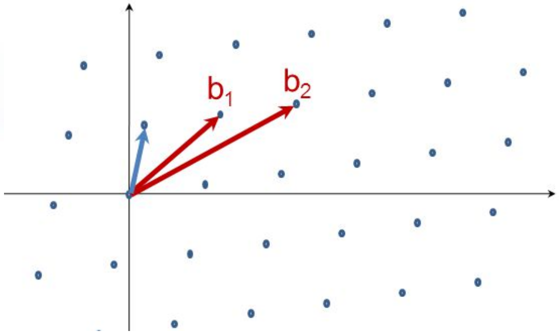


Side-Channel Analysis of Lattice-based PQC Candidates

Prasanna Ravi and Sujoy Sinha Roy

prasanna.ravi@ntu.edu.sg, sujoy.sinharoy@iaik.tugraz.at

Notice

- Talk includes published works from journals, conferences, and IACR ePrint Archive.
- Talk includes works of other researchers (cited appropriately)
- For easier explanation, we ‘simplify’ concepts
- Due to time limit, we do not exhaustively cover all relevant works.
  - Main focus on LWE/LWR-based PKE/KEM schemes
  - Timing, Power, and EM side-channels

Classification of PQC finalists and alternative candidates

Lattice-based Cryptography

- **Public Key Encryption (PKE)/ Key Encapsulation Mechanisms (KEM)**
  - **LWE/LWR-based** (Kyber, SABER, Frodo)
  - **NTRU-based** (NTRU, NTRUPrime)
- **Digital Signature Schemes (DSS)**
  - **LWE, Fiat-Shamir with Aborts** (Dilithium)
  - **NTRU, Hash and Sign** (FALCON)

This talk

Outline

- Background:
  - Learning With Errors (LWE) Problem
  - LWE/LWR-based PKE framework
- Overview of side-channel attacks:
  - Algorithmic-level
  - Implementation-level
- Overview of masking countermeasures
- Conclusions and future works

Given two linear equations with unknown x and y

$$3x + 4y = 26$$
$$2x + 3y = 19$$

or

$$\begin{bmatrix} 3 & 4 \\ 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 26 \\ 19 \end{bmatrix}$$

Find x and y.

Solving a system of linear equations

System of linear equations with unknown ***s***

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ \vdots \\ b_m \end{pmatrix}$$

Gaussian elimination solves ***s*** when number of equations $m \geq n$

Solving a system of linear equations with errors

Matrix **A** Vector **b**

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \\ \vdots \\ e_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ \vdots \\ b_m \end{pmatrix} \mod q$$

- Search **Learning With Errors** (LWE) problem:
  Given **(A, b)** → computationally infeasible to solve ***(s, e)***
- Decisional **Learning With Errors** (LWE) problem:
  Given **(A, b)** → hard to distinguish from uniformly random

LWE

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} * \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \pmod{q}$$

Uniformly random matrix

Ring LWE

$$\begin{pmatrix} a_0 & -a_3 & -a_2 & -a_1 \\ a_1 & a_0 & -a_3 & -a_2 \\ a_2 & a_1 & a_0 & -a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} * \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \pmod{q}$$

Matrix by rotating first column

Ring LWE

$$\begin{pmatrix} a_0 & -a_3 & -a_2 & -a_1 \\ a_1 & a_0 & -a_3 & -a_2 \\ a_2 & a_1 & a_0 & -a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} * \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} + \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \pmod q$$

$a(x) * s(x) + e(x) = b(x) \pmod q \pmod{x^4 + 1}$

where

$a(x) = (a_0 + a_1 x + a_2 x^2 + a_3 x^3)$

$s(x) = (s_0 + s_1 x + s_2 x^2 + s_3 x^3)$

$e(x) = (e_0 + e_1 x + e_2 x^2 + e_3 x^3)$

$b(x) = (b_0 + b_1 x + b_2 x^2 + b_3 x^3)$

Polynomial arithmetic

Module LWE

$$\left(\begin{array}{cccc|cccc} a_0 & -a_3 & -a_2 & -a_1 & a_8 & -a_{11} & -a_{10} & -a_9 \\ a_1 & a_0 & -a_3 & -a_2 & a_9 & a_8 & -a_{11} & -a_{10} \\ a_2 & a_1 & a_0 & -a_3 & a_{10} & a_9 & a_8 & -a_{11} \\ a_3 & a_2 & a_1 & a_0 & a_{11} & a_{10} & a_7 & a_8 \\ \hline a_4 & -a_7 & -a_6 & -a_5 & a_{12} & -a_{15} & -a_{14} & -a_{13} \\ a_5 & a_4 & -a_7 & -a_6 & a_{13} & a_{12} & -a_{15} & -a_{14} \\ a_6 & a_5 & a_4 & -a_7 & a_{14} & a_{13} & a_{12} & -a_{15} \\ a_7 & a_6 & a_5 & a_4 & a_{15} & a_{14} & a_{13} & a_{12} \end{array}\right) * \left(\begin{array}{c} s_0 \\ s_1 \\ s_2 \\ s_3 \\ \hline s_4 \\ s_5 \\ s_6 \\ s_7 \end{array}\right) + \left(\begin{array}{c} e_0 \\ e_1 \\ e_2 \\ e_3 \\ \hline e_4 \\ e_5 \\ e_6 \\ e_7 \end{array}\right) = \left(\begin{array}{c} b_0 \\ b_1 \\ b_2 \\ b_3 \\ \hline b_4 \\ b_5 \\ b_6 \\ b_7 \end{array}\right)$$

$$\begin{pmatrix} a_{0,0}(x) & a_{0,1}(x) \\ a_{1,0}(x) & a_{1,1}(x) \end{pmatrix} * \begin{pmatrix} s_0(x) \\ s_1(x) \end{pmatrix} + \begin{pmatrix} e_0(x) \\ e_1(x) \end{pmatrix} = \begin{pmatrix} b_0(x) \\ b_1(x) \end{pmatrix}$$

Learning with Rounding (LWR)

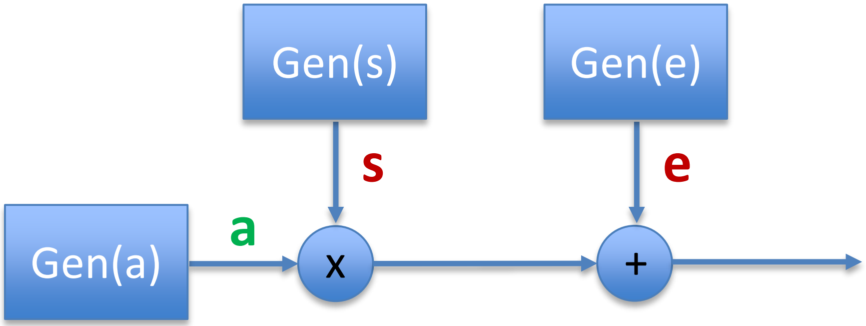
$$\left\lfloor \frac{p}{q} \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} \right\rceil = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ \vdots \\ b_m \end{pmatrix} \mod p$$

where $p < q$

- Errors are generated by performing rounding
- LWR can be extended to “Ring LWR” and “Module LWR”

Ring LWE-based PKE (IND-CPA secure)

- **Key Generation:**
  - **Output:** public key (pk), secret key (sk)



$b = a.s + e$ (Ring LWE sample)

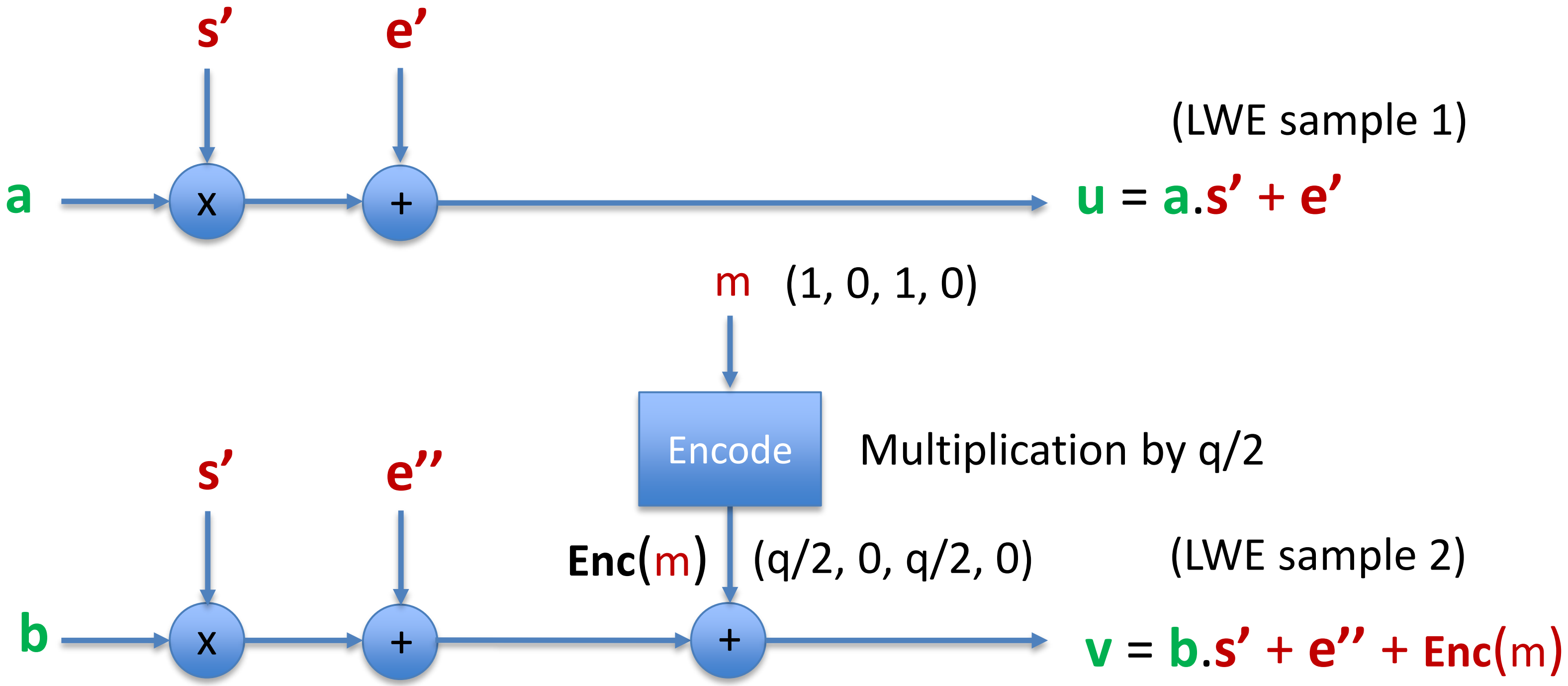
Arithmetic operations are performed in a polynomial ring R_q

Public Key (pk): (a,b)
Secret Key (sk): (s)

V. Lyubashevsky, C. Peikert, and O. Regev. "On Ideal Lattices and Learning with Errors Over Rings". IACR ePrint 2012/230.

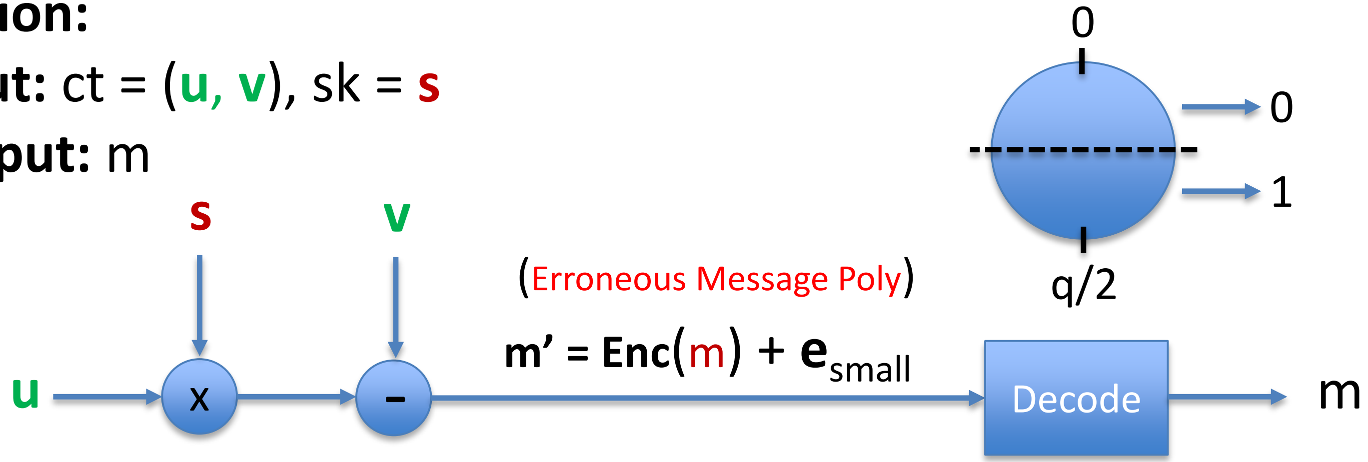
Ring LWE-based PKE (IND-CPA secure)

