guess don’t increase.
  - Every single block can be exploited equally.

Operations attacked in the literature:

- Message p encoding/decoding.
- Arithmetic operations (NTT, base-multiplication, …).
- Seed expansion with Keccak.
- Ciphertext comparison.

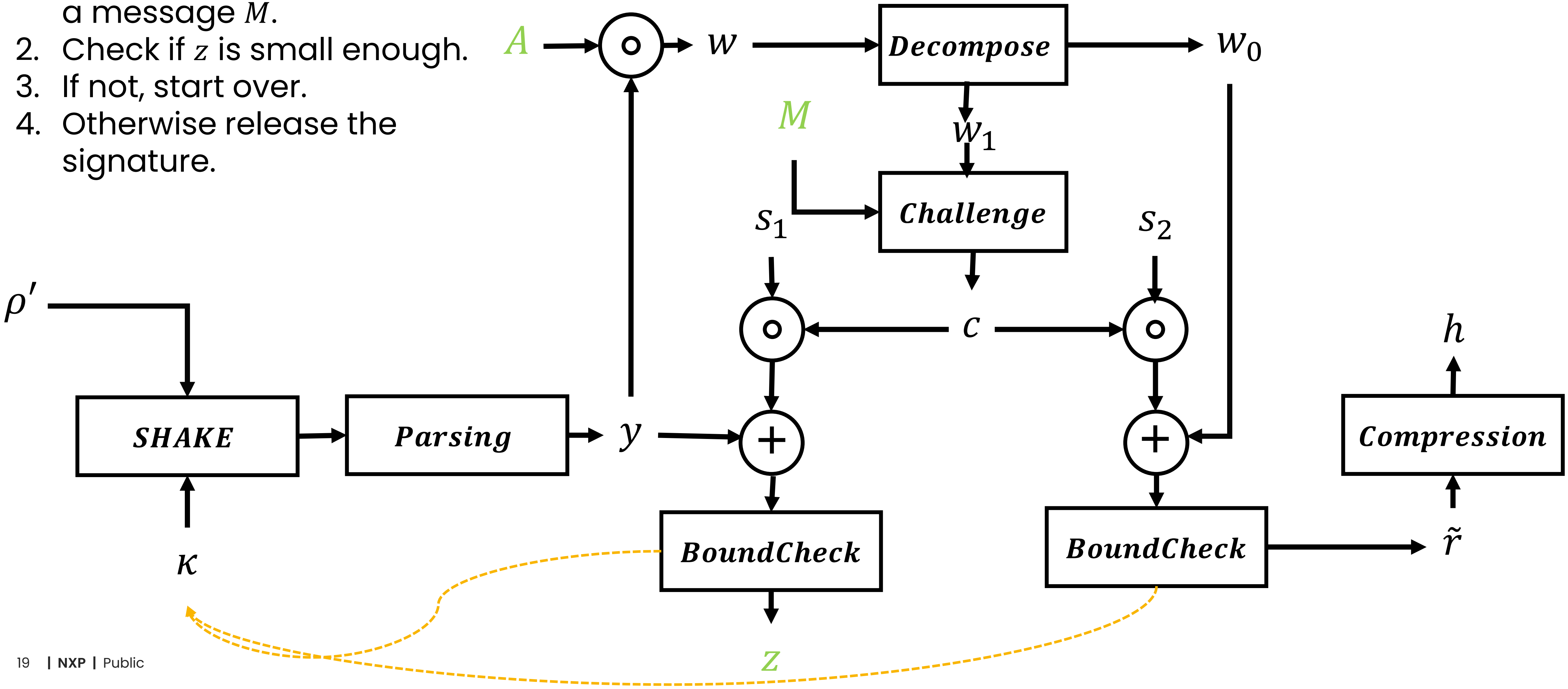
01

Side-channel attacks

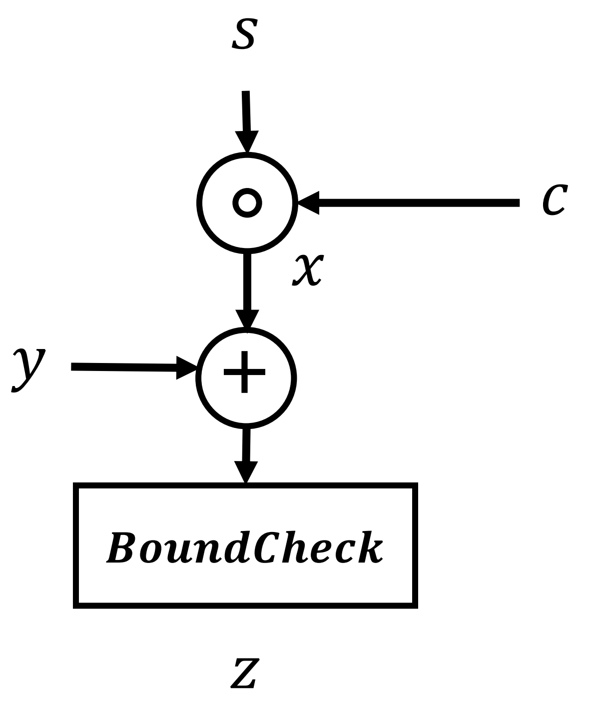
Dilithium

Dilithium Overview: Fiat-Shamir with Abort

1. Generate a signature z for a message M.
2. Check if z is small enough.
3. If not, start over.
4. Otherwise release the signature.



