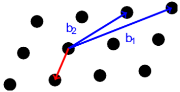
Post-Quantum Algorithms and Side-Channel Countermeasures

Jean-Sébastien Coron

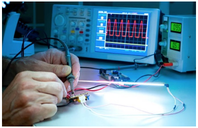
University of Luxembourg

Overview

- Post-quantum algorithms:
  - Overview of algorithms believed to be secure against quantum adversaries.
  - The Kyber and Dilithium lattice-based algorithms.



- Side-channel attacks
  - The threat of side-channel attacks
  - Relevance for Kyber and Dilithium



- Side-channel countermeasures
  - Security model for high-order security
  - Conversion between Boolean and arithmetic masking
  - Application to Kyber and Dilithium

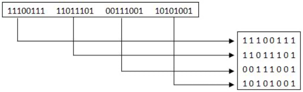
Why post-quantum cryptography

- Public-key cryptography is based on hard problems
  - RSA: hardness of factoring $N = pq$
  - ECC: hardness of finding d in $P = d.G$
  - We don't know any classical algorithm that can efficiently solve these problems.
  - but these problems are broken by a quantum computer

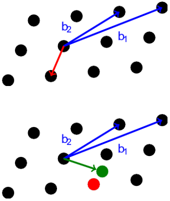
- Post-quantum hardness
  - In the quantum era, a problem should remain hard even when attacked by both classical and quantum computers.
  - Fortunately, we know many such problems !

Families of post-quantum schemes

- Code-based Cryptography (McEliece, 1978)
  - Relies on the hardness of decoding a general linear code
  - McEliece's encryption scheme.
  - Large key size

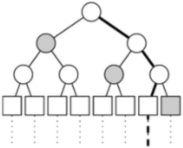


- Lattice-based cryptography
  - Based on the difficulty of certain problems in lattices (SVP and CVP)
  - NTRU (1996), a very fast public-key encryption scheme.



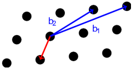
Families of post-quantum schemes (2)

- Multivariate cryptography
  - Matsumoto-Imai C^* scheme (1988), HFE [P96]
  - Security relies on the difficulty of solving systems of multivariate polynomial equations.
  - Short signatures
- Hash-based cryptography (Lamport, 1979)
  - Based on the security of cryptographic hash functions.
  - Mostly used for digital signatures.

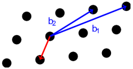


NIST's PQC standardization

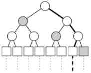
- The NIST competition: encryption/KEM and signatures
  - Round 1 (2017): 82 submissions. Round 2: 26 2nd round candidates. Round 3: 7 finalists and 8 alternates.
- 3rd round KEM selection (July 2022):
  - Kyber (draft standard August 2023)



- 3rd round Signature selection (July 2022)
  - Dilithium (draft standard August 2023)
  - Falcon



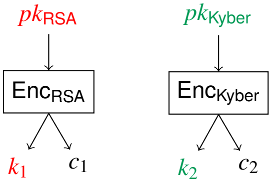
  - SPHINCS+



Transition to post-quantum algorithms

- Hybrid cryptography
  - Use of two (or more) cryptographic systems together.
  - During the transition to PQC, use both classical and PQC algorithms for robust security.
  - Recommended by ANSSI and BSI for transition phase

- Key establishment (KEM)
  - Derive a session key dependent on both classical and PQC algorithms.

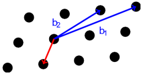


$$k = \mathsf{H}(\mathsf{H}(k_1, k_2), c_1, c_2)$$

- Signature
  - Concatenate the classical and PQC signatures
  - Signature is verified if *both* signatures are verified.

Kyber: post-quantum KEM

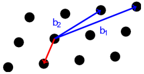
- Kyber
  - Key encapsulation mechanism (KEM) based on lattice-based cryptography.
  - Goal: provide security level equivalent or better than RSA, ECC, but resistant to quantum attacks.
  - Security based on the hardness of certain problems in ideal lattices: Module Learning With Errors (MLWE) problem.



- Performance:
  - Designed to have relatively small key sizes and ciphertexts, and to be computationally efficient.

Dilithium and Falcon signature schemes

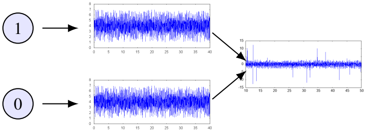
- Dilithium and Falcon
  - Both based on lattice-based cryptography, for security against quantum attacks
  - Dilithum: Fiat-Shamir with abort on module lattices.



  - Falcon: hash-and-sign on NTRU lattices
  - Both designed to be efficient with relatively small key and signature sizes
- Comparison
  - Falcon provides smaller key and signature sizes than Dilithium.
  - But Falcon requires Gaussian sampling, which is hard to secure against side-channel attacks.
  - Dilithium recommended by NIST to be the primary signature algorithm.

The challenge of side-channel attacks

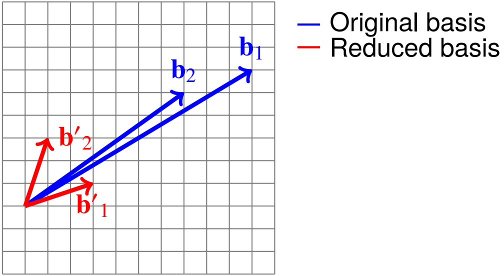
- Side-channel attacks
  - Timing attacks, power analysis attacks, electromagnetic attacks...