- **Encryption:**
  - **Input: pk** = (a,b), message m
  - **Output: ct** = (u,v)



Ring LWE-based PKE (IND-CPA secure)

- **Decryption:**
  - **Input:** ct = (**u**, **v**), sk = **s**
  - **Output:** m



$$\mathbf{v} - \mathbf{u}.\mathbf{s} = \mathbf{m'} = \mathbf{Enc}(m) + (\mathbf{e}.\mathbf{s'} + \mathbf{e''} + \mathbf{e'}.\mathbf{s})$$
$$= \mathbf{Enc}(m) + \mathbf{e}_{small}$$

General Framework for PKE

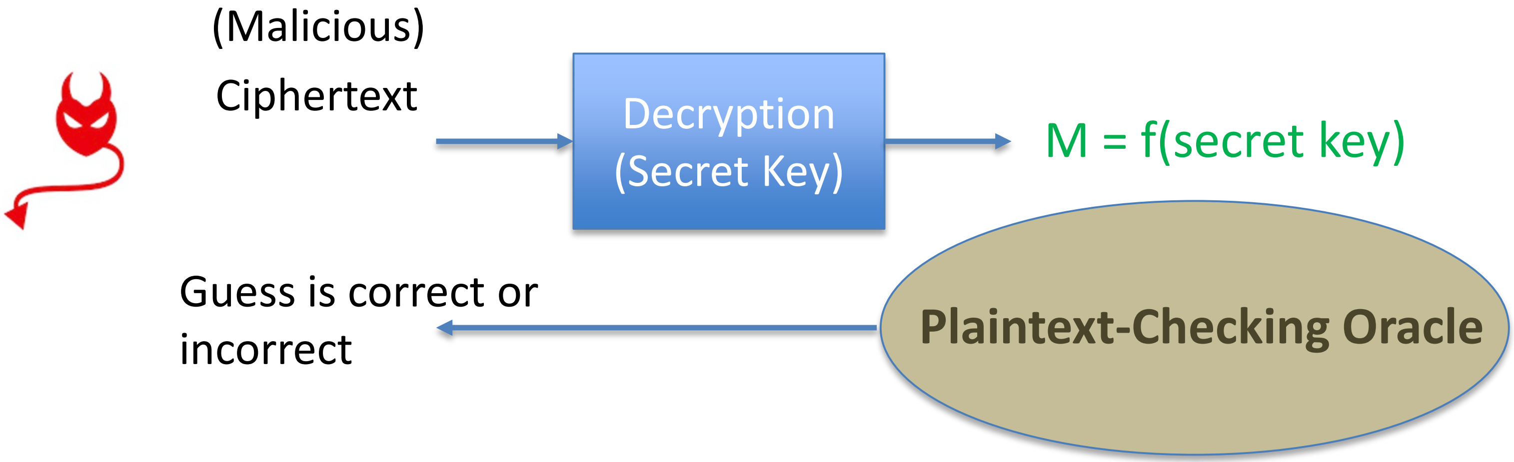
- The "ring LWE PKE" example can be extended to describe various standard/ring/module LWE/LWR-based schemes.

- Differences in them
  - Variant of LWE/LWR problem
  - Operating Ring, Modulus etc.
  - Choice of Distribution for secret and error.
  - Choice of Error Correcting Code (to reduce decryption failures)
  - Specific optimization techniques
  - Protocol-level differences
  - …

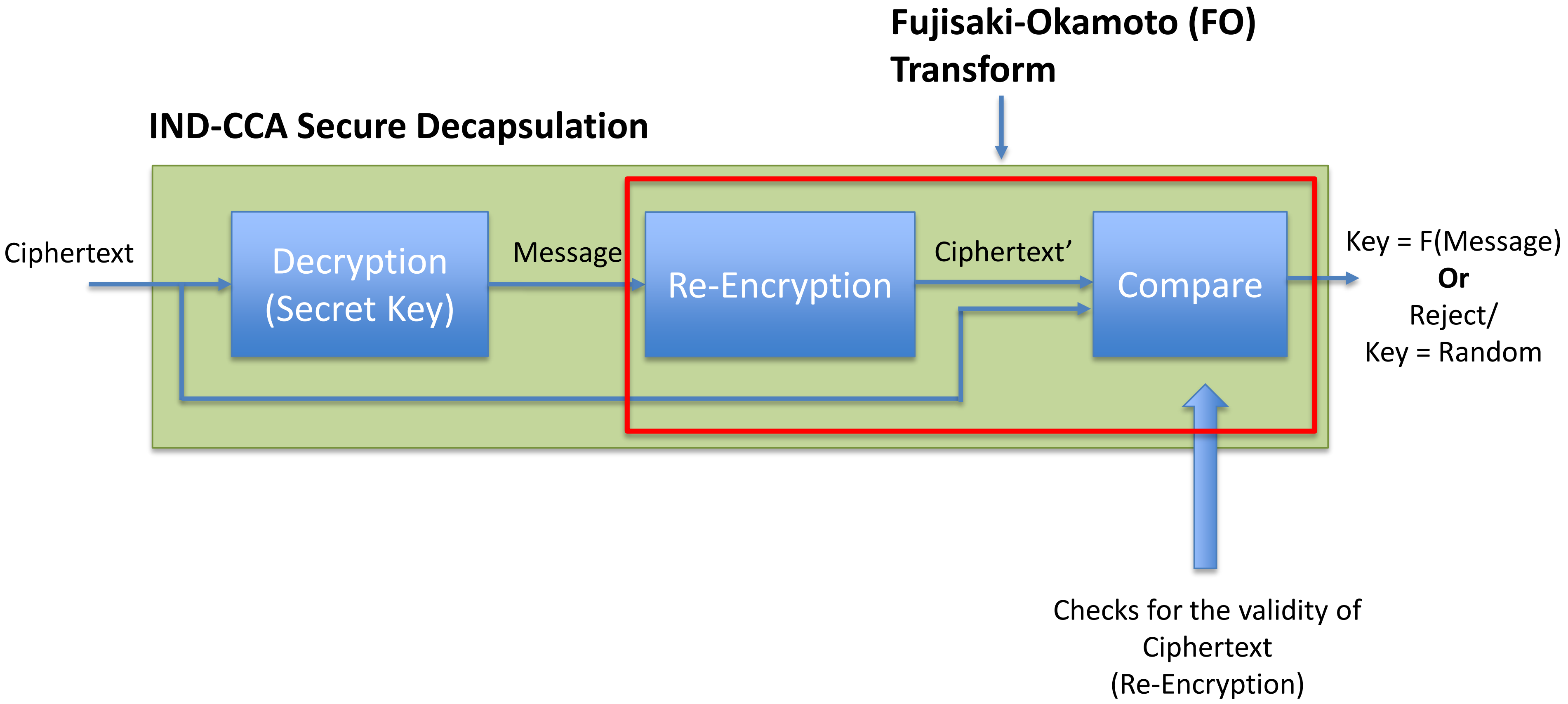
We will use the "ring LWE PKE" example for different side-channel attacks

Chosen Ciphertext Attack (CCA): Key Recovery

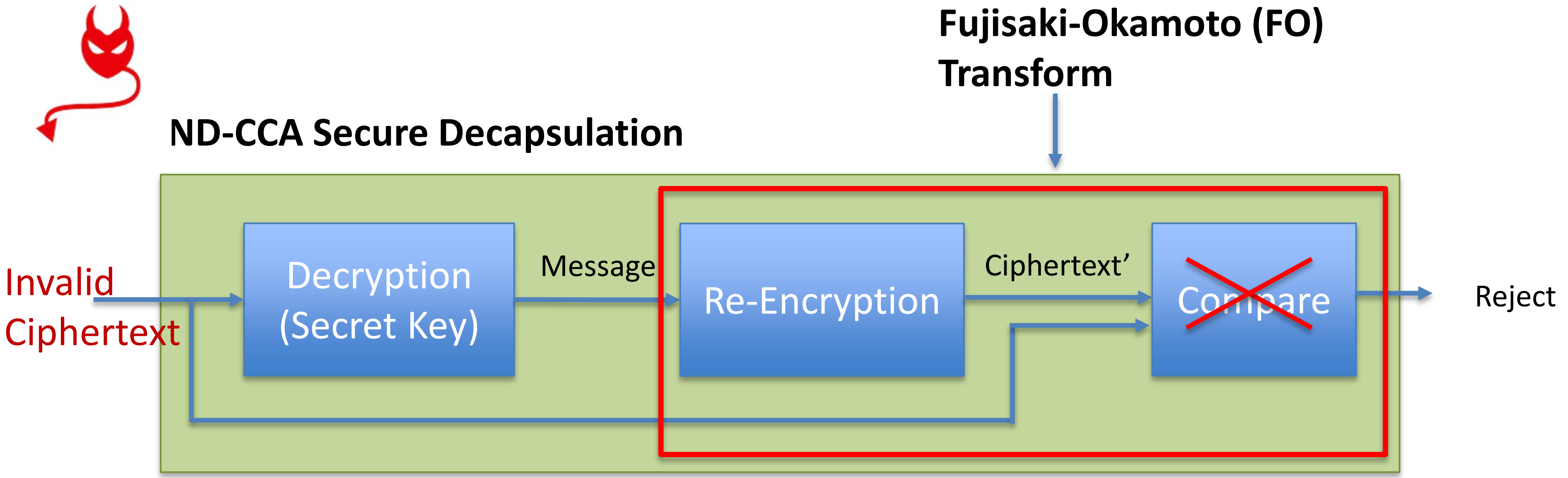
Attacker targets the decryption procedure of IND-CPA PKE



CCA-security using FO transformation



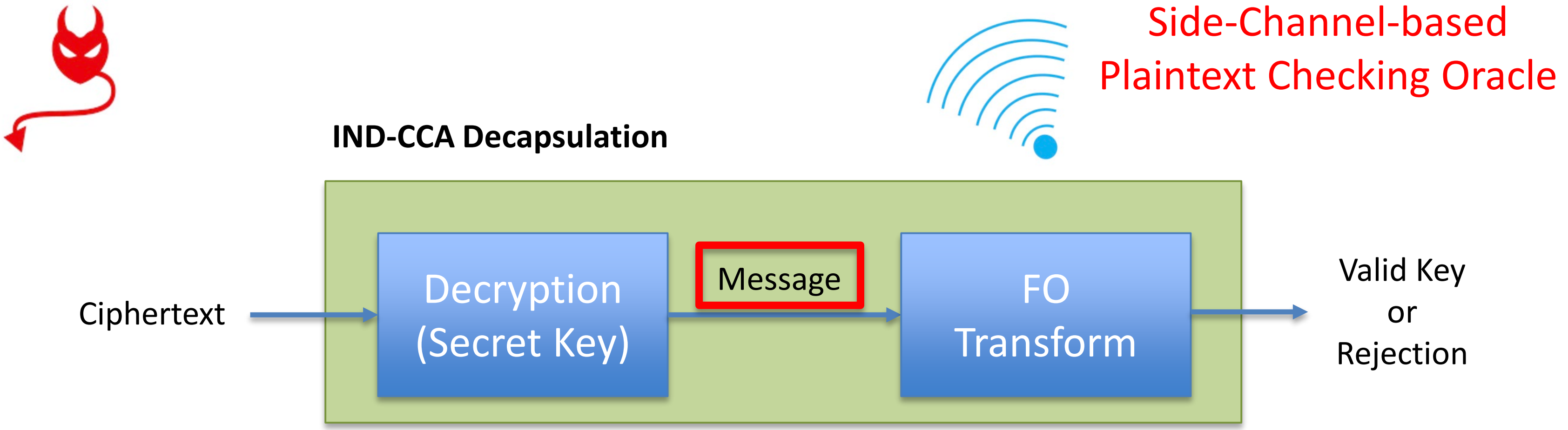
CCA-security using FO transformation



Attacker cannot gain any information about the message.

Can attacker use side-channel(s) to guess the messages?