Dilithium: Attack vector $\mathbf{x} = \boldsymbol{s} \odot \boldsymbol{c}$



Observations:

- The signature is $z = y + s \odot c$ where we call $x = s \odot c$.
- Both s and c have a small norm hence x is small.
- The result polynomial x can be expressed as (first coeff.):

$$x_o = c_0 s_0 - \sum c_i s_{n-i}$$

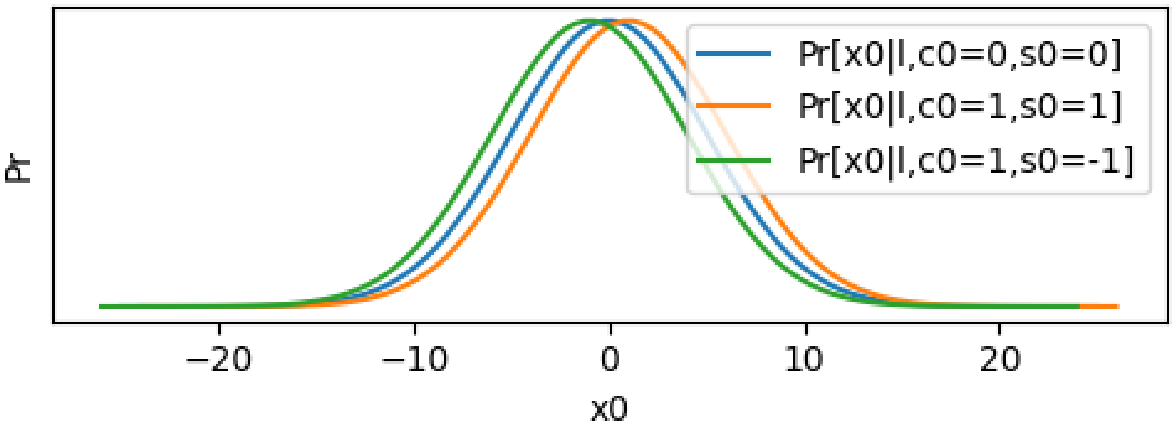
where no modular reduction occurs.

- Mean of distribution of x_0 depends on $c_0 s_0$.

$$x_0 \leftarrow N(c_0 s_0, \sigma^2)$$

Side-channel adversary:

- Filter the signatures to keep only $c_0 = 1$.
- Sample $\Pr[x_0 | l, c_0 = 1]$ through SCA.
- Compute the mean of that distribution.
- Recover the secret key coefficient s_0.

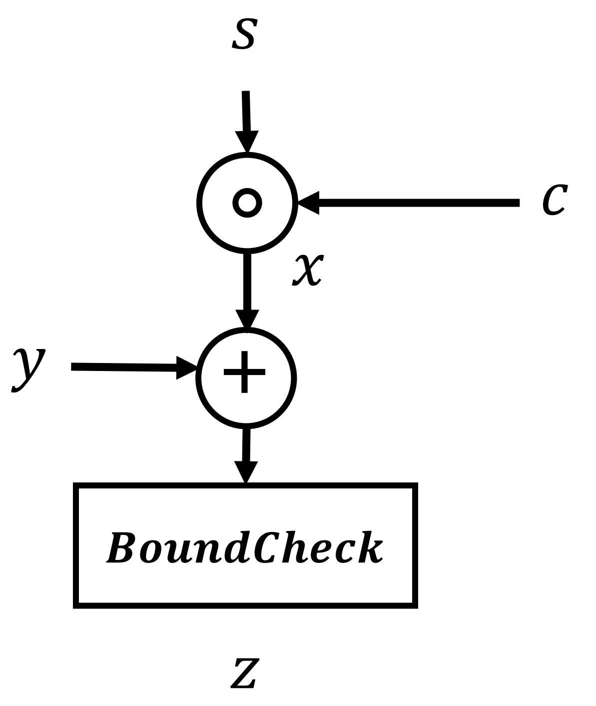


"Exploiting Small-Norm Polynomial Multiplication with Physical Attacks: Application to CRYSTALS-Dilithium", Bronchain et al, TCHES 2024.

Dilithium: Attack vector y

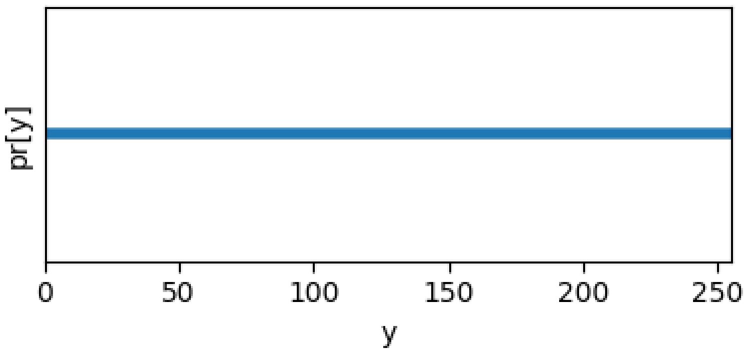
Observations:

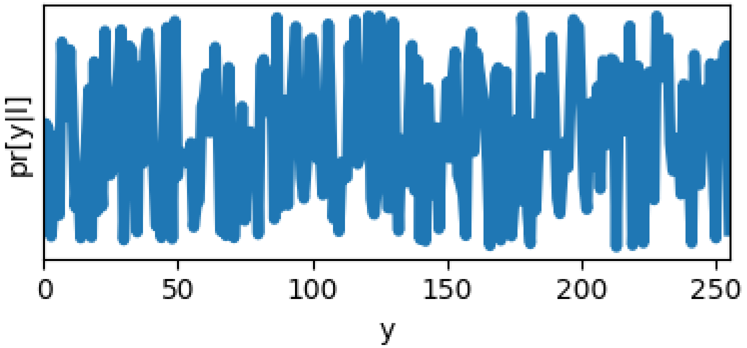
- The signature is $z = y + s \odot c$ where we call $x = s \odot c$.
- The released signature z is given to the adversary.
- y must be uniform to perfectly hide $x = s \odot c$.