- PQC and side-channel attacks:
  - PQC algorithms, like all cryptographic systems, are susceptible to side-channel attacks.
  - Side-channel resistance less well understood and more challenging.

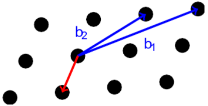
Hard lattice problems in cryptography

- Lattice
  - Regular grid of points in multidimensional space, defined by a basis of vectors.
- Shortest Vector Problem (SVP)
  - Given a lattice basis, find the shortest non-zero vector.
  - Believed to be hard even for quantum computers.
  - LLL algorithm provides an approximation in polynomial-time.



Hard lattice problems in cryptography

- Learning With Errors (LWE):
  - Given $\vec{A} \in \mathbb{Z}_q^{\ell \times n}$ such that $\vec{A} \cdot \vec{s} = \vec{e}$ for small $\vec{e}$, recover $\vec{s}$.
- Ring-LWE and Module-LWE:
  - Variant of LWE where the secret and errors come from a polynomial ring.
  - Offers efficiency advantages.
- Significance:
  - Reduction from worst-case lattice problems.
  - Believed to be hard against both classical and quantum adversaries.

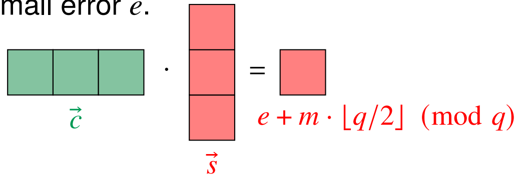


LWE-based encryption [R05]

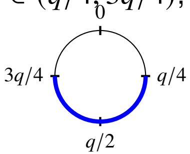
- Key generation
  - Secret-key: $\vec{s} \in (\mathbb{Z}_q)^n$
- Encryption of $m \in \{0, 1\}$
  - A vector $\vec{c} \in \mathbb{F}_q$ such that

$$\langle \vec{c}, \vec{s} \rangle = e + m \cdot \lfloor q/2 \rfloor \pmod q$$

  for a small error e.



- Decryption
  - Compute $m = \text{th}(\langle \vec{c}, \vec{s} \rangle \bmod q)$
  - where $\text{th}(x) = 1$ if $x \in (q/4, 3q/4)$, and 0 otherwise.

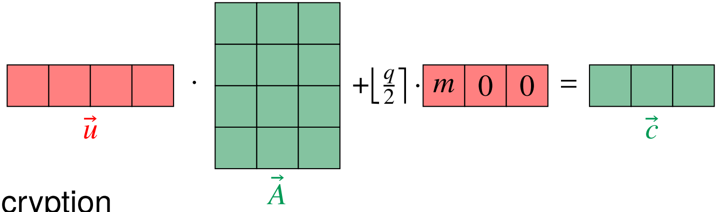


LWE-based public-key encryption

- Key generation
  - Secret-key: $\vec{s} \in (\mathbb{Z}_q)^n$, with $s_1 = 1$.
  - Public-key: $\vec{A}$ such that $\vec{A} \cdot \vec{s} = \vec{e}$ for small $\vec{e}$
    - Every row of $\vec{A}$ is an LWE encryption of 0.
- Encryption of $m \in \{0, 1\}$

$$\vec{c} = \vec{u} \cdot \vec{A} + (m \cdot \lfloor q/2 \rceil, 0, \ldots, 0)$$

  - for a small $\vec{u}$



- Decryption
  - Compute $m = \text{th}(\langle \vec{c}, \vec{s} \rangle \bmod q)$

RLWE-based schemes

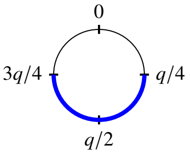
- RLWE-based scheme
  - We replace $\mathbb{Z}_q$ by the polynomial ring $R_q = \mathbb{Z}_q[x]/< x^\ell + 1 >$, where ℓ is a power of 2.
  - Addition and multiplication of polynomials are performed modulo $x^\ell + 1$ and prime q.
  - We can take $m \in R_2 = \mathbb{Z}_2[x]/<x^\ell + 1>$ instead of $\{0, 1\}$: more bandwidth.
- Ring Learning with Error (RLWE) assumption
  - $t = a \cdot s + e$ for small s, $e \leftarrow R$
  - Given t, a, it is difficult to recover s.

RLWE-based public-key encryption

- Key generation
  - $t = a \cdot s + e$ for random $a \leftarrow R_q$ and small $s, e \leftarrow R$.
- Public-key encryption of $m \in R_2$
  - $c = (a \cdot r + e_1,\ t \cdot r + e_2 + \lfloor q/2 \rceil m)$, for small e_1, e_2 and r.
- Decryption of $c = (u, v)$
  - Compute $m = \text{th}(v - s \cdot u)$

$$\begin{aligned} v - s \cdot u &= t \cdot r + e_2 + \lfloor q/2 \rceil m - s \cdot (a \cdot r + e_1) \\ &= (t - a \cdot s) \cdot r + e_2 + \lfloor q/2 \rceil m - s \cdot e_1 \\ &= \lfloor q/2 \rceil m + \underbrace{e \cdot r + e_2 - s \cdot e_1}_{\text{small}} \end{aligned}$$

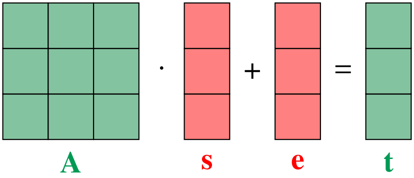
  - $m \in R_2 = \mathbb{Z}_2[x]/{<}x^\ell + 1{>}$: more bandwidth.