Side-Channel Assisted Chosen Ciphertext Attacks



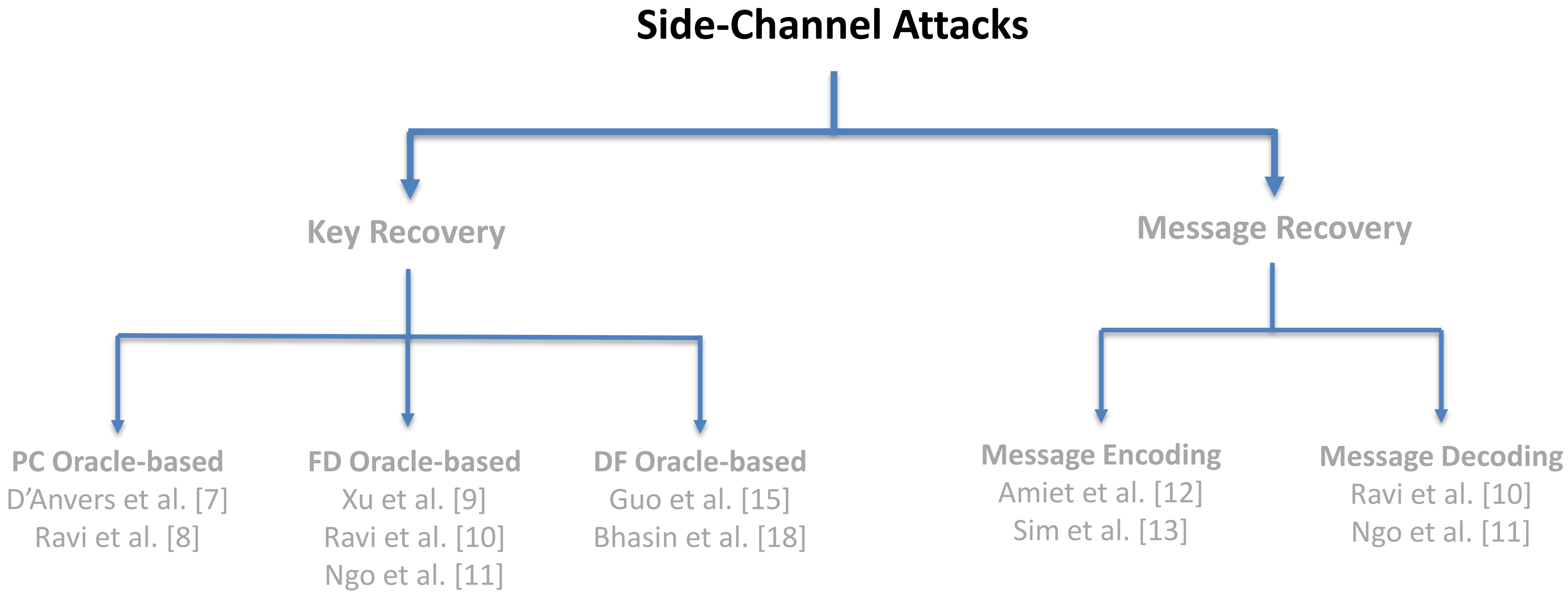
Outline

- Background:
  - Learning With Error (LWE) Problem
  - LWE/LWR-based PKE framework
- **Overview of side-channel attacks:**
  - **Algorithmic-level**
  - **Implementation-level**
- Overview of masking countermeasures:
- Conclusions and future works and Conclusion:

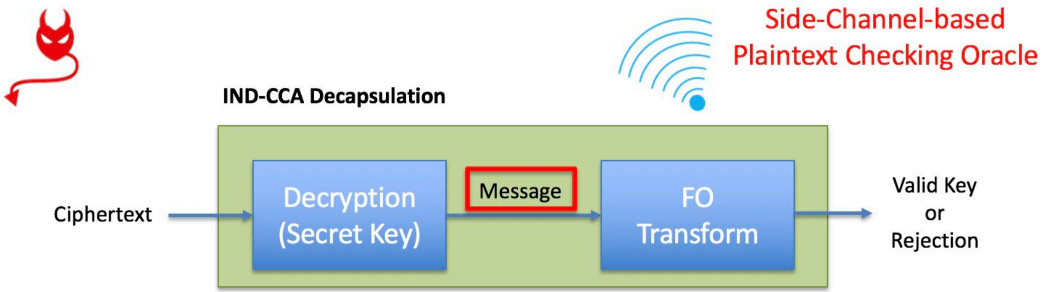
Outline

- Background:
  - Learning With Error (LWE) Problem
  - LWE/LWR-based PKE framework
- **Overview of side-channel attacks:**
  - **Algorithmic-level**
  - **Implementation-level**
- Overview of masking countermeasures:
- Conclusions and future works and Conclusion:

Classification of SCA of lattice-based PKE/KEMs:



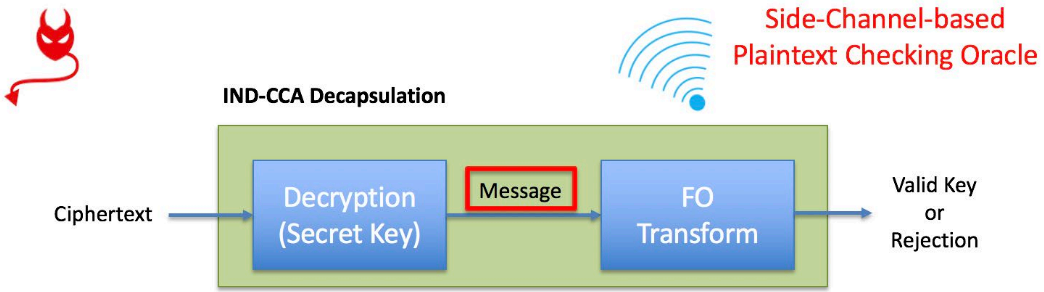
Side-Channel Assisted Chosen Ciphertext Attacks



- Bauer et al. [BGRR19] – Proposed to use SCA to assist chosen ciphertext attacks for LWE/LWR-based PKE/KEMs.
- D'Anvers et al. [DTVV19] demonstrated a concrete side-channel based Plaintext checking Oracle Attack:
  - **Target Schemes: LAC** and RAMSTAKE
  - **Timing Side-Channel**: Variable run-time of error correcting codes
- Ravi et al. [RRCB20] generalized the attack to constant time implementations:
  - **EM Side-Channel**: Extension of technique to multiple LWE/LWR-based PKE/KEMs

[BGRR19] Bauer, Aurélie, Henri Gilbert, Guénaël Renault, and Mélissa Rossi. "Assessment of the key-reuse resilience of NewHope." In *Cryptographers' Track at the RSA Conference*, pp. 272-292. Springer, Cham, 2019.
[DTVV19] D'Anvers, Jan-Pieter, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. "Timing attacks on error correcting codes in post-quantum schemes." In *Proceedings of ACM Workshop on Theory of Implementation Security Workshop*, pp. 2-9. 2019.
[RRCB20] Ravi, Prasanna, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. "Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020): 307-335.

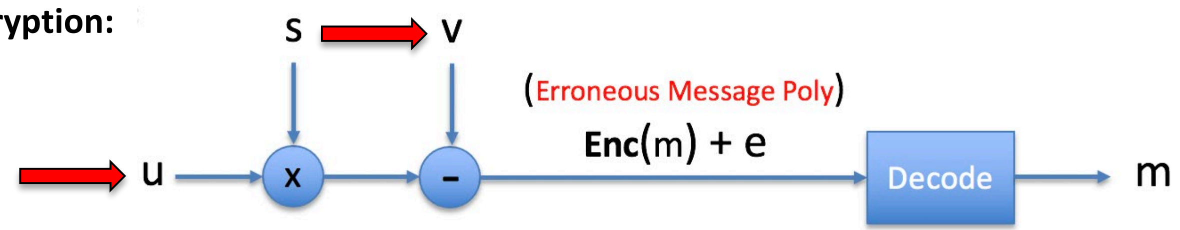
Side-Channel Assisted Chosen Ciphertext Attacks



- Plaintext-Checking (PC) Oracle based attack consists of two parts:
  - **Part-I**: Construction of **malicious ciphertexts**
  - **Part-II**: **Perform SCA** to obtain useful information about **decryption output** for malicious ciphertexts

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

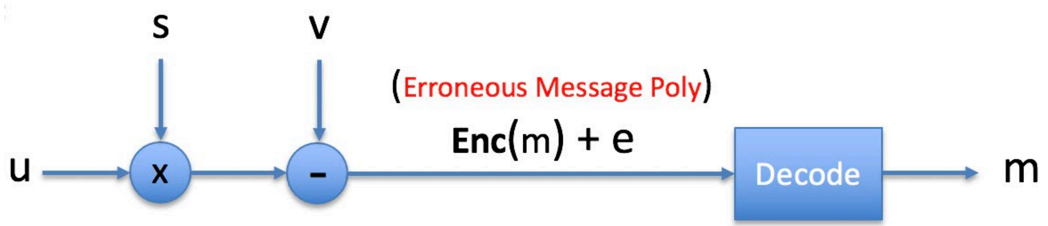
- **Decryption:**



Chosen u	k	0	0	0	0	0	0
u.s	$k.s_0$	$k.s_1$	$k.s_2$	$k.s_3$	$k.s_4$	$k.s_5$	$k.s_6$
Chosen v	p	0	0	0	0	0	0
m' = u.s - v	$k.s_0$-p	$k.s_1$	$k.s_2$	$k.s_3$	$k.s_4$	$k.s_5$	$k.s_6$
m = Decode(**m**)	$f(s_0)$	0	0	0	0	0	0
	m_0	m_1	m_2	m_3	m_4	m_5	m_6

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- **Decryption:**



m'=Decode(**m'**) | $f(s_0)$ | 0 | 0 | 0 | 0 | 0 | 0

m = [0, 0, 0, 0, 0, 0, 0, 0, … , 0] (O)
or
m = [1, 0, 0, 0, 0, 0, 0, 0, … , 0] (X)

Secret Coeff.	(k,p)	
	(21,3)	(12,1)
-1	X	X
0	X	O
1	O	O

Binary Distinguisher for every candidate of s_0
(Round5)

Recover s_0 using two ciphertext queries

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- Polynomial multiplication in polynomial rings have special rotational properties.

$$R_q = \mathbb{Z}_q[x] \bmod (x^n - 1) \qquad R_q = \mathbb{Z}_q[x] \bmod (x^n + 1)$$

- Multitplication of a polynomial with x^i **”rotates”** the polynomial by ”i” positions (cyclic or anti-cyclic)

Chosen u	k	0	0	0	0	0	0
Chosen v	p	0	0	0	0	0	0
m’=Decode(**m’**)	$f(s_0)$	0	0	0	0	0	0

Recover s_0 using knowledge of O/X

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- Polynomial multiplication in polynomial rings have special rotational properties.

$$R_q = \mathbb{Z}_q[x] \bmod (x^n - 1) \qquad R_q = \mathbb{Z}_q[x] \bmod (x^n + 1)$$

- Multitplication of a polynomial with x^i **"rotates"** the polynomial by "i" positions (cyclic or anti-cyclic)

Chosen u	0	k	0	0	0	0	0
Chosen v	p	0	0	0	0	0	0
m'=Decode(**m'**)	$f(s_{n-1})$	0	0	0	0	0	0

Recover s_{n-1} using knowledge of O/X

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- Polynomial multiplication in polynomial rings have special rotational properties.

$$R_q = \mathbb{Z}_q[x] \bmod (x^n - 1) \qquad R_q = \mathbb{Z}_q[x] \bmod (x^n + 1)$$

- Multitplication of a polynomial with x^i **”rotates”** the polynomial by ”i” positions (cyclic or anti-cyclic)

Chosen u	0	0	k	0	0	0	0
Chosen v	p	0	0	0	0	0	0
m’=Decode(**m’**)	$f(s_{n-2})$	0	0	0	0	0	0

Recover s_{n-2} using knowledge of O/X

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- Polynomial multiplication in polynomial rings have special rotational properties.

$$R_q = \mathbb{Z}_q[x] \bmod (x^n - 1) \qquad R_q = \mathbb{Z}_q[x] \bmod (x^n + 1)$$

- Multitplication of a polynomial with x^i **”rotates”** the polynomial by ”i” positions (cyclic or anti-cyclic)

- No Rotation property in schemes based on Standard LWE/LWR (FrodoKEM) - But, attack still works…
  - Location of non-zero bit of message changes (depending upon secret coefficient to recover)

Chosen u	0	0	0	k	0	0	0
Chosen v	p	0	0	0	0	0	0
m’=Decode(**m’**)	$f(s_{n-3})$	0	0	0	0	0	0

Recover s_{n-3} using knowledge of O/X

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- Polynomial multiplication in polynomial rings have special rotational properties.

$$R_q = \mathbb{Z}_q[x] \bmod (x^n - 1) \qquad R_q = \mathbb{Z}_q[x] \bmod (x^n + 1)$$

- Multitplication of a polynomial with x^i **"rotates"** the polynomial by "i" positions (cyclic or anti-cyclic)

- No Rotation property in schemes based on Standard LWE/LWR (FrodoKEM) - But, attack still works…
  - Location of non-zero bit of message changes (depending upon secret coefficient to recover)

Chosen u	k	0	0	0	0	0	0
Chosen v	p	0	0	0	0	0	0
m'=Decode(**m'**)	$f(s_0)$	0	0	0	0	0	0

Recover s_{n-2} using knowledge of O/X

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- ❑ Polynomial multiplication in polynomial rings have special rotational properties.

$$R_q = \mathbb{Z}_q[x] \bmod (x^n - 1) \qquad R_q = \mathbb{Z}_q[x] \bmod (x^n + 1)$$

- ❑ Multitplication of a polynomial with x^i **”rotates”** the polynomial by ”i” positions (cyclic or anti-cyclic)