Side-channel adversary:

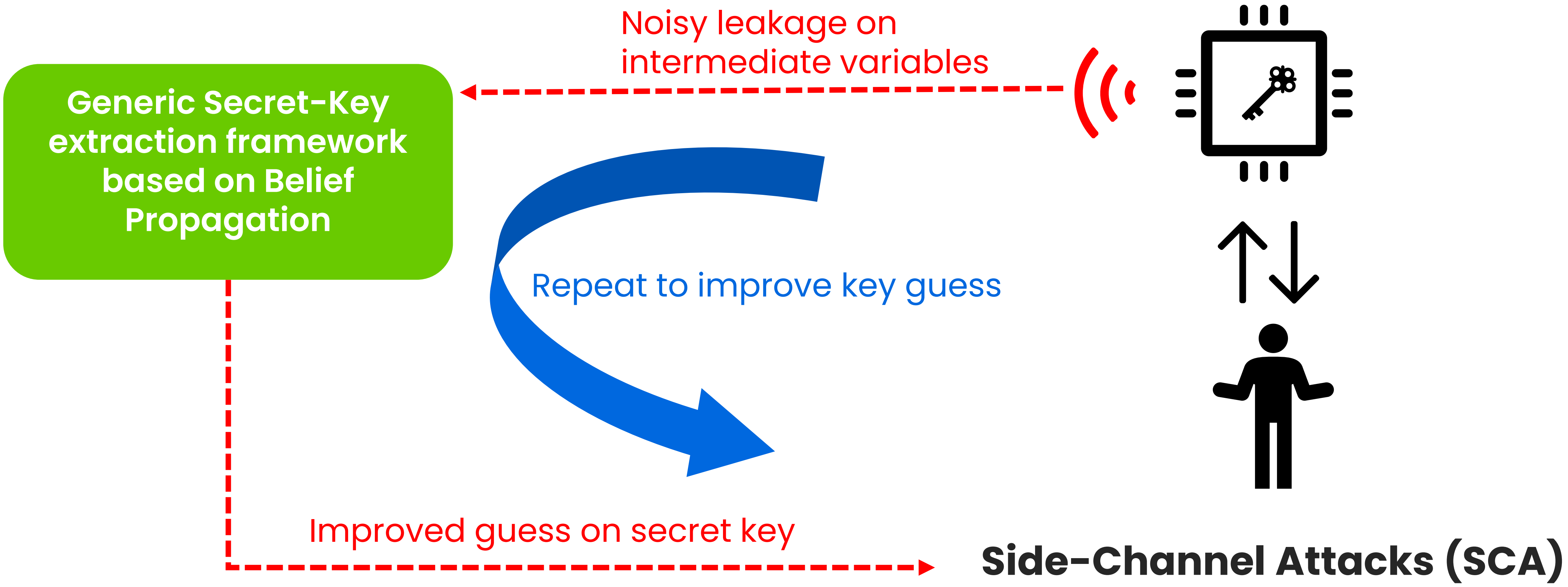
- Collect signatures (z, c)
- Record the corresponding leakages on y.
- Estimate the posterior distribution of $\Pr[y \mid l]$.
- z does not hide perfectly x anymore, recover the key.





"Exploiting Small-Norm Polynomial Multiplication with Physical Attacks: Application to CRYSTALS-Dilithium", Bronchain et al, TCHES 2024.

Generic framework



"Exploiting Small-Norm Polynomial Multiplication with Physical Attacks: Application to CRYSTALS-Dilithium", Bronchain et al, TCHES 2024.

Generic framework – Methodology

<u>Methodology:</u>

1. Collect N signature (c^i, z^i) and leakage on x_0^i and/or y_0^i.
2. Estimate distribution $\Pr\left[x_0^i \middle| l^i\right]$.
3. Build the linear system of equations.
4. Solve the system to recover distribution on s_j.

$$\underbrace{\begin{pmatrix} c_0^0 & \cdots & c_{n-1}^0 \\ \vdots & \ddots & \vdots \\ c_0^{N-1} & \cdots & c_{n-1}^{N-1} \end{pmatrix}}_{\text{Public polynomial (integers)}} \cdot \underbrace{\begin{pmatrix} s_0 \\ \vdots \\ s_{n-1} \end{pmatrix} = \begin{pmatrix} x_0^0 \\ \vdots \\ x_0^{N-1} \end{pmatrix}}_{\text{Secret polynomial (distributions)}}$$

<u>Used Solver:</u>

- Belief propagation based:
  - Iterative message passing algorithm
- Well studied in SCA context but heuristic.

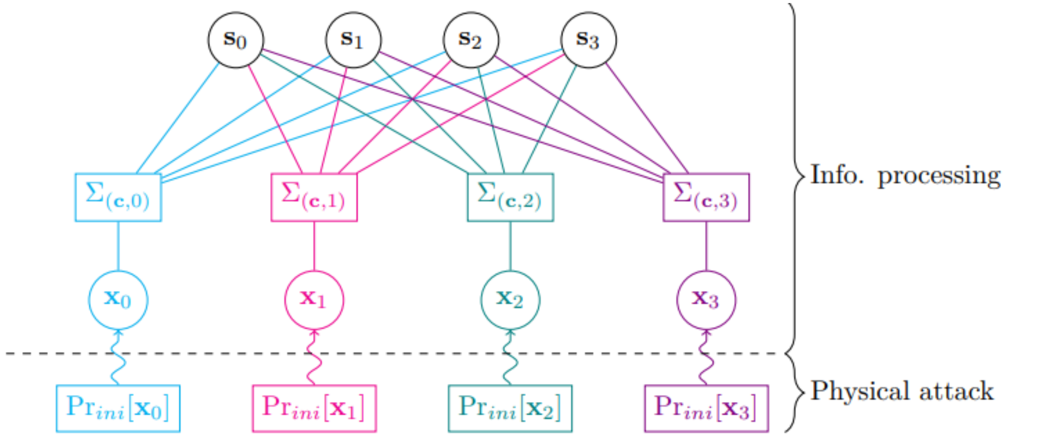
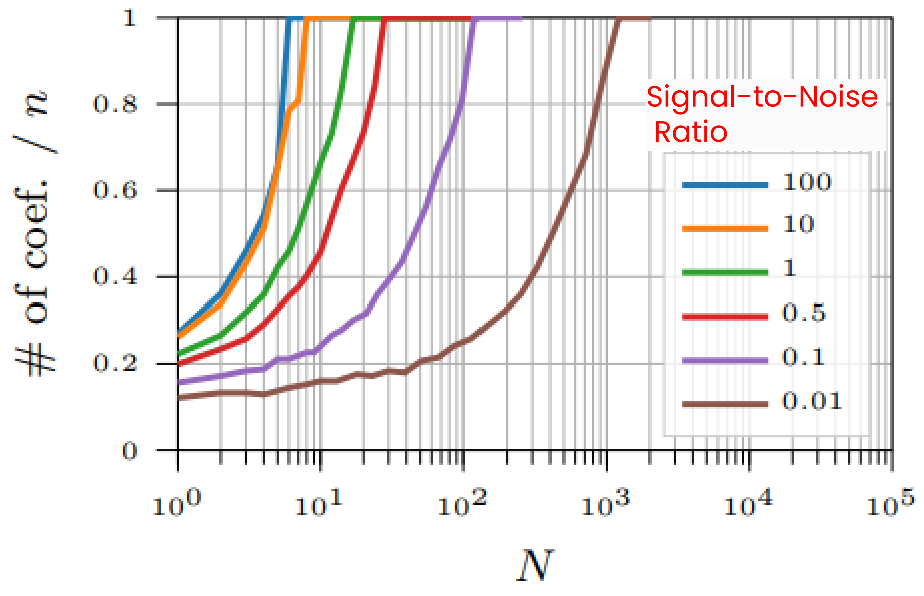


Figure 1: Example factor graph for parameters $N = 1$ trace and polynomial of degree $n = 4$.