The Kyber scheme

- Polynomial operations:
  - Polynomial ring $\mathbb{Z}_q[X]/(X^{256}+1)$ with prime $q = 3329 = 2^{12} - 2^9 - 2^8 + 1$.
  - Use k-vectors and $k \times k$-matrices of ring elements
  - Kyber512: $k = 2$, Kyber768: $k = 3$, Kyber1024: $k = 4$

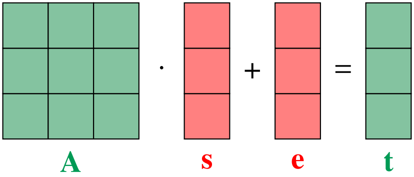
- Module Learning with Error (M-LWE)
  - $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, for a small error $\mathbf{e}$
  - Given $\mathbf{t}$, $\mathbf{A}$, difficult to recover $\mathbf{s}$.



Kyber key generation

Key generation

- Generate matrix $\mathbf{A}$ and small vectors $\mathbf{s}$ and $\mathbf{e}$
- Compute $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
  - Secret key : $\mathbf{s}$
  - Public key : $(\mathbf{A}, \mathbf{t})$



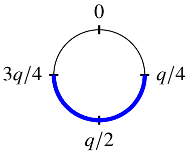
Kyber basic encryption

Encryption of a binary message $m \in \mathbb{Z}[X]/(X^{256}+1)$

- Generate small vectors $\mathbf{r}$ and $\mathbf{e_1}$, and small polynomial e_2
- Compute $\mathbf{u} = \mathbf{A}^T \cdot \mathbf{r} + \mathbf{e_1}$
- Compute $v := \mathbf{t}^T \cdot \mathbf{r} + e_2 + \lfloor q/2 \rceil m$
  - Ciphertext : $(\mathbf{u}, v)$

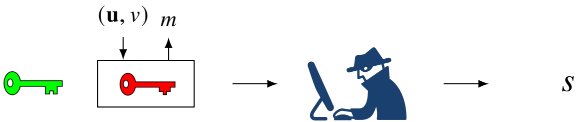
Decryption

- Compute $m = \mathrm{th}(v - \mathbf{s}^T \cdot \mathbf{u})$
  - where $\mathrm{th}(x) = 1$ if $x \in (q/4, 3q/4)$, and 0 otherwise.



From CPA to CCA security: FO transform

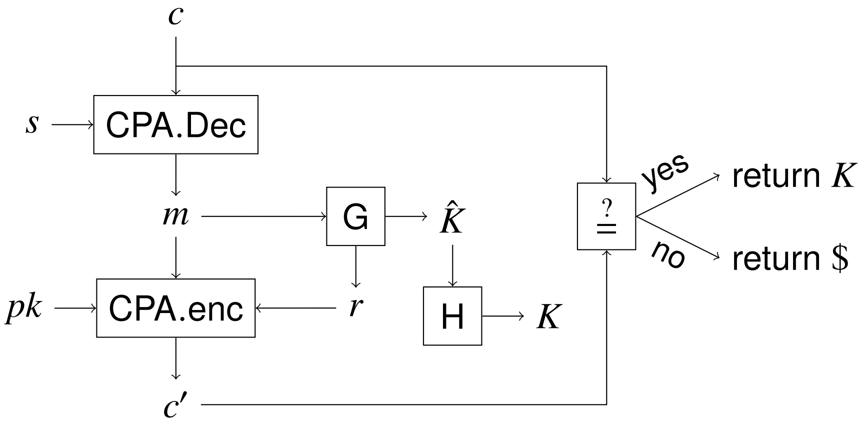
- Chosen Plaintext Attack (CPA) security
  - The previous scheme is CPA secure: given only the public-key, a ciphertext reveals no information on the plaintext.
- Chosen Ciphertext Attack (CCA) insecurity
  - The previous scheme is not CCA secure
  - By submitting a series of ciphertexts $(\mathbf{u}, v)$ and getting the corresponding m, one can recover the secret s.



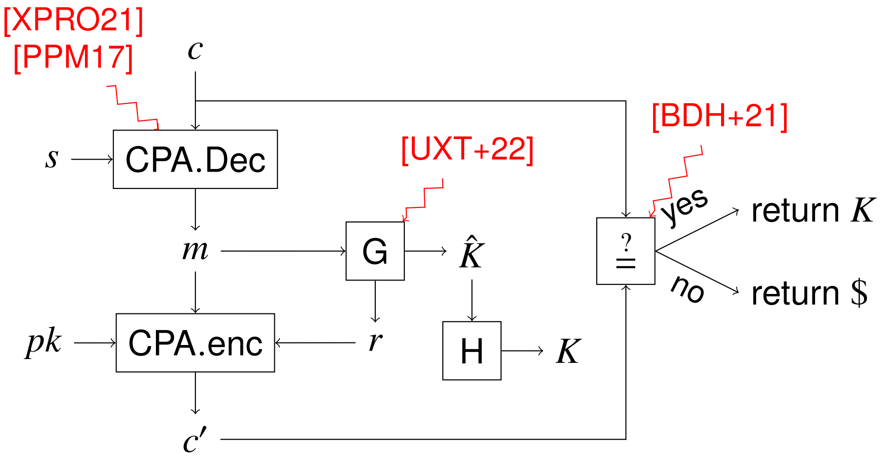
- CCA-security: Fujisaki-Okamoto transform
  - Generic transformation from CPA to CCA-security.
  - Verify correct decryption by re-encrypting the message and comparing the ciphertexts.