- ❑ No Rotation property in schemes based on Standard LWE/LWR (FrodoKEM) - But, attack still works…
  - ❑ Location of non-zero bit of message changes (depending upon secret coefficient to recover)

Chosen u	k	0	0	0	0	0	0
Chosen v	0	p	0	0	0	0	0
m’=Decode(**m’**)	0	$f(s_1)$	0	0	0	0	0

Recover s_{n-1} using knowledge of O/X

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- Polynomial multiplication in polynomial rings have special rotational properties.

$$R_q = \mathbb{Z}_q[x] \bmod (x^n - 1) \qquad R_q = \mathbb{Z}_q[x] \bmod (x^n + 1)$$

- Multitplication of a polynomial with x^i **”rotates”** the polynomial by ”i” positions (cyclic or anti-cyclic)

- No Rotation property in schemes based on Standard LWE/LWR (FrodoKEM) - But, attack still works…
  - Location of non-zero bit of message changes (depending upon secret coefficient to recover)

Chosen u	k	0	0	0	0	0	0
Chosen v	0	0	p	0	0	0	0
m’=Decode(**m’**)	0	0	$f(s_2)$	0	0	0	0

Recover s_{n-1} using knowledge of O/X

PC Oracle-based SCA: Constructing Malicious CTs (Part-I)

- Polynomial multiplication in polynomial rings have special rotational properties.

$$R_q = \mathbb{Z}_q[x] \bmod (x^n - 1) \qquad R_q = \mathbb{Z}_q[x] \bmod (x^n + 1)$$

- Multitplication of a polynomial with x^i **”rotates”** the polynomial by ”i” positions (cyclic or anti-cyclic)

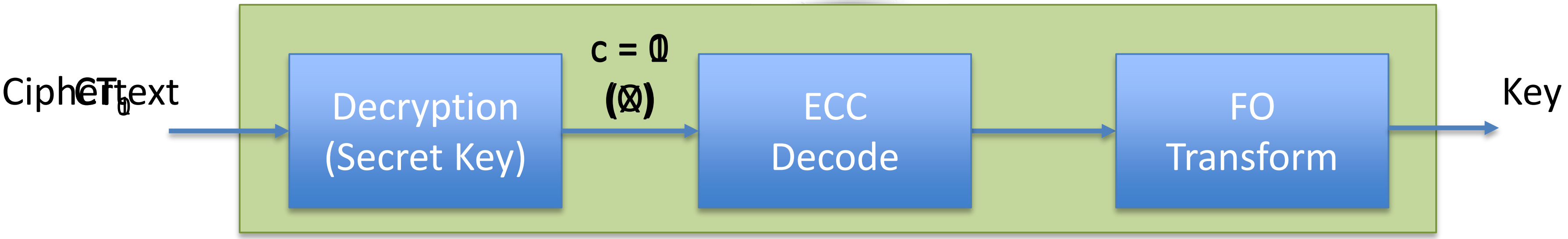
- No Rotation property in schemes based on Standard LWE/LWR (FrodoKEM) - But, attack still works…
    - Location of non-zero bit of message changes (depending upon secret coefficient to recover)

Chosen u	k	0	0	0	0	0	0
Chosen v	0	0	0	p	0	0	0
m’=Decode(**m’**)	0	0	0	$f(s_3)$	0	0	0

Recover s_{n-1} using knowledge of O/X

PC Oracle-based SCA: Using SCA as O/X distinguisher (Part-II)

- D’Anvers et al. [DTVV19] exploited variable runtime of error correcting codes in LAC and RAMSTAKE.
  - O - Valid codeword, X - Invalid codeword
  - Decode_Time(O) << Decode_Time(X)



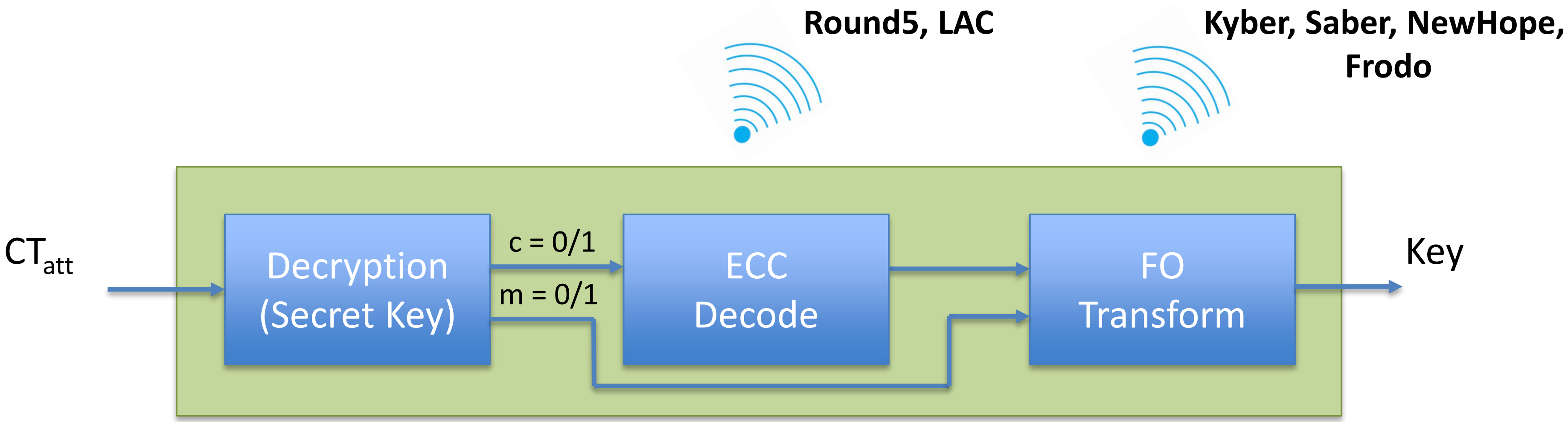
- **Pre-Processing Phase (Template Generation):**
  - Create ciphertexts for both classes: O and X.
  - Query ciphertexts to build template for O and X.

- **Attack Phase (Template Matching):**
  - Query with malicious chosen ciphertexts and classify as O or X
  - Use O/X info. to recover secret key

PC Oracle-based SCA: Using SCA as O/X distinguisher (Part-II)

- Attack generalized to constant-time implementations by Ravi et al. [RRCB20] using the EM side-channel for multiple LWE/LWR-based PKE/KEMs.

- Vulnerable operations leaking EM side-channel information about O/X:
  - **ECC Decoding Procedure** (Decode(O) != Decode(X))
  - **FO Transform** (Hash(0,pk) != Hash(1,pk))



PC Oracle-based SCA: Experimental Results

Tabulation of attack complexity on different LWE/LWR-based based PKE/KEMs (Source: Ravi et al. [RRCB20])
Target: ARM Cortex-M4, EM-side channel

Scheme	# Coeffs	# traces for template	# Attack traces	Time (Minutes)
Kyber (KYBER512)	512	2 x 50 = 100	7.7k	10.8
Round5 (R5ND_1KEM_5d)	490	2 x 50 = 100	2.9k	4.5
LAC (LAC128)	512	2 x 50 = 100	3.0k	25

- ❑ **ADVANTAGE:**
  - ❑ Easy SCA (**Classification Problem with two classes**) – No sophisticated SCA setup required.
  - ❑ Non-profiled Attack
  - ❑ Attack done in a matter of a few minutes (few thousand traces).

- ❑ **COUNTERMEASURE:** Concrete Masking (additive sharing of message)

Classification of SCA on LWE/LWR-based PKE/KEMs:

SCA
(Algorithmic Level)

Key Recovery (Chosen Ciphertext)

PC Oracle-based
D’Anvers et al. [DTVV19]
Ravi et al. [RRCB20]

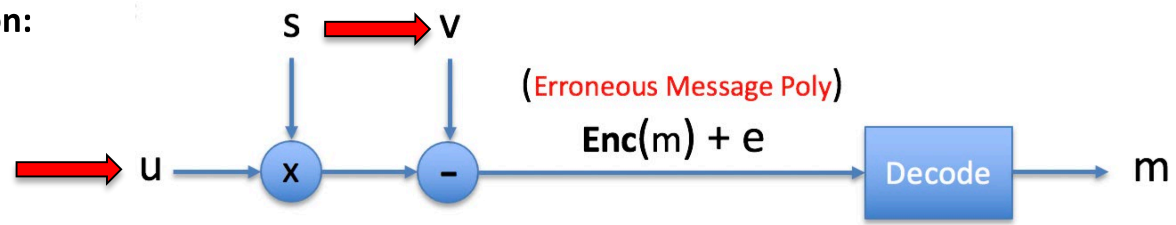
A Few Observations on the PC Oracle-based SCA…

- Key recovery still requires a few thousand traces.

- Can we do better with much fewer traces???

A Few Observations on the PC Oracle-based SCA…

- **Decryption:**



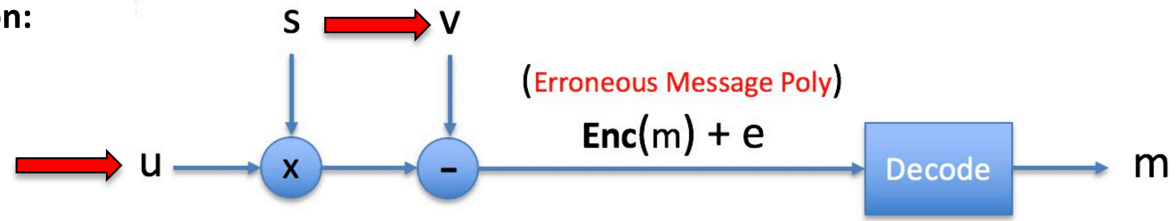
Chosen u	k	0	0	0	0
u.s	$k.s_0$	$k.s_1$	$k.s_2$	$k.s_3$	$k.s_4$
Chosen v	p	0	0	0	0
m’ = u.s - v	$k.s_0$-p	$k.s_1$	$k.s_2$	$k.s_3$	$k.s_4$
m’=Decode(**m’**)	$f(s_0)$	0	0	0	0

Full Decryption Oracle

Can we use all the message bits?

Full Decryption (FD) Oracle-based SCA:

- Decryption:



Chosen u	k	0	0	0	0	
u.s	$k.s_0$	$k.s_1$	$k.s_2$	$k.s_3$	$k.s_4$	
Chosen v	p	p	p	p	p	**Choose different v**
m' = u.s - v	$k.s_0$-p	$k.s_1$-p	$k.s_2$-p	$k.s_3$-p	$k.s_4$-p	
m'=Decode(**m'**)	$f(s_0)$	$f(s_1)$	$f(s_2)$	$f(s_3)$	$f(s_4)$	

Full Decryption (FD) Oracle-based SCA:

Chosen u	k	0	0	0	0	0	0
Chosen v	p	p	p	p	p	p	p
m’=Decode(**m’**)	$f(s_0)$	$f(s_1)$	$f(s_2)$	$f(s_3)$	$f(s_4)$	$f(s_5)$	$f(s_6)$

- ❑ Proposed by Xu et al. [XPRO20]:
  - ❑ Full Key Recovery for Kyber512 in **8** queries (improved to 6 queries by Ravi et al. [RBRC20])
- ❑ Ngo et al. [NDGJ21] proposed improved techniques for key recovery with FD oracle:
  - ❑ Error Correction mechanism for noise in recovered message (Saber - **16** queries)

[XPRO20] Xu, Zhuang, Owen Pemberton, Sujoy Sinha Roy, and David Oswald. *Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems with Chosen Ciphertexts: The Case Study of Kyber*. Cryptology ePrint Archive, Report 2020/912, 2020. https://eprint.iacr.org/2020/912, 2020.
[RBRC20] Ravi, Prasanna, Shivam Bhasin, Sujoy Sinha Roy, Anupam Chattopadhyay. "On Exploiting Message Leakage in (few) NIST PQC Candidates for Practical Message Recovery and Key Recovery Attacks." Cryptology ePrint Archive, Report 2020/1559, 2020. https://eprint.iacr.org/2020/1559, 2020.
[NDGJ21] Ngo, Kalle, Elena Dubrova, Qian Guo, and Thomas Johansson. "A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM.” Cryptology ePrint Archive, Report 2021/079, 2021. https://eprint.iacr.org/2021/079, 2021.