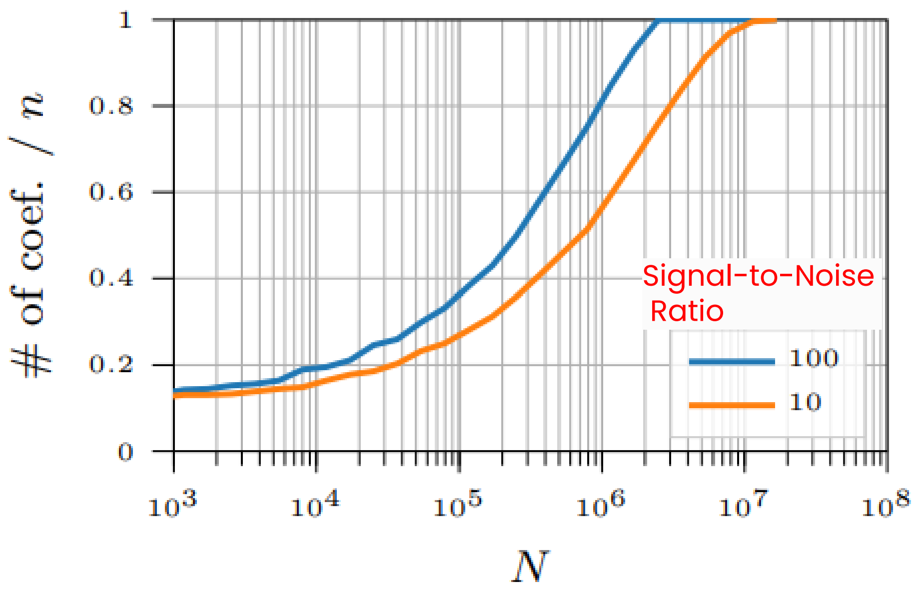
"Exploiting Small-Norm Polynomial Multiplication with Physical Attacks: Application to CRYSTALS-Dilithium", Bronchain et al, TCHES 2024.

Generic Framework – Key Recovery Efficiency

SCA on accepted signatures



SCA on rejected signatures (w early abort)



<u>Observation from simulated experiments:</u>

- When noise is low, <10 signatures are needed to recover the full key.
- Increasing the noise makes the number of traces needed increasing linearly.
- Rejected signature (with and without early-abort) are also exploitable but require much more traces.

"Exploiting Small-Norm Polynomial Multiplication with Physical Attacks: Application to CRYSTALS-Dilithium", Bronchain et al, TCHES 2024.

Exploiting rejected signatures

From the reference implementation of Dilithium:

[dilithium/ref/poly.c at master · pq-crystals/dilithium · GitHub]

```
/* It is ok to leak which coefficient violates the bound since
   the probability for each coefficient is independent of secret
   data but we must not leak the sign of the centralized representative. */
```

- The rejection probability is independent of the secret if…
- The sign of y is NOT leaked.

With SCA:

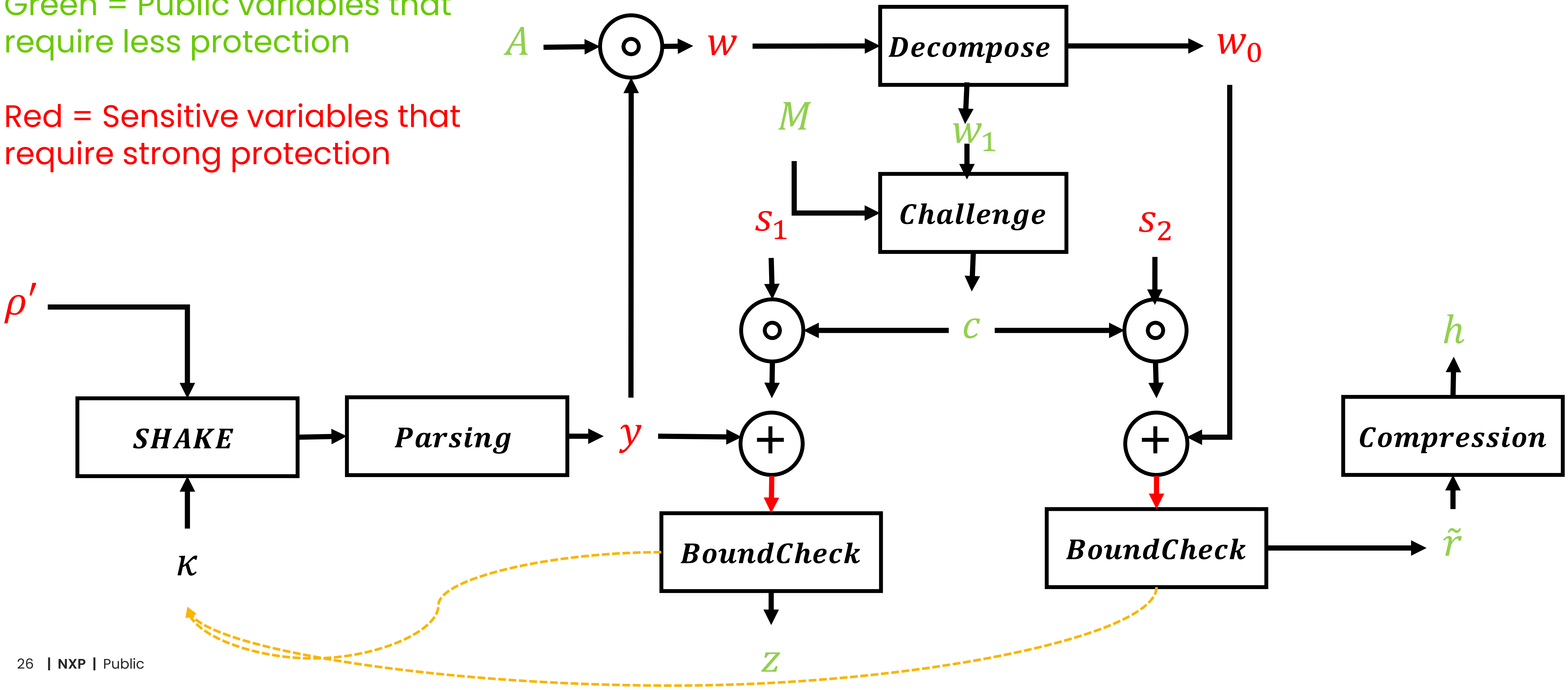
- Biased posterior distribution of y thanks to SCA.
- The rejection probability is dependent of the secret.
- Early rejection strategy leaks the exact rejected coefficient.
- Even without early rejection, the attack can be mounted (but less efficient).