IND-CCA decryption with the FO transform

- Given the ciphertext c, the secret-key s and the public-key pk, recover the session key K

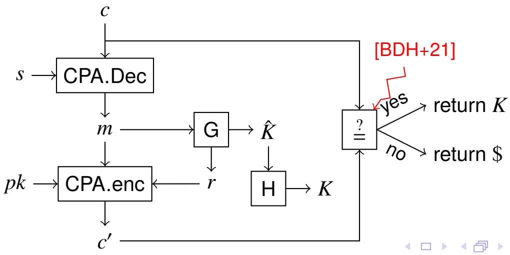


Side-channel attacks on the FO transform



Side channel attack on the FO transform [BDH+21]

- Without the FO transform, decryption failure attack
  - We submit perturbed ciphertexts $\vec{c'} = \vec{c} + \delta$
  - We obtain a plaintext m' and compare to m: plaintext checking oracle.
  - We can recover $\langle \vec{c}, \vec{s} \rangle$ by binary search, and eventually $\vec{s}$
- With FO, template attack against ciphertext comparison
  - The SCA gives us a plaintext checking oracle.
  - Use the above attack and recover the key

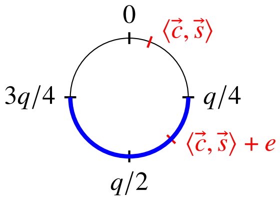


Decryption failure attack against CPA scheme

- Decryption failure attack without the FO transform
  - Assume that an attacker can submit a LWE ciphertext $\vec{c}$ and obtain $m = \text{th}(\langle \vec{c}, \vec{s} \rangle \bmod q)$
  - Submit a perturbed ciphertext with a small error

$$\vec{c'} = \vec{c} + (e, 0, \ldots, 0)$$

  - Get $m' = \text{th}(\langle \vec{c'}, \vec{s} \rangle) = \text{th}(\langle \vec{c}, \vec{s} \rangle + e)$ (with $s_1 = 1$).



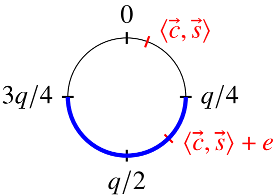
- Key recovery
  - Recover the value of $\langle \vec{c}, \vec{s} \rangle \bmod q$ by binary search.
  - With n such equations, recover $\vec{s}$

Decryption failure attack against CPA scheme

- Decryption failure attack without the FO transform
  - Assume that an attacker can submit a LWE ciphertext $\vec{c}$ and obtain $m = \text{th}(\langle \vec{c}, \vec{s} \rangle \bmod q)$
  - Submit a perturbed ciphertext with a small error

$$\vec{c'} = \vec{c} + (e, 0, \ldots, 0)$$

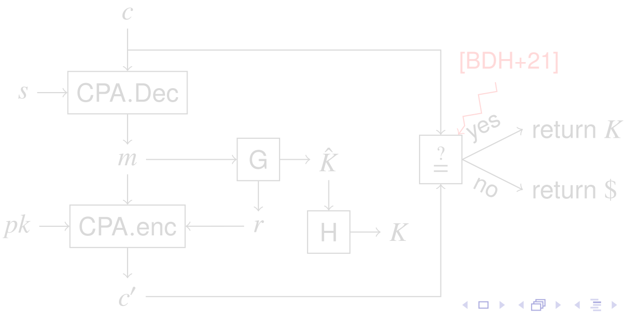
  - Get $m' = \text{th}(\langle \vec{c'}, \vec{s} \rangle) = \text{th}(\langle \vec{c}, \vec{s} \rangle + e)$ (with $s_1 = 1$).



- Key recovery
  - Recover the value of $\langle \vec{c}, \vec{s} \rangle \bmod q$ by binary search.
  - With n such equations, recover $\vec{s}$

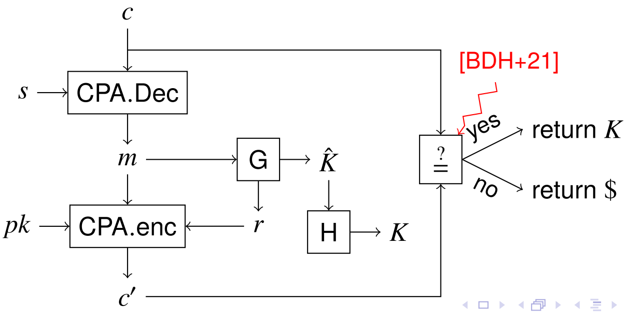
SCA on FO transform

- The previous attack is prevented by the FO transform
  - But one can perform a SCA on the FO transform !
- Decryption failure attack on ciphertext comparison
  - Submit a perturbed ciphertext $\tilde{c} = c + \delta$
  - If still decrypts to the same m, then re-encrypted c' and $\tilde{c}$ will differ on a single coefficient
  - otherwise they will differ on *all* coefficients
  - can be distinguished by SCA [BDH+21]