Classification of SCA on LWE/LWR-based PKE/KEMs:

SCA
(Algorithmic Level)

Key Recovery (Chosen Ciphertext)

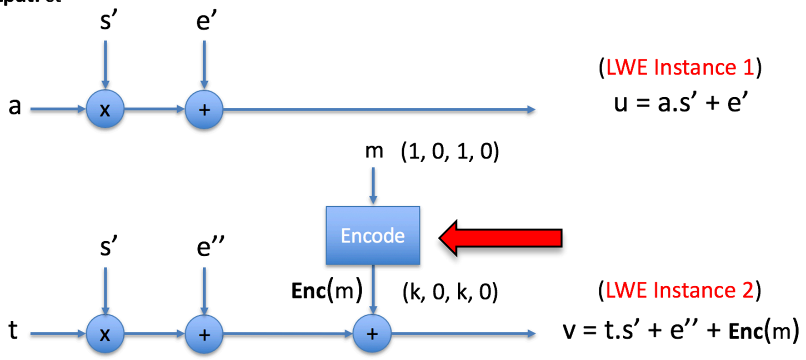
PC Oracle-based
D’Anvers et al. [7]
Ravi et al. [8]

FD Oracle-based
Xu et al. [XPRO20]
Ravi et al. [RBRC20]
Ngo et al. [NDGJ21]

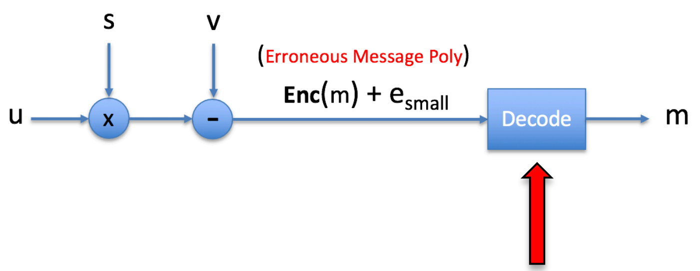
How does an attacker perform full message recovery through SCA???

Encoding and Decoding Functions:

- **Encryption:**
  - **Input:** pk = (a,t), message m
  - **Output:** ct

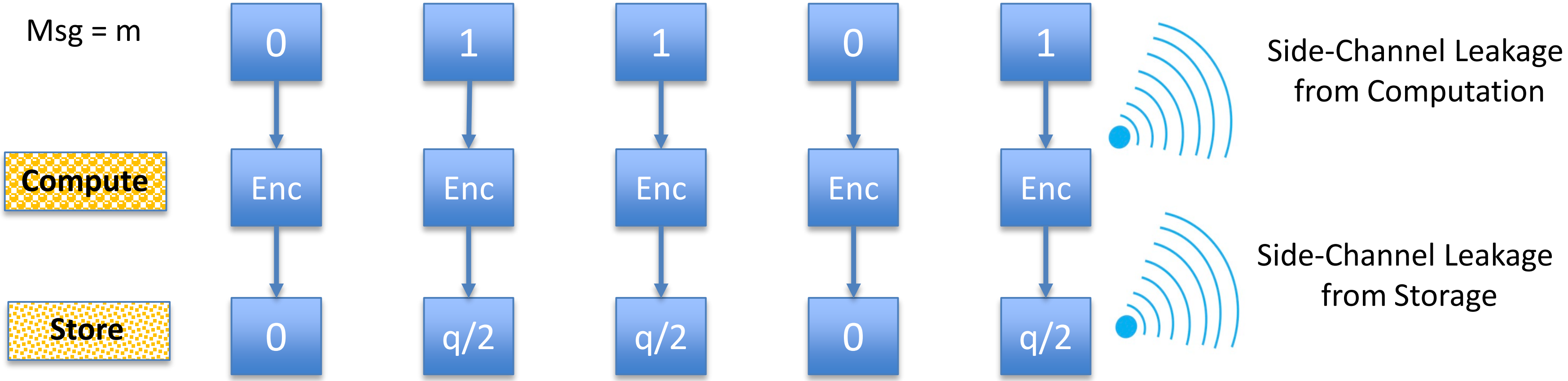


- **Decryption:**
  - **Input:** ct = (u, v), sk = s
  - **Output:** m



- Used to convert message to polynomial and vice versa.
- **Encode** and **Decode** - Unique for LWE/LWR-based PKE scheme
- **Bitwise** manipulation of the message.
- Does **bitwise manipulation** lead to **side-channel leakage?**

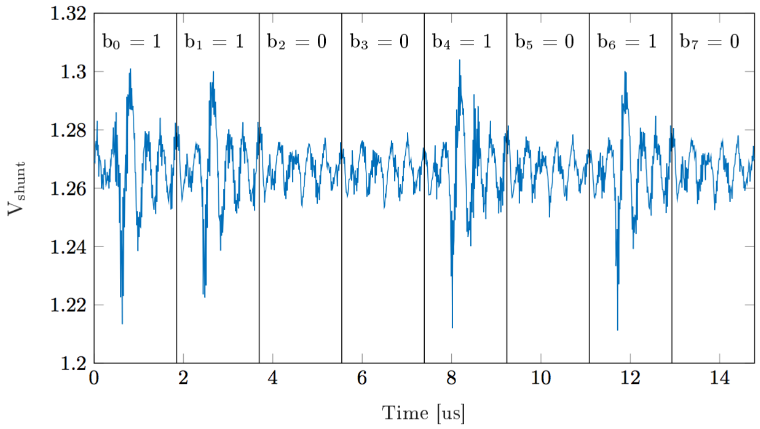
SCA of Message Encoding



- **Observation:** Only two possible types of operation for each bit – (**0 encoded to 0**) or (**1 encoded to q/2**)

SCA of Message Encoding

- Amiet et al. [ACLZ20] – Single trace message recovery attack on NewHope (Template Matching)



Single Side channel trace from message encoding Operation
NewHope – Unoptimized Impl. On ARM Cortex-M4
Source: Amiet et al. [12]

- Sim et al. [SKL+20] – Generalization of attack to multiple schemes (Kyber, SABER, Frodo, Round5, LAC)

[ACLZ20] Amiet, Dorian, Andreas Curiger, Lukas Leuenberger, and Paul Zbinden. "Defeating NewHope with a single trace." In *International Conference on Post-Quantum Cryptography*, pp. 189-205. Springer, Cham, 2020.
[SKL+20] Sim, Bo-Yeon, Jihoon Kwon, Joohee Lee, Il-Ju Kim, Tae-Ho Lee, Jaeseung Han, Hyojin Yoon, Jihoon Cho, and Dong-Guk Han. "Single-Trace Attacks on Message Encoding in Lattice-Based KEMs." *IEEE Access* 8 (2020): 183175-183191.

Defending against SCA of Message Encoding

- Idea 1: **Parallelize** the Encoding Procedure
  - Vectorization in HW/SW platforms.
  - Simultaneous leakage from multiple bits - Removes leakage from individual bits

- Idea 2: **Shuffle** the Order of Encoding (Sim et al.[SKL+20], Amiet et al. [ACLZ20])
  - Shuffle the order of processing of message bits
  - Can recover all message bits, but not the correct order.

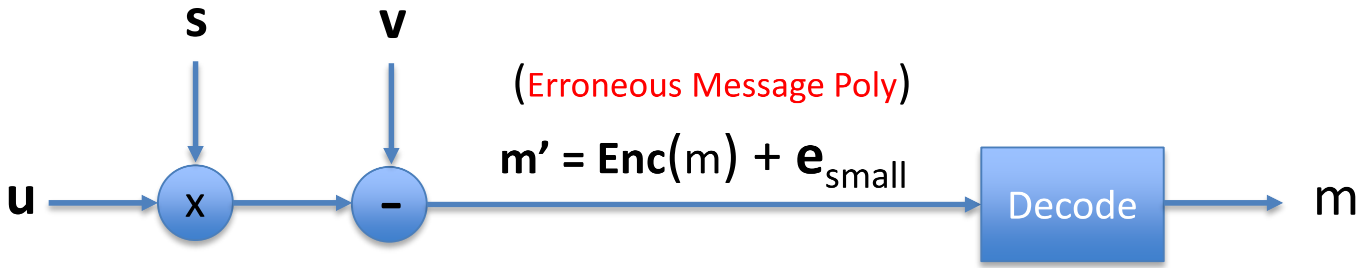
- But, do these techniques help thwart the attack???

- Ravi et al. [RBRC20] showed that “**Ciphertext Malleability**” in LWE/LWR-based PKEs can be used to defeat the aforementioned designs.

[RBRC20] Ravi, Prasanna, Shivam Bhasin, Sujoy Sinha Roy, Anupam Chattopadhyay. "On Exploiting Message Leakage in (few) NIST PQC Candidates for Practical Message Recovery and Key Recovery Attacks." Cryptology ePrint Archive, Report 2020/1559, 2020. https://eprint.iacr.org/2020/1559, 2020.
[ACLZ20] Amiet, Dorian, Andreas Curiger, Lukas Leuenberger, and Paul Zbinden. "Defeating NewHope with a single trace." In *International Conference on Post-Quantum Cryptography*, pp. 189-205. Springer, Cham, 2020.
[SKL+20] Sim, Bo-Yeon, Jihoon Kwon, Joohee Lee, Il-Ju Kim, Tae-Ho Lee, Jaeseung Han, Hyojin Yoon, Jihoon Cho, and Dong-Guk Han. "Single-Trace Attacks on Message Encoding in Lattice-Based KEMs." *IEEE Access* 8 (2020): 183175-183191.

Ciphertext Malleability in LWE/LWR-based PKE

- ❑ **Decrypt(ct = (u,v), sk = s) = m:**



$$\mathbf{u.s} = \underset{\text{Large}}{\cancel{\mathbf{a.s.s'}}} + \underset{\text{Small}}{\mathbf{e'.s}}$$

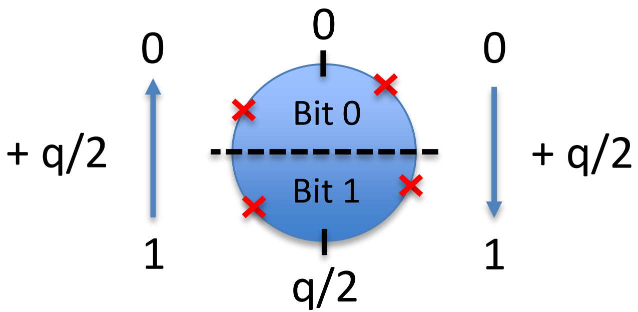
$$\begin{aligned} \mathbf{v} &= \mathbf{t.s'} + \mathbf{e''} + \mathbf{Enc}(m) \\ &= \mathbf{a.s.s'} + \mathbf{e.s'} + \mathbf{e''} + \mathbf{Enc}(m) \\ &= \underset{\text{Large}}{\cancel{\mathbf{a.s.s'}}} + \mathbf{Enc}(m) + \underset{\text{Small}}{\mathbf{e.s'} + \mathbf{e''}} \end{aligned}$$

- ❑ **Few Observations**:
  - ❑ Message polynomial only **additively hidden** within the ciphertext component **v**.
  - ❑ **No diffusion** of the message polynomial.
  - ❑ $m_i = f(v[i])$ - Each coefficient v[i] determines corresponding message bit m_i

Ciphertext Malleability in LWE/LWR-based PKE

- **Valid Ciphertext v:**
  - **v** = **t**.s' + **e''** + **Enc**(m)
- **Adding (q/2) to v[i]:**
  - **v'** = **v** + (q/2).x^i
  - With (**v'** – **u**.**s** = **m'**)
    m'[i] = **m**[i] + **e**[i] + q/2
  - m'_i = Decode(**m'**[i])
    = Flip(m'_i)

Decoding of m'[i]

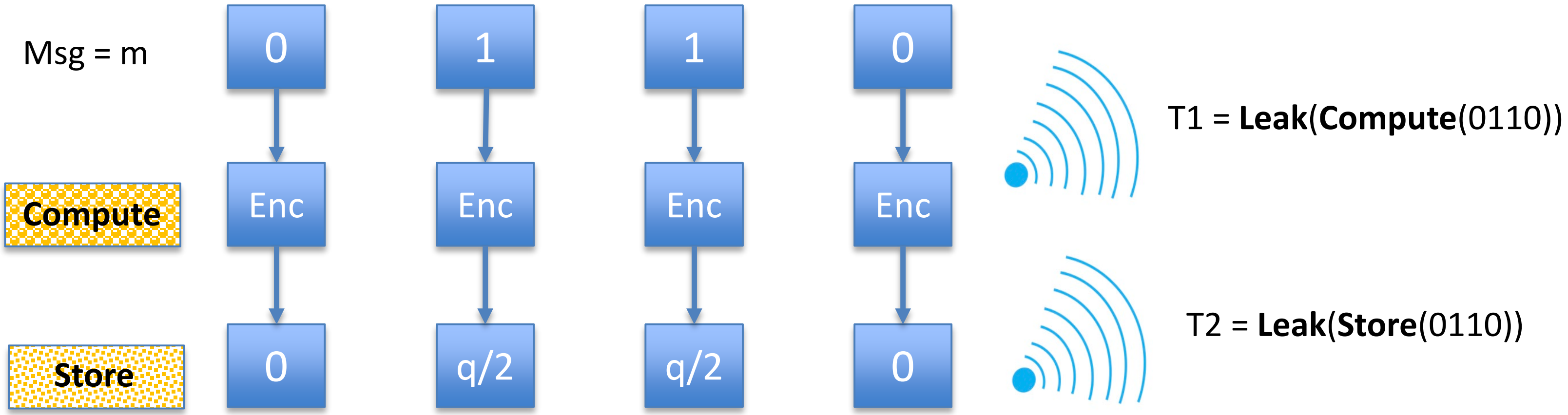


Malleability Property:
To flip m_i , add q/2 to v[i]