"Exploiting Small-Norm Polynomial Multiplication with Physical Attacks: Application to CRYSTALS-Dilithium", Bronchain et al, TCHES 2024.

Sensitivity analysis of Dilithium Sign

Green = Public variables that require less protection

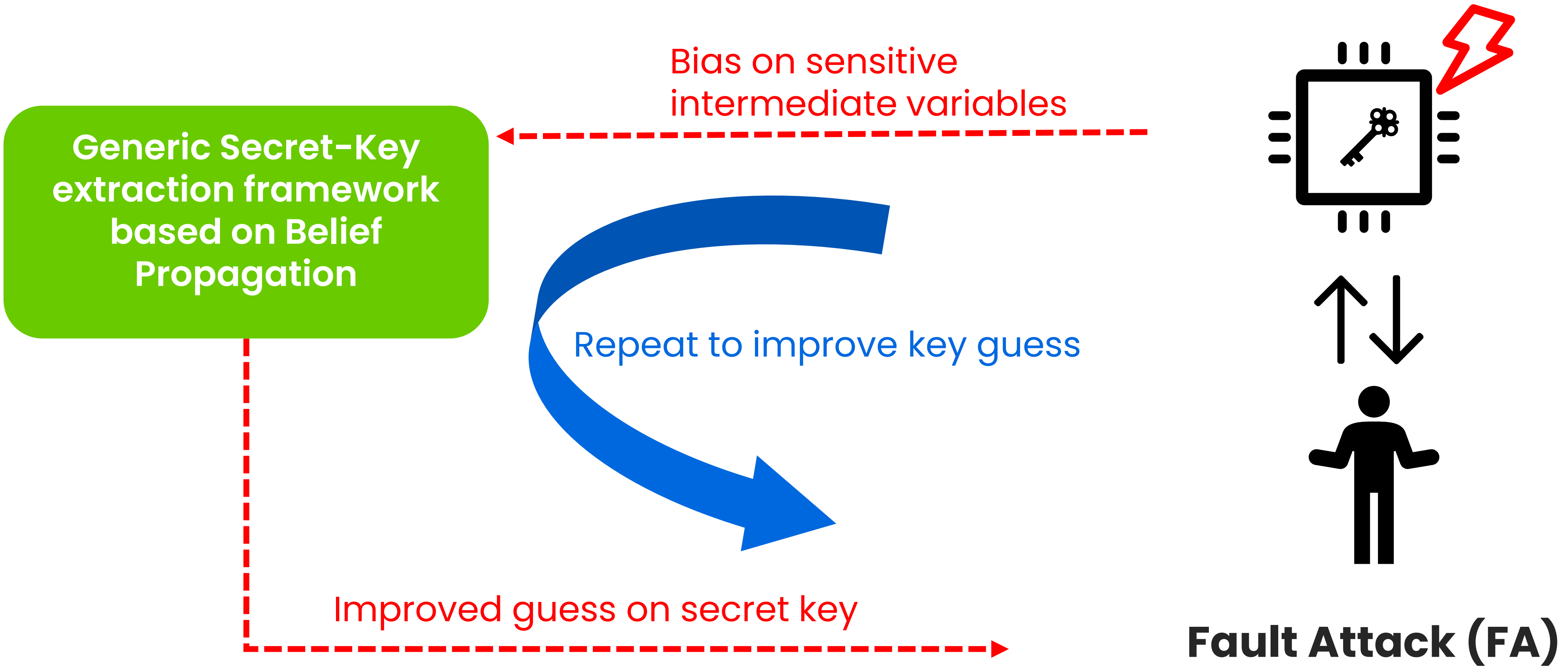
Red = Sensitive variables that require strong protection



Fault Attack

Dilithium

Generic Framework – Bias with fault attack



"Exploiting Small-Norm Polynomial Multiplication with Physical Attacks: Application to CRYSTALS-Dilithium", Bronchain et al, TCHES 2024.

From MLWE to RLWE with fault attack (overview)

<u>MLWE:</u>

- Manipulated variables are polynomials.
- Security depends on the module (k, ℓ) and size of polynomials.

$$\underbrace{\begin{bmatrix} A[0,0] & \cdots & A[0,\ell-1] \\ \vdots & \ddots & \vdots \\ A[k-1,0] & \cdots & A[k-1,\ell-1] \end{bmatrix}}_{\text{Public matrix } A} \cdot \underbrace{\begin{bmatrix} s_1[0] \\ \vdots \\ s_1[\ell-1] \end{bmatrix} + \begin{bmatrix} s_2[0] \\ \vdots \\ s_2[k-1] \end{bmatrix}}_{\text{Secret Keys}} = \underbrace{\begin{bmatrix} t\,[0] \\ \vdots \\ t[k-1] \end{bmatrix}}_{\text{Public Key}}$$

<u>Fault Injections:</u>

- Force the target to manipulate corrupted data.
- Observe the resulting faulted signatures.
- Use them to reduce the hardness of the problem.