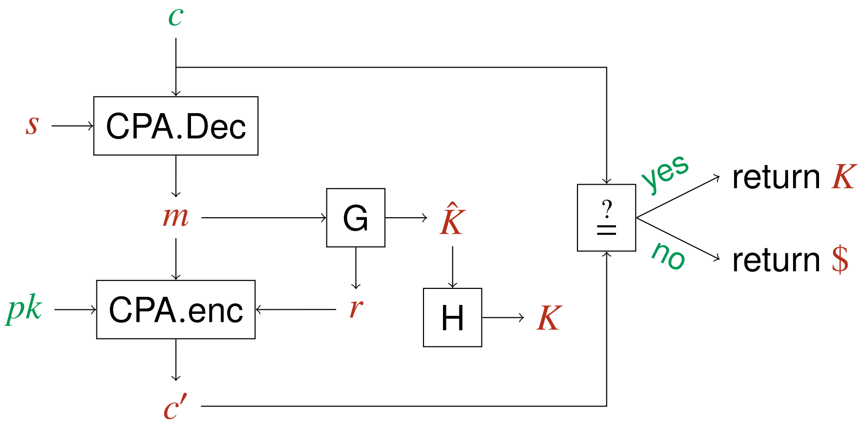
SCA on FO transform

- The previous attack is prevented by the FO transform
  - But one can perform a SCA on the FO transform !
- Decryption failure attack on ciphertext comparison
  - Submit a perturbed cipherext $\tilde{c} = c + \delta$
  - If still decrypts to the same m, then re-encrypted c' and $\tilde{c}$ will differ on a single coefficient
  - otherwise they will differ on *all* coefficients
  - can be distinguished by SCA [BDH+21]



Full masking of IND-CCA decryption

- The variables s, m, c', $\hat{K}$ and K must be masked.
  - The operations CPA.Dec, CPA.enc, G, H and $\stackrel{?}{=}$ must also be masked.



Higher-Order Masking

Basic principle

Each sensitive variable x is shared into n variables:

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

- Generate $n-1$ random variables $x_1, x_2, \ldots, x_{n-1}$
- Initially let $x_n = x \oplus x_1 \oplus x_2 \oplus \cdots \oplus x_{n-1}$

Security against DPA attack of order $n-1$

- Any subset of $n-1$ shares is uniformly and independently distributed

$\Rightarrow$ If we probe at most $n-1$ shares x_i, we learn nothing about x

Higher-Order Masking

Basic principle

Each sensitive variable x is shared into n variables:

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

- Generate $n-1$ random variables $x_1, x_2, \ldots, x_{n-1}$
- Initially let $x_n = x \oplus x_1 \oplus x_2 \oplus \cdots \oplus x_{n-1}$

Security against DPA attack of order $n-1$

- Any subset of $n-1$ shares is uniformly and independently distributed

  $\Rightarrow$ If we probe at most $n-1$ shares x_i, we learn nothing about x

High-order masking of Boolean circuits

Ishai-Sahai-Wagner private circuit [ISW03]

- The adversary can probe any subset of at most t wires
- Algorithm to transform any Boolean circuit C of size $|C|$ into a circuit of size $O(|C| \cdot t^2)$ that is perfectly secure against such an adversary.

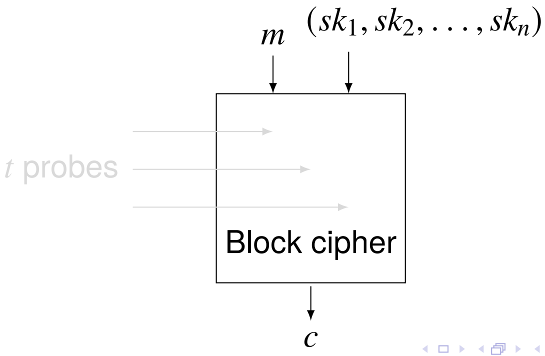
- Any Boolean circuit can be written with only Xor gates $c = a \oplus b$ and And gates $c = a \times b$.
  - High-order masking of $c = a \oplus b$: easy since linear.
  - High-order masking of $c = a \times b$: ISW multiplication gadget.
- Generic ISW transform
  - Not very adequate for lattice-based algorithms combining arithmetic and Boolean operations.

ISW security model

- The t-probing model
  - Protected block-cipher takes as input $n = 2t + 1$ shares sk_i of the secret key sk, with

$$sk = sk_1 \oplus \cdots \oplus sk_n$$

  - Prove that even if the attacker probes t variables in the block-cipher, he learns nothing about the secret-key sk.