Ciphertext Malleability as a tool for SCA:

- **Idea 1: Parallelized** Encoding Procedure (x4)
  - **Step 1:** Query Decapsulation device with valid ct = (u,v)

v = **t**.**s'** + **e''** + **Enc**(m)

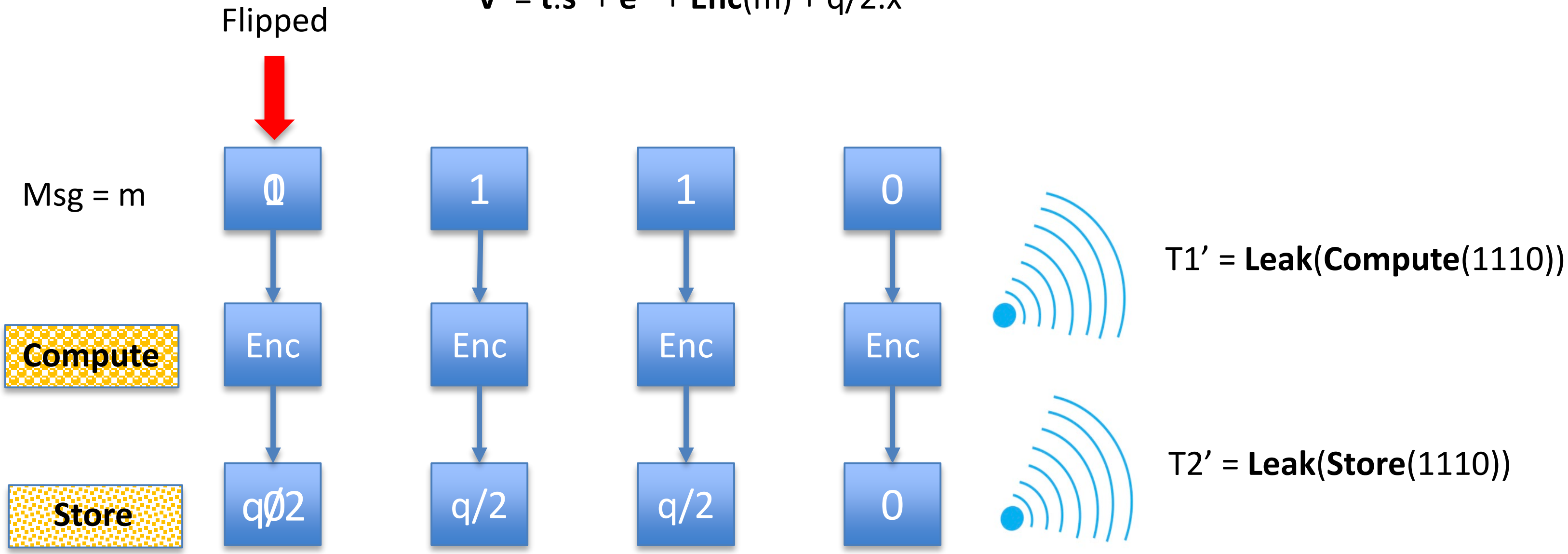


Message Encoding (Re-Encryption)

Ciphertext Malleability as a tool for SCA:

- **Idea 1: Parallelized** Encoding Procedure (x4)
  - **Step 2:** Modify **v** to construct **v'** and query **v'**

$$\mathbf{v'} = \mathbf{t}.s' + \mathbf{e''} + \mathbf{Enc}(m) + q/2.x^0$$



Message Encoding (Re-Encryption)

Ciphertext Malleability as a tool for SCA:

- **Idea 1: Parallelized** Encoding Procedure (x4)
  - **Step 3:** Compare the leakages T1 and T1**’** (resp. T2 and T2’)
    - If T1’ > T1, flip is from 0 to 1 => $m_0 = 0$
    - If T1’ < T1, flip is from 1 to 0 => $m_0 = 1$
    - Attack Simultaneously all nibbles of the message
  - If (x 4) parallelization, full message recovery in 5 traces.
  - If (x n) parallelization, full message recovery in (n+1) traces.

Ciphertext Malleability as a tool for SCA:

- **Idea 2: Shuffle** the order of Encoding of bits
  - **Step 1:** Query Decapsulation device with valid ct = (u,v)

$$\mathbf{v} = \mathbf{t}.s' + \mathbf{e''} + \mathbf{Enc}(m)$$

  - **Step 2:** Retrieve all the bits from leakage and let Hamming Weight(m') = X' (number of 1s)
  - **Step 3:** Modify **v** to construct **v'** and query **v'**

$$\mathbf{v'} = \mathbf{t}.s' + \mathbf{e''} + \mathbf{Enc}(m) + q/2.x^0$$

  - **Step 4:** Retrieve all the bits from leakage and let Hamming Weight(m'') = X''
  - **Step 5:** Compare X and X' to retrieve m0
    - If X'' = X' + 1, flip is from 0 to 1 => $m_i = 0$
    - If X'' = X' – 1, flip is from 1 to 0 => $m_i = 1$
  - If "k" bits in message, message recovery can be done in (k+1) traces.

Ciphertext Malleability as a tool for SCA:

- ❑ Can also be extended to **masked** implementations albeit with higher number of traces [RBRC20].
- ❑ Attack also applies to message decoding procedure in decryption [RBRC20].
- ❑ Protections increase attacker’s complexity, but do not prevent attack.
- ❑ **Shuffling** + **Masking** - Considered to be secure for message encoding and decoding operation.
- ❑ **Advantages**:
  - ❑ Very Effective (Full Message Recovery)
- ❑ **Disadvantages**:
  - ❑ Relatively high SNR required (Identify Precise Leakage Points, Distniguish single bit changes)
  - ❑ Attack can be made effective using more sophisticated setup (trace filtering, synchronization)
- ❑ **Leakage from Encoding/Decoding** + **“Ciphertext Malleability”** - Improved/Enhanced SCA for message recovery

[RBRC20] Ravi, Prasanna, Shivam Bhasin, Sujoy Sinha Roy, Anupam Chattopadhyay. "On Exploiting Message Leakage in (few) NIST PQC Candidates for Practical Message Recovery and Key Recovery Attacks." Cryptology ePrint Archive, Report 2020/1559, 2020. https://eprint.iacr.org/2020/1559, 2020.

Classification of SCA on LWE/LWR-based PKE/KEMs:

SCA
(Algorithmic Level)

Key Recovery (Chosen Ciphertext)

PC Oracle-based
D'Anvers et al. [DTVV19]
Ravi et al. [RRBC20]

FD Oracle-based
Xu et al. [XPRO20]
Ravi et al. [RBRC20]
Ngo et al. [NDGJ21]

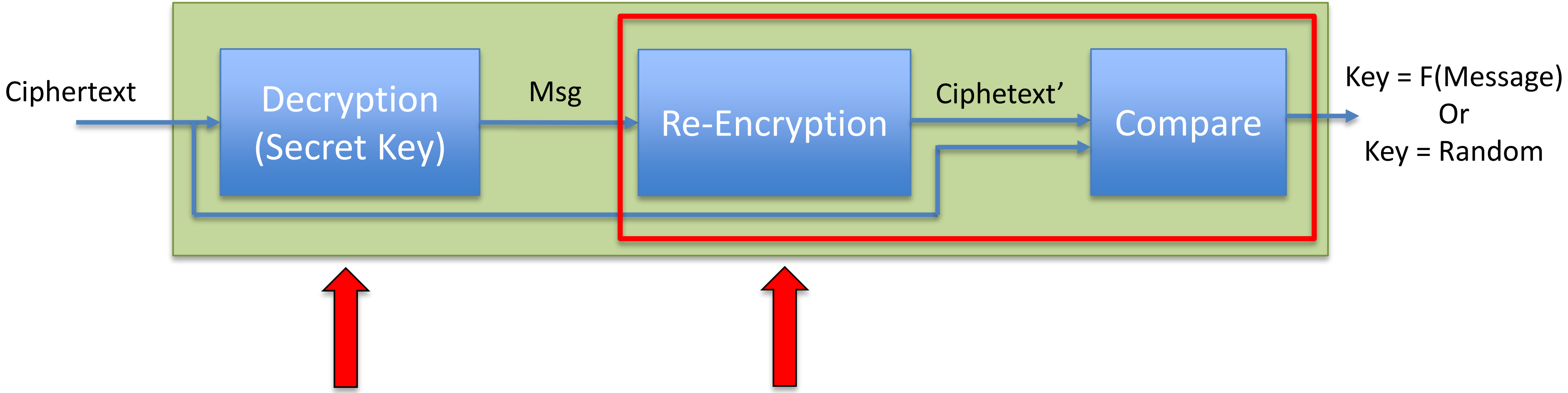
Message Recovery

Classification of SCA on LWE/LWR-based PKE/KEMs:

- **SCA (Algorithmic Level)**
  - **Key Recovery (Chosen Ciphertext)**
    - **PC Oracle-based**
      - D’Anvers et al. [DTVV19]
      - Ravi et al. [RRBC20]
    - **FD Oracle-based**
      - Xu et al. [XPRO20]
      - Ravi et al. [RBRC20]
      - Ngo et al. [NDGJ21]
  - **Message Recovery**
    - **Message Encoding**
      - Amiet et al. [ACLZ20]
      - Sim et al. [SKL+20]
    - **Message Decoding**
      - Ravi et al. [RBRC20]
      - Ngo et al. [NDGJ21]

Classification of SCA on LWE/LWR-based PKE/KEMs:

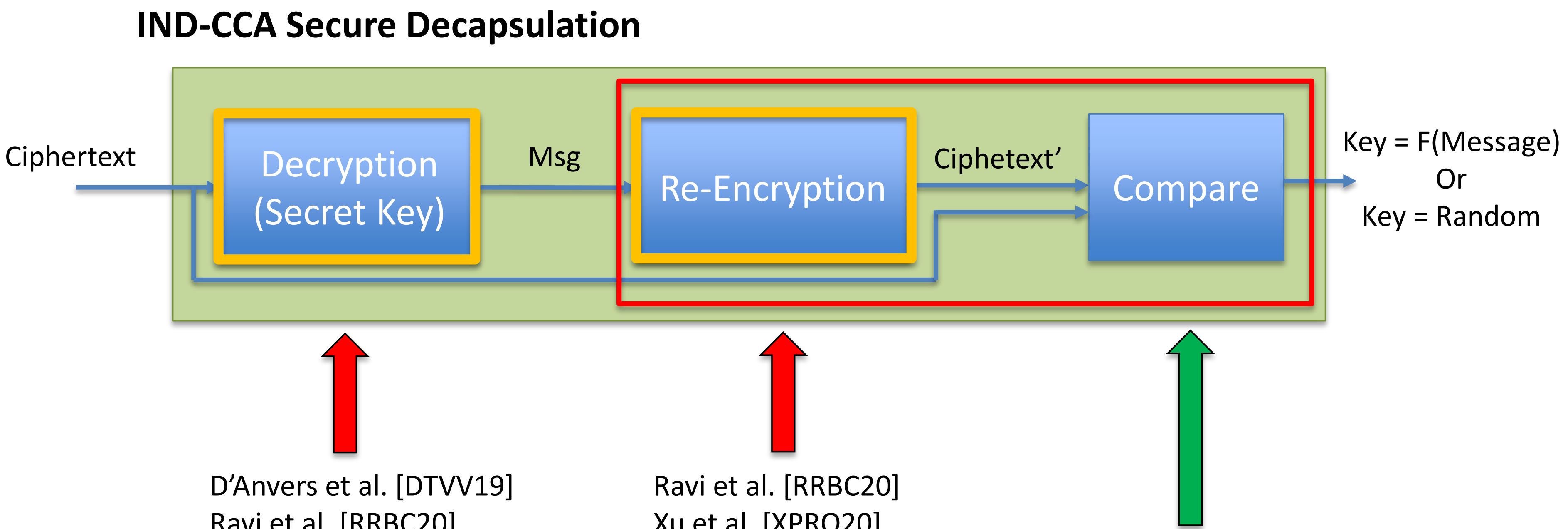
IND-CCA Secure Decapsulation



D'Anvers et al. [DTVV19]
Ravi et al. [RRBC20]
Ravi et al. [RBRC20]
Ngo et al. [NDGJ21]

Ravi et al. [RRBC20]
Xu et al. [XPRO20]
Sim et al. [SKL+20]
Amiet et al. [ACLZ20]

Defending against SCA on LWE/LWR-based PKE/KEMs:



D'Anvers et al. [DTVV19]
Ravi et al. [RRBC20]
Ravi et al. [RBRC20]
Ngo et al. [NDGJ21]

Ravi et al. [RRBC20]
Xu et al. [XPRO20]
Sim et al. [SKL+20]
Amiet et al. [ACLZ20]

Does it contain any sensitive information???