Known by adversary

Target of the adversary

<u>RLWE:</u>

- Security depends on the size of polynomials.

$$a \cdot s_1[0] + s_2[0] = b$$

<u>Key Recovery:</u>

- Dilithium polynomials size is chosen to be used in MLWE, not RLWE.
- The hardness of the instance is decreased.
- Lattice solving tools can practically recover the secret keys.

From MLWE to RLWE with fault attack (1)

$$\left.\begin{aligned}
z[0] &= \mathrm{ExpandMask}(seed, \ell.\kappa + 0) + cs_1[0] \\
z[1] &= \mathrm{ExpandMask}(seed, \ell.\kappa + 1) + cs_1[1] \\
\\
z[0] &= y[0] + cs_1[0] \\
z[1] &= y[1] + cs_1[1]
\end{aligned}\right\} \text{Correct Execution}$$

$$\left.\begin{aligned}
z[0] &= \mathrm{ExpandMask}(seed, \ell.\kappa + 0) + cs_1[0] \\
z[1] &= \mathrm{ExpandMask}(seed, \ell.\kappa + \mathbf{0}) + cs_1[1] \\
\\
z[0] &= y[0] + cs_1[0] \\
z[1] &= y[0] + cs_1[1]
\end{aligned}\right\} \text{Faulted Execution}$$

From MLWE to RLWE with fault attack (2)

```
push      { r3, r4, r5, r6, r7, lr }
mov       r5,r0
mov       r6,r1
uxth      r4,r2
add.w     r7,r0,#0x1000
loop:
mov       r2,r4
mov       r0,r5
mov       r1,r6
add.w     r5,r5,#0x400
bl        pqcrystals_dilithium2_ref_poly_uniform_gamma1
adds      r4, #0x1
cmp       r5,r7
uxth      r4,r4
bne       loop
pop       { r3, r4, r5, r6, r7, pc }
```

Instruction skip

From MLWE to RLWE with fault attack (3)

Adv obtains signature (z, c)

First Signature

$$z[0] = y[0] + cs_1[0]$$
$$z[1] = y[0] + cs_1[1]$$

$$z[0] - z[1] = c(s_1[0] - s_1[1])$$

Second Signature

$$z'[1] = y'[1] + c's_1[1]$$
$$z'[2] = y'[1] + c's_1[2]$$

$$z'[1] - z'[2] = c'(s_1[1] - s_1[2])$$

Third Signature

$$z''[2] = y''[2] + c''s_1[2]$$
$$z''[3] = y''[2] + c''s_1[3]$$

$$z''[2] - z''[3] = c''(s_1[2] - s_1[3])$$

3 equations with 4 secret polynomials

Express all $s_1[i]$ as function of $s_1[0]$ and known data

Side-Channel countermeasures

Masking: introduction

Intuitive view:

- Masking splits a secret x into d "shares" $(x_0, \dots, x_{d-1})$.
- Every subset of d–1 shares is independent of the secret x.
- If the adversary must recombine the d shares to obtain the x.
- Under noisy knowledge of x_i, the information on x decreases exponentially with the number of shares.

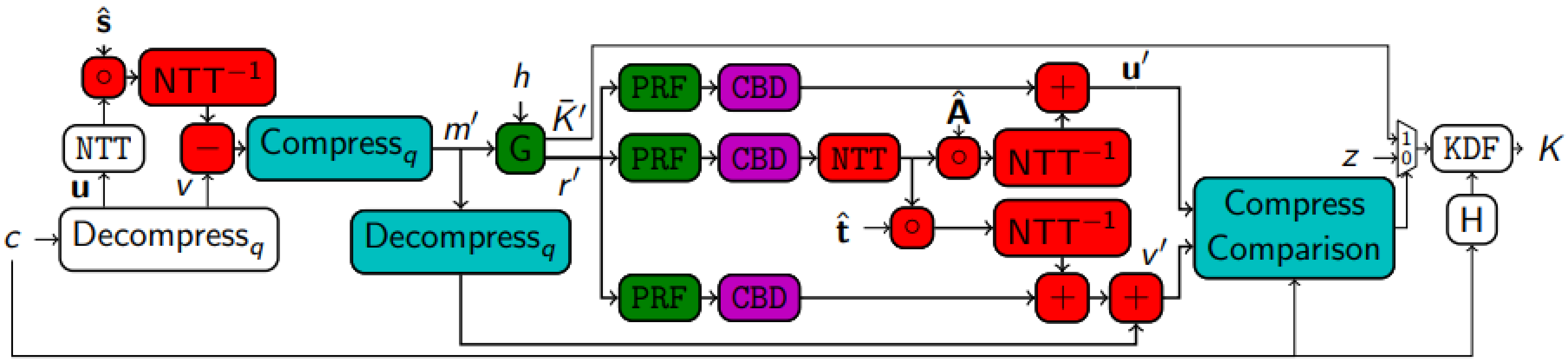
Boolean masking:

- The recombination of shares is done with XOR
$$x = x_0 \oplus x_1 \oplus \cdots \oplus x_{d-1}$$
- Very efficient to protect:
  - Boolean operations.
  - Symmetric key cryptography.
  - Keccak.

Arithmetic masking:

- The recombination of shares is done with modular additions
$$x = (x_0 + x_1 + \cdots + x_{d-1}) \bmod q$$
- Very efficient to protect:
  - Modular operations.
  - Polynomial operations.