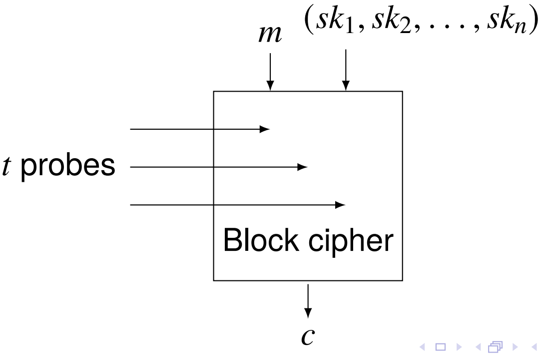


ISW security model

- The t-probing model
  - Protected block-cipher takes as input $n = 2t + 1$ shares sk_i of the secret key sk, with

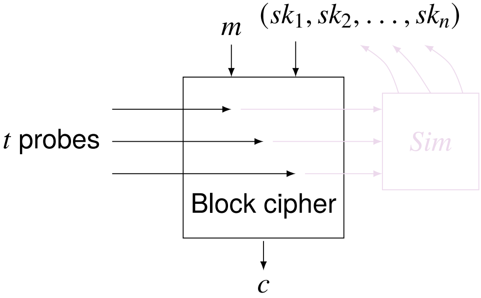
$$sk = sk_1 \oplus \cdots \oplus sk_n$$

  - Prove that even if the attacker probes t variables in the block-cipher, he learns nothing about the secret-key sk.



ISW security model

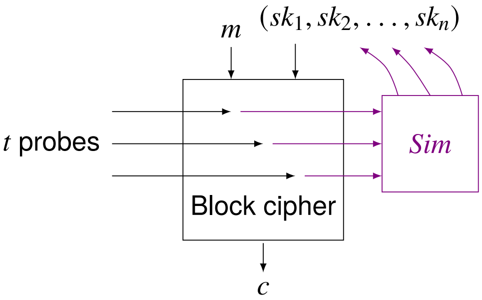
- Simulation framework of [ISW03]:



- Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
- Those $n-1$ shares sk_i are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the t probes, since he could simulate those t probes by himself.

ISW security model

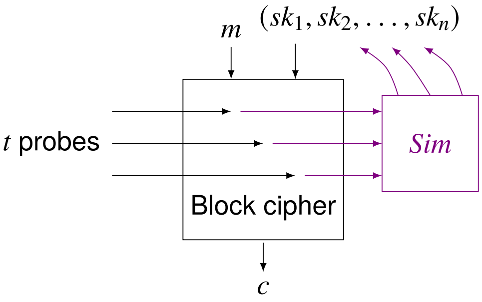
- Simulation framework of [ISW03]:



- Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
- Those $n-1$ shares sk_i are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the t probes, since he could simulate those t probes by himself.

ISW security model

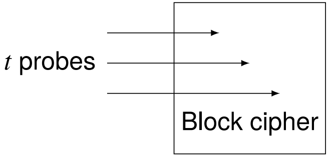
- Simulation framework of [ISW03]:



  - Show that any t probes can be perfectly simulated from at most $n - 1$ of the sk_i's.
  - Those $n - 1$ shares sk_i are initially uniformly and independently distributed.
  - $\Rightarrow$ the adversary learns nothing from the t probes, since he could simulate those t probes by himself.

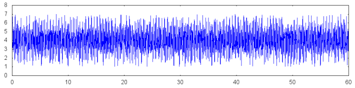
Probing Model vs. Reality

- Probing model
  - The attacker can choose at most t variables
  - He learns the value of those t variables.
- Reality with power attack
  - The attacker gets a sequence of power consumptions correlated to the variables.
  - Noisy leakage but not limited to t variables



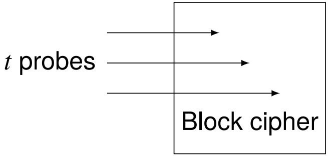
Probing model

Real life leakage



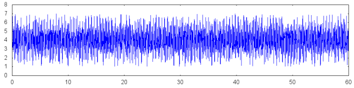
Relevance of probing model

- t-probing model
  - With security against t probes, combining t power consumption points as in a t-th order DPA will reveal no information to the adversary.
  - To recover the key, attacker must perform an attack of order at least $t + 1 \Rightarrow$ more complex.



Probing model

Real life leakage



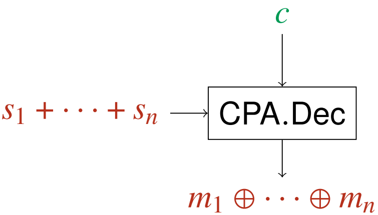
Relevance of probing model

- t-probing model
  - With security against t probes, combining t power consumption points as in a t-th order DPA will reveal no information to the adversary.
  - To recover the key, attacker must perform an attack of order at least $t+1 \Rightarrow$ more complex.

Never publish a high-order masking scheme without a proof of security !

- So many things can go wrong.
- Many countermeasures without proofs have been broken in the past.
- We have a poor intuition of high-order security.