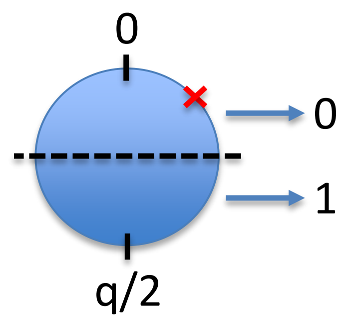
Analysis of Ciphertext Comparison:

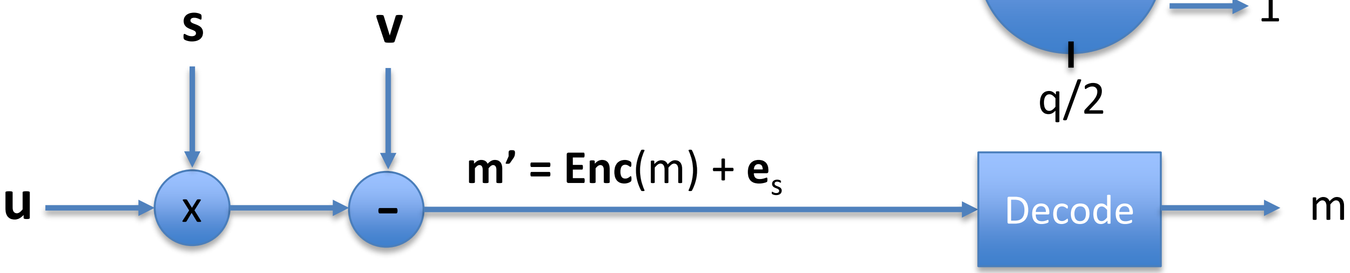
- For valid Ciphertexts ------ Comparison: **PASS**
- For Invalid Ciphertexts ------ Comparison: **FAIL**
- The comparison always fails for invalid ciphertexts (used in chosen ciphertext attacks)
- So, do we need to protect ciphertext comparison???
- **Revelation:** ”How comparison fails” leaks information about secret key (Guo et al. in [GJN20])
- **Decryption Failure Oracle-based SCA**

[GJN20] Qian Guo, Thomas Johansson, Alexander Nilsson. "A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM." https://eprint.iacr.org/2020/743 In IACR-CRYPTO 2020.

Decryption Failure (DF) Oracle-based SCA:

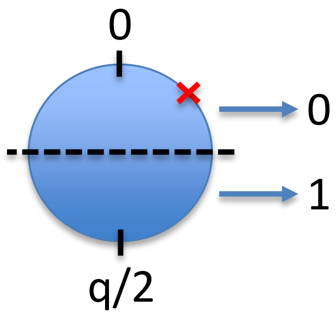
- **Modus Operandi**:
  - Construct a valid ciphertext ct = (**u**,**v**) for message m.



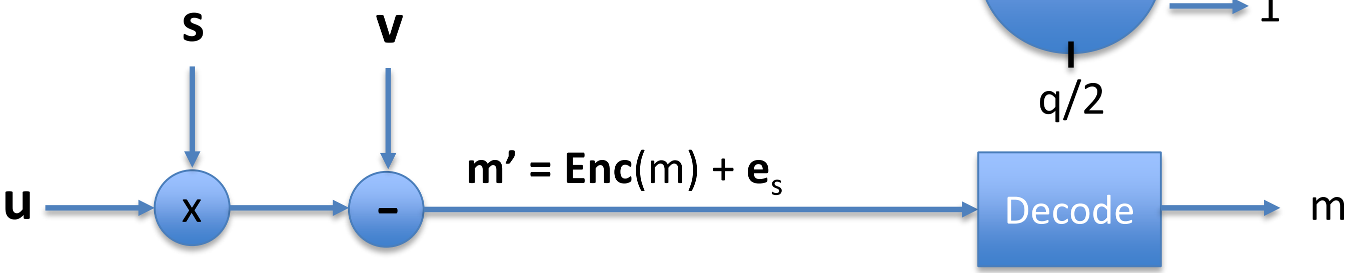


Decryption Failure (DF) Oracle-based SCA:

Decoding of **m’[i]**



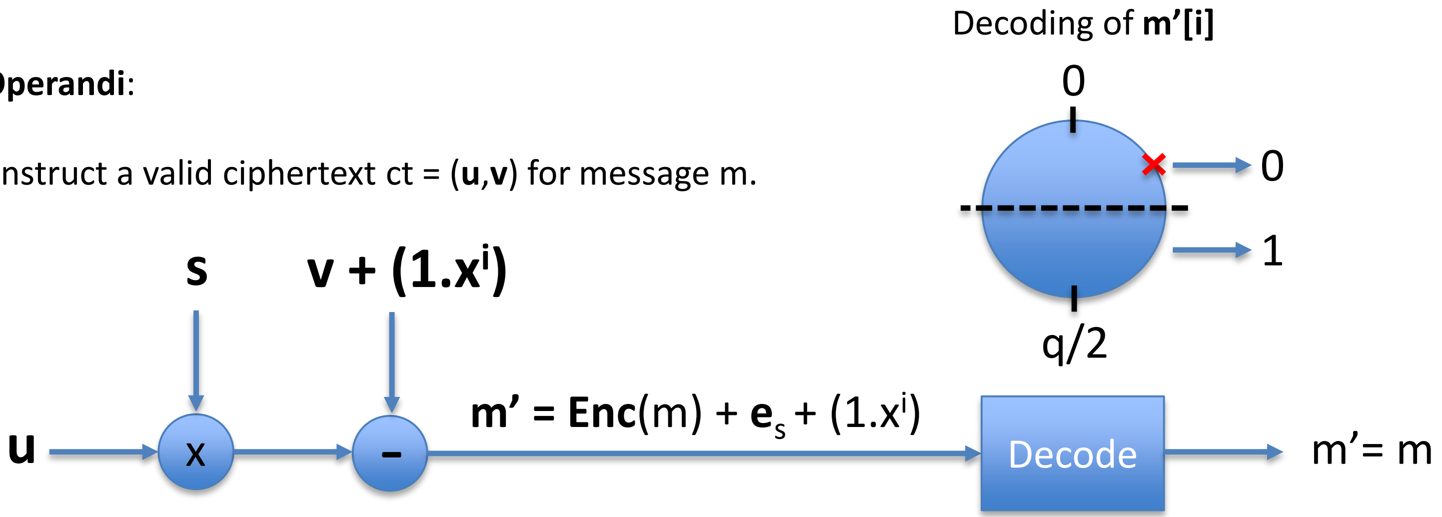
- **Modus Operandi**:
  - Construct a valid ciphertext ct = (**u**,**v**) for message m.



  - Add a small error to the i^{th} coefficient of **v** (v[i]) and observe change in the message m’.

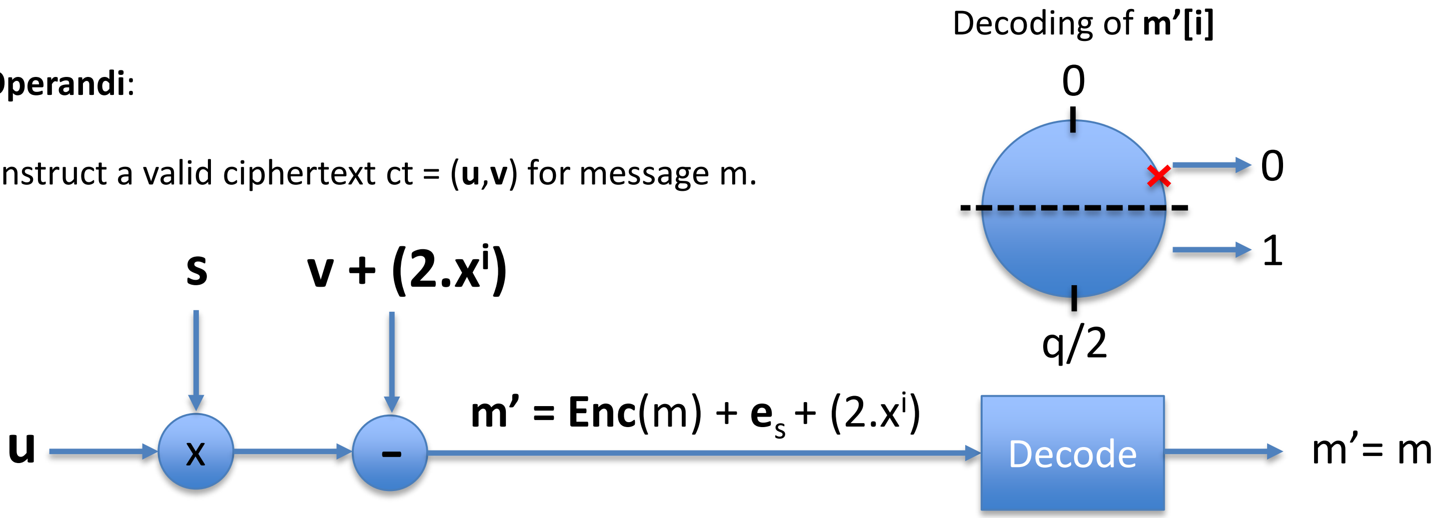
Decryption Failure (DF) Oracle-based SCA:

- ❑ **Modus Operandi**:
  - ❑ Construct a valid ciphertext ct = (**u**,**v**) for message m.



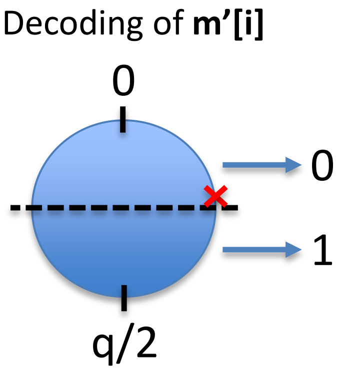
Decryption Failure (DF) Oracle-based SCA:

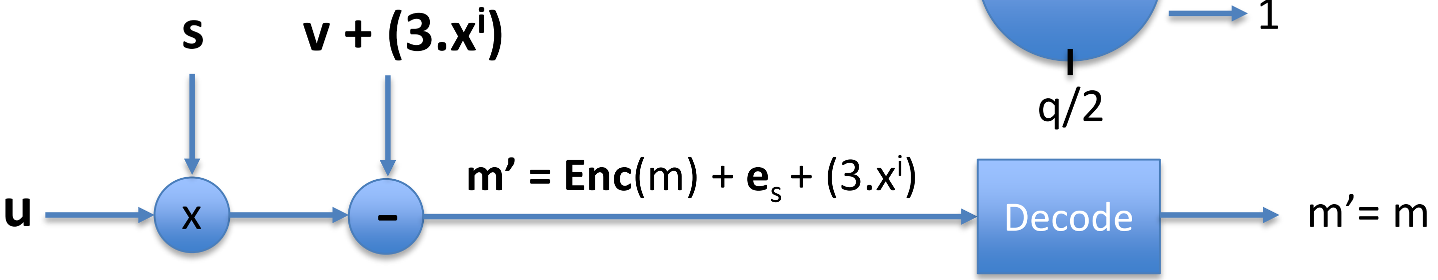
- **Modus Operandi**:
  - Construct a valid ciphertext ct = (**u**,**v**) for message m.



Decryption Failure (DF) Oracle-based SCA:

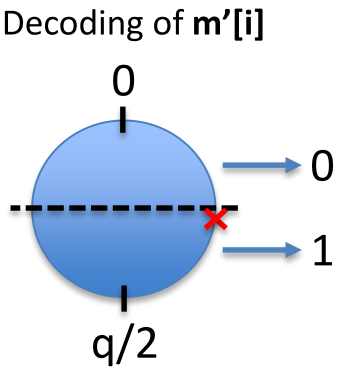
- **Modus Operandi**:
  - Construct a valid ciphertext ct = (**u**,**v**) for message m.

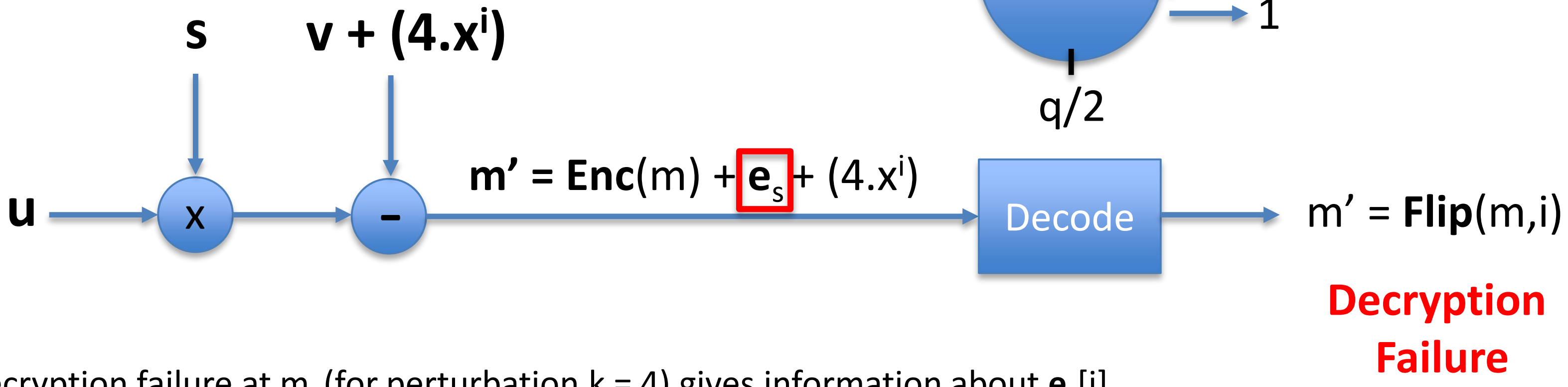




Decryption Failure (DF) Oracle-based SCA:

- **Modus Operandi**:
  - Construct a valid ciphertext ct = (**u**,**v**) for message m.