Different type of operation and type of masking: Kyber



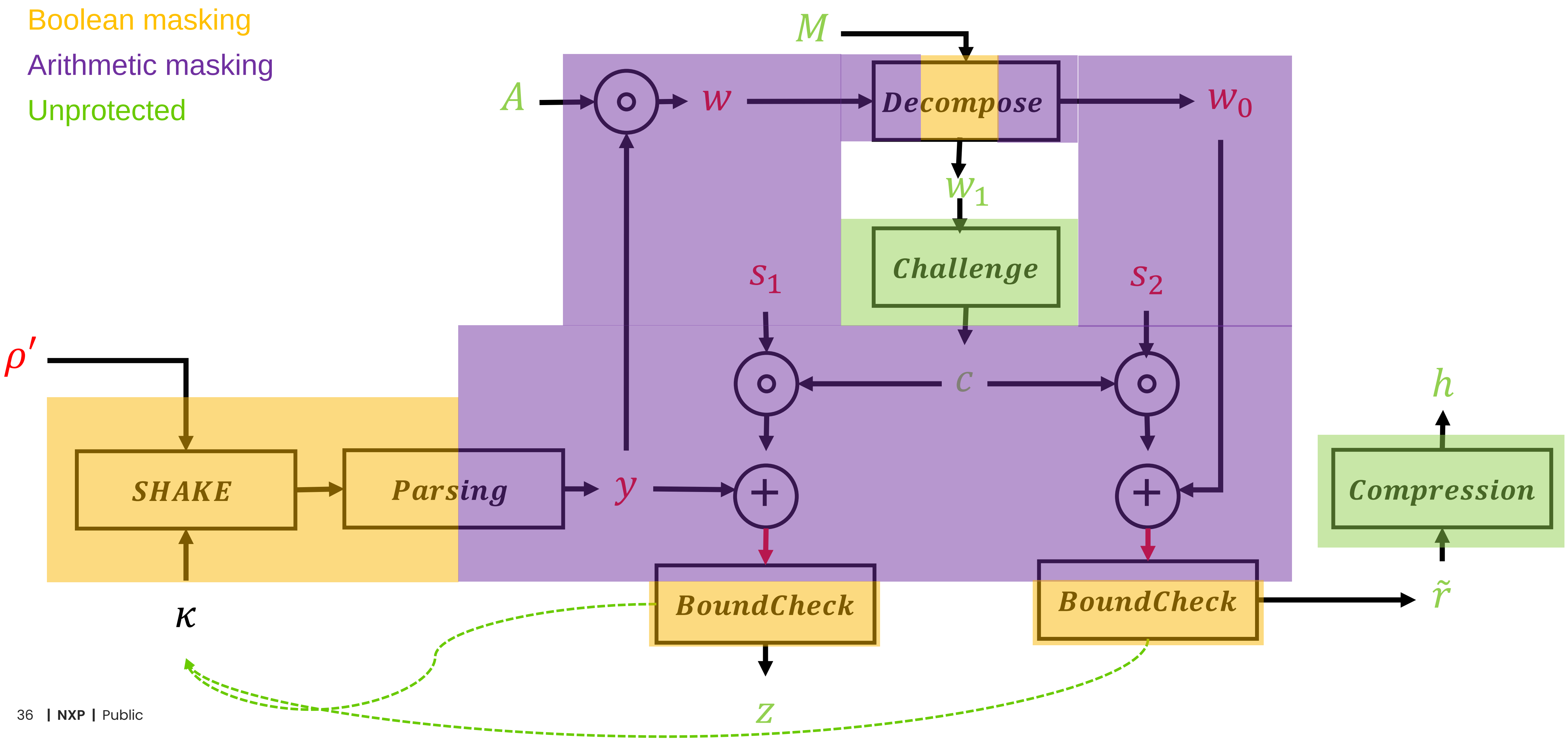
Poly. arithmetic (red):
- Arith. masking.
- Linear overheads.

Hash functions (green):
- Boolean masking.
- Quadratic overheads.

Poly. sampl. (purple) & compress. (cyan):
- Boolean & arith. masking.
- Quadratic overheads.

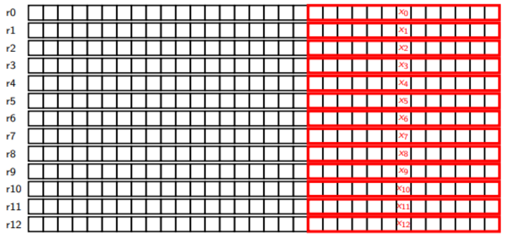
 ''Bitslicing Arithmetic/Boolean Masking Conversions for Fun and Profit with Application to Lattice-Based KEMs', Bronchain & Cassiers, TCHES 2022.

Different type of operation and type of masking: Dilithium



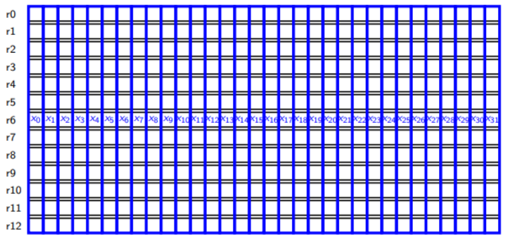
Speeding up software hardened PQC: bitslice and canonical representation

Canonical



- $+$ Arithmetic operations ($+$, $\times$).
- $-$ Single-bit processing.
- $-$ Memory/registers usage.

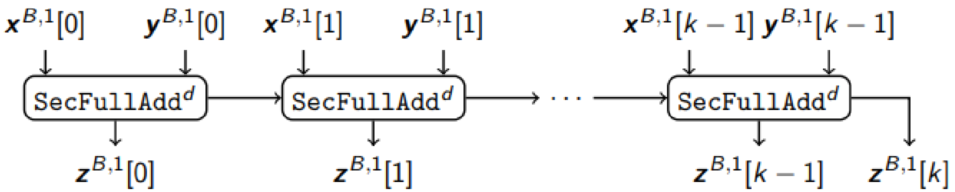
Bitslice



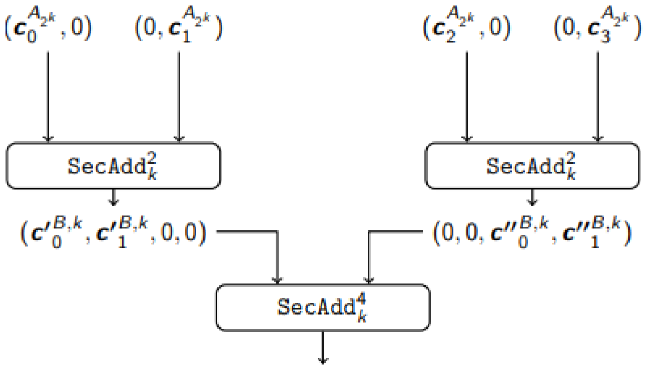
- $+$ Bitwise operations: throughput.
- $+$ Security: registers fully used.
- $-$ Representation change cost.