Masking Kyber CPA decryption



- Arithmetic masking

$$s = s_1 + \cdots + s_n \pmod q$$

- Boolean masking

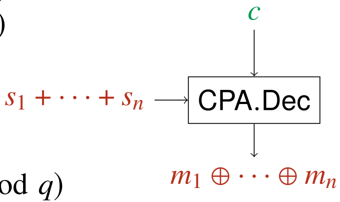
$$m = m_1 \oplus \cdots \oplus m_n \in \{0, 1\}$$

- High-order masking
  - We must process the shares s_i of s independently
  - without leaking information about s and m

Masking Kyber CPA decryption

- Kyber CPA decryption
  - $m = \text{th}(v - \mathbf{s}^T \cdot \mathbf{u})$ for $c = (\mathbf{u}, v)$.
  - Write $\mathbf{s} = \mathbf{s}_1 + \cdots + \mathbf{s}_n \pmod q$

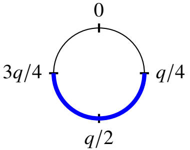
$$\mathbf{s}^T \cdot \mathbf{u} = \sum_{i=1}^{n} \mathbf{s}_i^T \cdot \mathbf{u}$$



$$\Rightarrow v - \mathbf{s}^T \cdot \mathbf{u} = w_1 + \cdots + w_n \pmod q$$

  - each w_i is computed independently from $\mathbf{s}_i$
- How to high-order compute ?

$$m_1 \oplus \cdots \oplus m_n = \text{th}(w_1 + \cdots + w_n \bmod q)$$

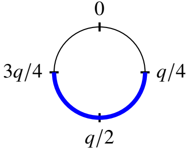


Arithmetic vs Boolean conversion

- How to high-order compute ?

$$m_1 \oplus \cdots \oplus m_n = \text{th}(w_1 + \cdots + w_n \bmod q)$$

  - If $q = 2^k$, then $m_1 \oplus \cdots \oplus m_n$ is MSB of $w_1 + \cdots + w_n \bmod q$



- Arithmetic vs Boolean masking conversions
  - We first convert from arithmetic to Boolean conversion

$$x_1 \oplus \cdots \oplus x_n = w_1 + \cdots + w_n \bmod q$$

  - We extract the MSB

$$m_1 \oplus \cdots \oplus m_n = \text{MSB}(x_1 \oplus \cdots \oplus x_n)$$

Boolean vs arithmetic masking

- Conversion between Boolean and arithmetic masking

$$x_1 \oplus \cdots \oplus x_n = w_1 + \cdots + w_n \bmod 2^k$$

	Direction	First-order complexity	High-order complexity
Goubin's algorithm [Gou01]	B → A A → B	$\mathcal{O}(1)$ $\mathcal{O}(k)$	- -
[CGV14]	B → A A → B	-	$\mathcal{O}(n^2 \cdot k)$
[BCZ18]	B → A	-	$\mathcal{O}(2^n)$

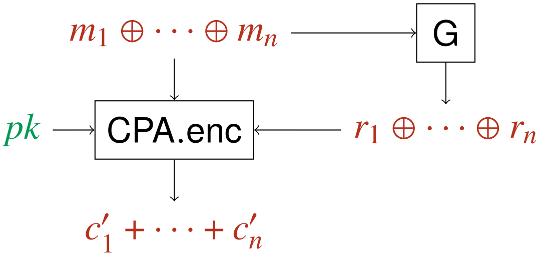
Masking Kyber CPA decryption

- How to high-order compute with prime q ?

$$m_1 \oplus \cdots \oplus m_n = \mathrm{th}(w_1 + \cdots + w_n \bmod q)$$

- Conversion from $A \bmod q$ to Boolean + secure ANDs [BGR+21]
  - $A \bmod q \rightarrow B$ [BBE+18] + 4 secure ANDs.
  - Complexity: $\mathcal{O}(n^2 \log q)$
- Modulus switching to modulo 2^k, then A to B [CGMZ22]
  - $A \bmod q \rightarrow A \bmod 2^k \rightarrow B$.
  - Complexity $\mathcal{O}(n^2 \log n)$.

High-order masking of Kyber.Decaps



- Masking Kyber re-encryption
  - Binomial sampling with $e = H_w(x) - H_w(y)$
  - 1-bit B $\to A \bmod q$, complexity $\mathcal{O}(n^2)$ [SPOG19]
- Masking the polynomial comparison
  - Hybrid approach with masked compressed coefficients and masked uncompressed coefficients [CGMZ23]
  - Complexity $\mathcal{O}(n^2)$

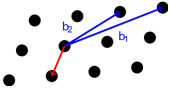
Fully masked implementation of Kyber

	Security order t							
	0	1	2	3	4	5	6	7
Intel i7	133	1 164	2 225	4 723	6 613	11 177	14 174	19 806
ARM Cortex-M3	3 173	21 492	39 539	69 348	-	-	-	-

Table: Kyber.Decaps cycles counts on Intel(R) Core(TM) i7-1065G7 and ARM Cortex-M3, in thousands of cycles.

The Dilithium signature scheme

- Dilithium
  - Lattice-based signature scheme, selected by NIST for standardization.
  - Security based on the hardness of the Module-Learning-With-Errors (MLWE) and the Module Short Integer Solution (MSIS) problems.