<u>Bitslicing enables:</u>

- Efficient protected FullAdd.
- Which enables efficient protected Add.
- Which enables efficient all other masking gadgets.



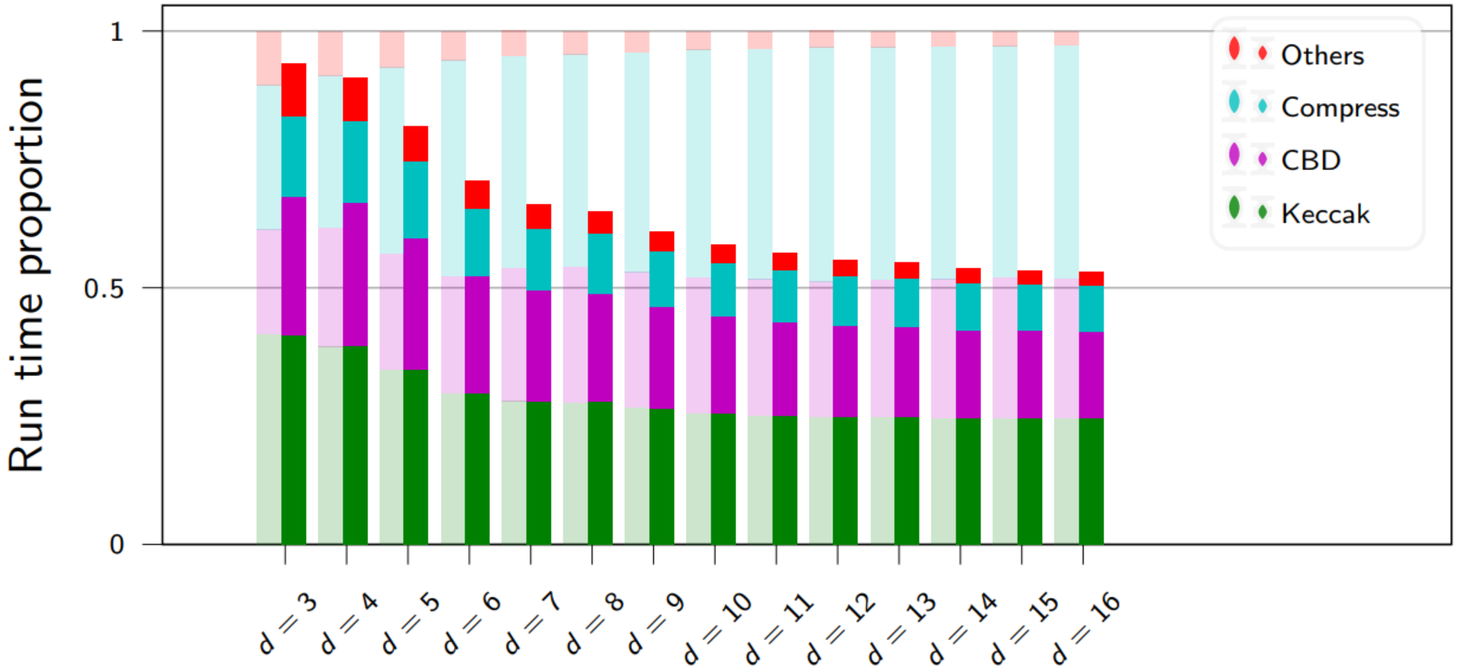
Secure Addition



Secure Arithmetic to Boolean conversion

’’Bitslicing Arithmetic/Boolean Masking Conversions for Fun and Profit with Application to Lattice-Based KEMs’, Bronchain & Cassiers, TCHES 2022.

Improving the conversion gadgets (Kyber768 on a Cortex-M4)



<u>Remarks on performance split-up:</u>

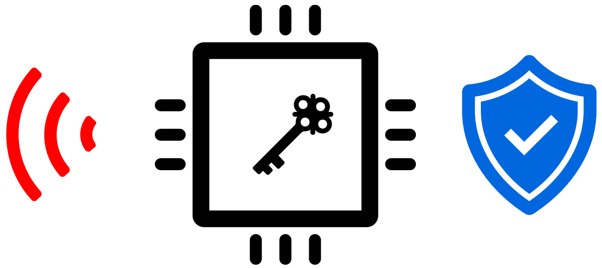
- Improving the core masking gadget gave a large speed up on the over all scheme.
- Secure SHA3 is the bottleneck of the scheme.

 ''Bitslicing Arithmetic/Boolean Masking Conversions for Fun and Profit with Application to Lattice-Based KEMs', Bronchain & Cassiers, TCHES 2022.

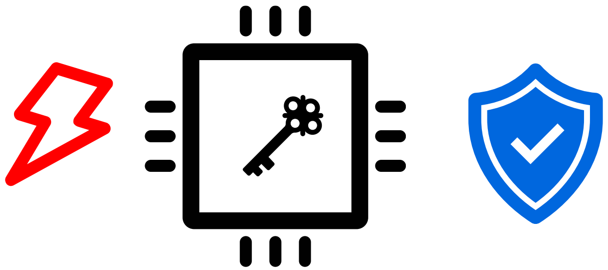
Conclusions

Hardening for SCA and FA

Side-channel countermeasures:
- SHA-3/SHAKE requires Boolean masking.
- Polynomial arithmetic requires Arithmetic masking.
- Both are well understood, but conversions between them are costly.



Fault-attack countermeasures:
- Control-flow integrity.
- Re-computation of critical operations.



All these countermeasures have significant impact on run-time and memory consumption.

"Protecting Dilithium Against Leakage Revisited Sensitivity Analysis and Improved Implementations", Azouaoui et al., TCHES 2023.

Conclusions

- Migration to PQC is a difficult & hot topic, particularly in embedded environments
- Specific attacks:
  - Large attack surface.
  - Still very active area of research.
- Many other practical challenges
  - Memory consumption on (very limited devices).
  - Available hardware (co-processors).
  - Efficient side-channel and fault countermeasures.

Get in touch

Olivier, Bronchain
olivier.bronchain@nxp.com

nxp.com

nxp.com