  - Fiat-Shamir with Aborts technique [L09]
- Key generation (simplified)
  - Compute $t = \mathbf{A} \cdot \mathbf{s}_1 + \mathbf{s}_2$
  - Public-key: $(\mathbf{A}, t)$, secret-key: $(\mathbf{s}_1, \mathbf{s}_2)$

Overview of Dilithium (without pk compression)

- Signature generation
  - $\mathbf{y} \leftarrow D$, with uniform small coefficients
  - $\mathbf{w} = \mathbf{A} \cdot \mathbf{y}$
  - $(\mathbf{w}_0, \mathbf{w}_1) = \text{Decompose}_q(\mathbf{w})$, $c = H(M \| \mathbf{w}_1)$
  - $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1, \quad \tilde{\mathbf{r}} = \mathbf{w}_0 - c\mathbf{s}_2$
  - Rejection sampling on $\|\mathbf{z}\|_\infty$ and $\|\tilde{\mathbf{r}}\|_\infty$
  - Return $(\mathbf{z}, c)$
- Signature verification of $(\mathbf{z}, c)$
  - $\mathbf{w}'_1 = \text{HighBits}(\mathbf{Az} - c\mathbf{t})$
  - Check $\|\mathbf{z}\|_\infty$ and $c = H(M \| \mathbf{w}'_1)$

$$\mathbf{Az} - c\mathbf{t} = \mathbf{A}(\mathbf{y} + c\mathbf{s}_1) - c(\mathbf{As}_1 + \mathbf{s}_2) = \mathbf{Ay} - c\mathbf{s}_2 \simeq \mathbf{Ay}$$

Side-channel attacks on Dilithium

- Key generation
  - $t = \mathbf{A} \cdot \mathbf{s}_1 + \mathbf{s}_2$
- Signature generation
  - $\mathbf{y} \leftarrow D$, with uniform small coefficients
  - $\mathbf{w} = \mathbf{A} \cdot \mathbf{y}$
  - $(\mathbf{w}_0, \mathbf{w}_1) = \text{Decompose}_q(\mathbf{w})$
  - $c = H(M \| \mathbf{w}_1)$
  - $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$
  - $\tilde{\mathbf{r}} = \mathbf{w}_0 - c\mathbf{s}_2$
  - Rejection sampling on $\|\mathbf{z}\|_\infty$ and $\|\tilde{\mathbf{r}}\|_\infty$
  - Return $(\mathbf{z}, c)$

Side-channel attacks on Dilithium

- Key generation
  - $t = \mathbf{A} \cdot \mathbf{s}_1 + \mathbf{s}_2 \longleftarrow$ [HLK+21]
- Signature generation
  - $\mathbf{y} \leftarrow D$, with uniform small coefficients $\longleftarrow$ [MUTS22]
  - $\mathbf{w} = \mathbf{A} \cdot \mathbf{y}$
  - $(\mathbf{w}_0, \mathbf{w}_1) = \text{Decompose}_q(\mathbf{w}) \longleftarrow$ [BVC+23]
  - $c = H(M \| \mathbf{w}_1)$
  - $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1 \longleftarrow$ [CKA+21]
  - $\tilde{\mathbf{r}} = \mathbf{w}_0 - c\mathbf{s}_2$
  - Rejection sampling on $\|\mathbf{z}\|_\infty$ and $\|\tilde{\mathbf{r}}\|_\infty$
  - Return $(\mathbf{z}, c)$

High-order masking of Dilithium

- The variables $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{y}$, $\mathbf{w}_0$, $\tilde{\mathbf{r}}$ must be masked.
  - $\mathbf{z}$ must also be masked before rejection sampling.
- Key generation
  - $t = \mathbf{A} \cdot \mathbf{s}_1 + \mathbf{s}_2$
- Signature generation
  - $\mathbf{y} \leftarrow D$, with uniform small coefficients
  - $\mathbf{w} = \mathbf{A} \cdot \mathbf{y}$
  - $(\mathbf{w}_0, \mathbf{w}_1) = \text{Decompose}_q(\mathbf{w})$
  - $c = H(M \| \mathbf{w}_1)$
  - $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$
  - $\tilde{\mathbf{r}} = \mathbf{w}_0 - c\mathbf{s}_2$
  - Rejection sampling on $\|\mathbf{z}\|_\infty$ and $\|\tilde{\mathbf{r}}\|_\infty$
  - Return $(\mathbf{z}, c)$

Fully masked implementation of Dilithium

	Security order t				
	0	1	2	3	4
Dilithium2	506	26602 (×53)	54945 (×109)	101020 (×200)	155025 (×306)
Dilithium3	853	36986 (×43)	83696 (×98)	130590 (×153)	205473 (×241)
Dilithium5	989	38069 (×38)	87809 (×89)	137034 (×139)	201838 (×204)
NTTs	-	304	451	598	732
Sample y	-	3034	5135	7890	13127
Compute Ay	-	616	916	1121	1515
Decompose	-	13088	32963	52030	84131
$z = y + c \cdot s_1$	-	355	528	641	872
Reject	-	18956	42856	67281	103840
$w - c \cdot s_2$	-	262	390	487	635

- Effect of high-order masking
  - Slow operation (NTT) become fast.
  - Fast operation (Decompose, Reject) become slow.

Conclusion

- Challenges of protecting post-quantum algorithms against side-channel attacks
  - Side-channel attacks are powerful (template attacks, fault attacks, etc.)
  - Variety of operations in post-quantum algorithms. Boolean vs arithmetic operations.
- New challenges and future work
  - Minimize the complexity penalty in countermeasures.
  - Side-channel friendly schemes: Raccoon signature NIST submission