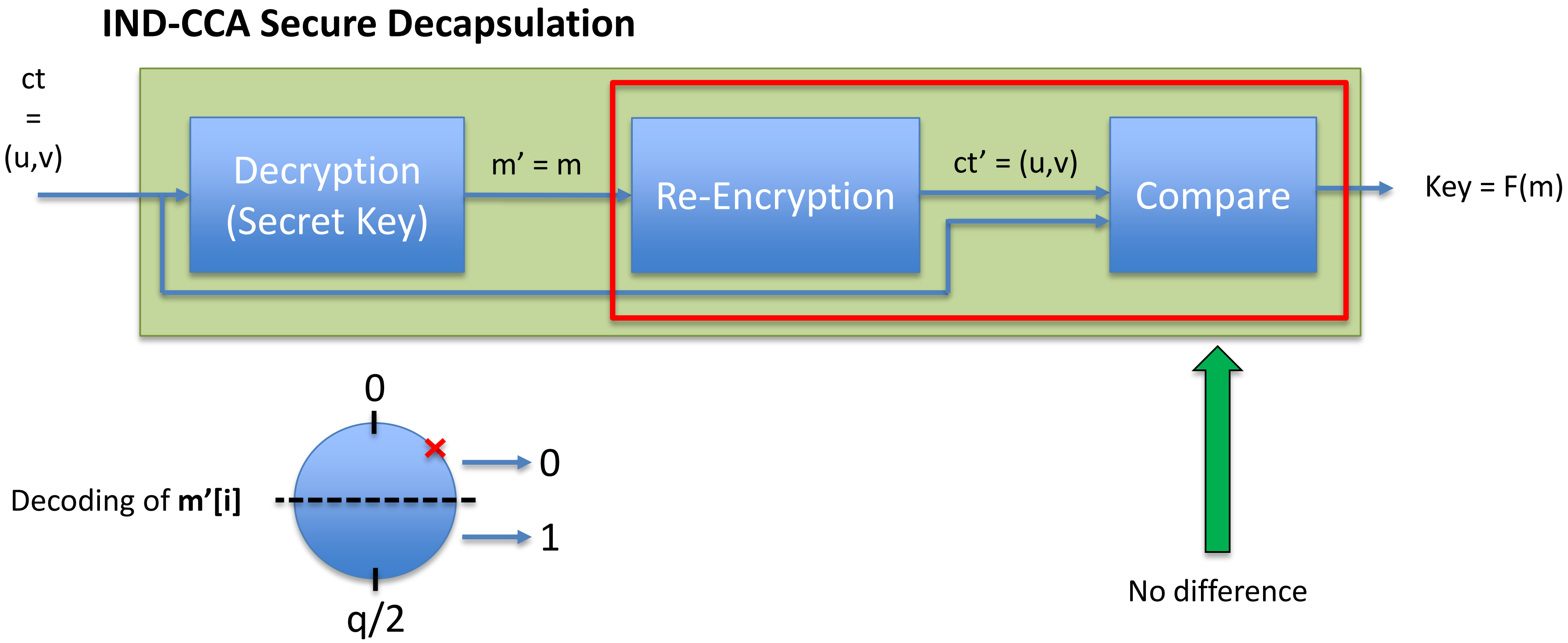




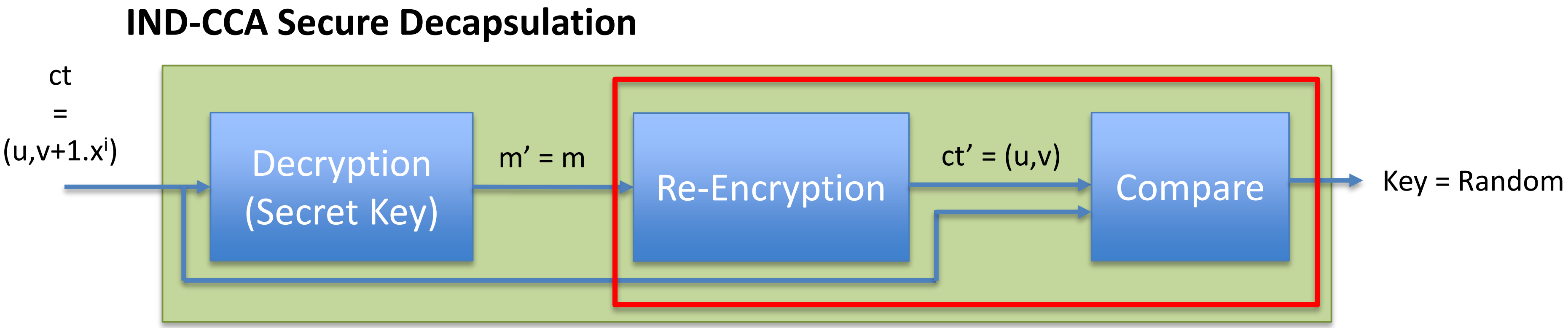
  - Decryption failure at m_i (for perturbation k = 4) gives information about $\mathbf{e_s}$[i]
  - $\mathbf{e_s}$ - **secret dependent** error polynomial
  - Attacker can obtain linear hints about secret through decryption failures
  - Enough number of hints potentially reveals the secret key.

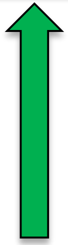
How does an attacker identify decryption failures through SCA???

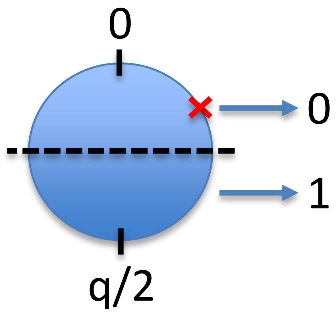
Decryption Failure (DF) Oracle-based SCA:



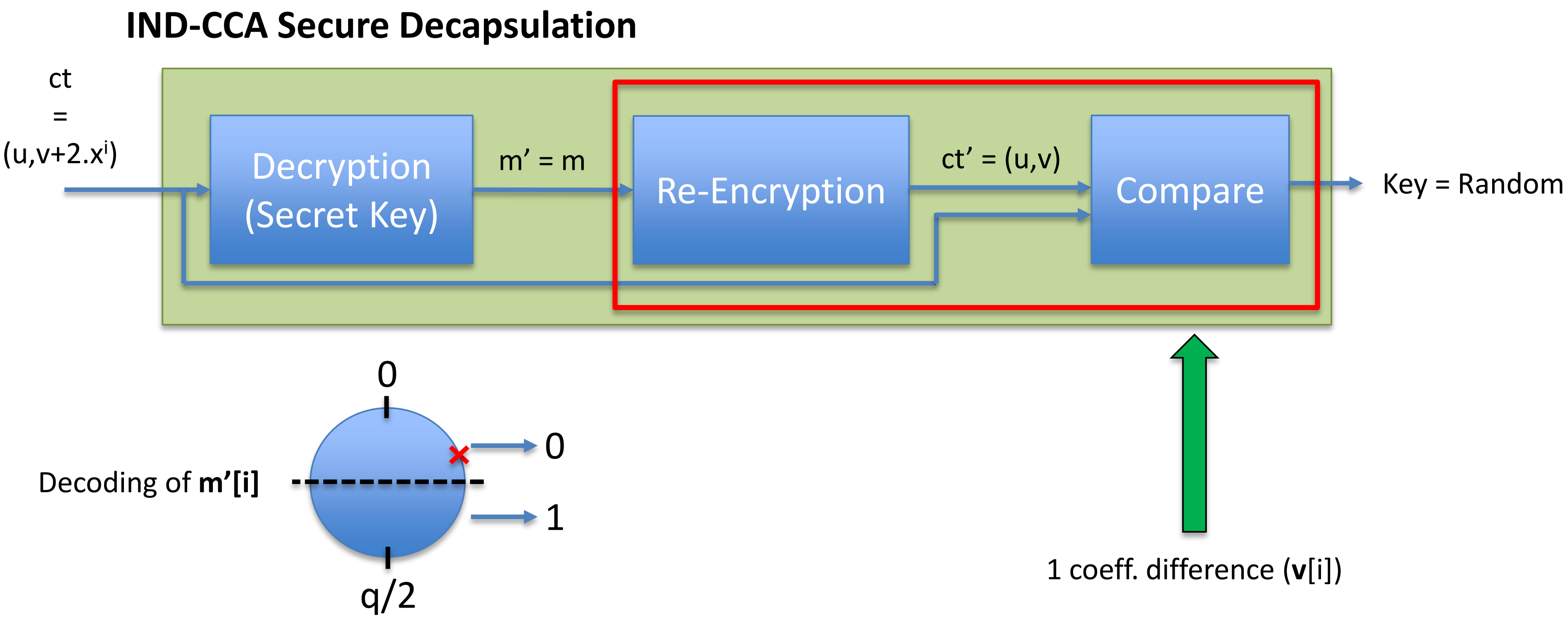
Decryption Failure (DF) Oracle-based SCA:



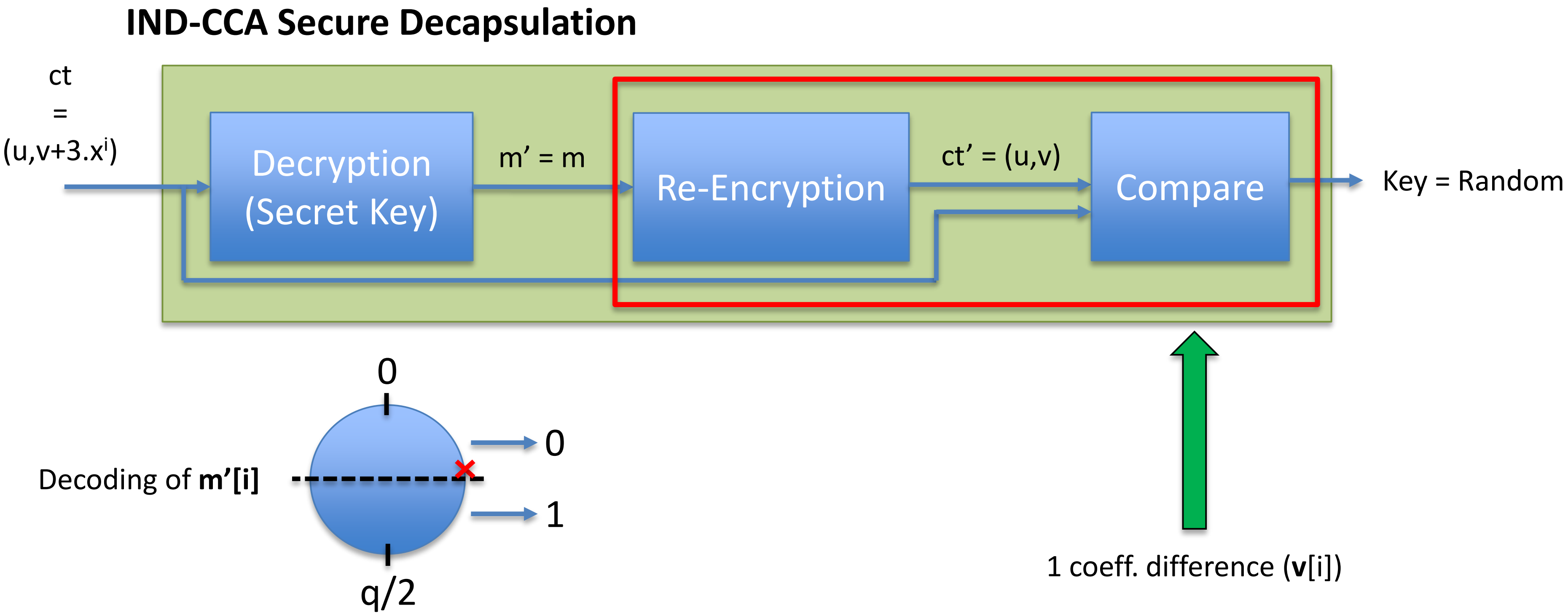




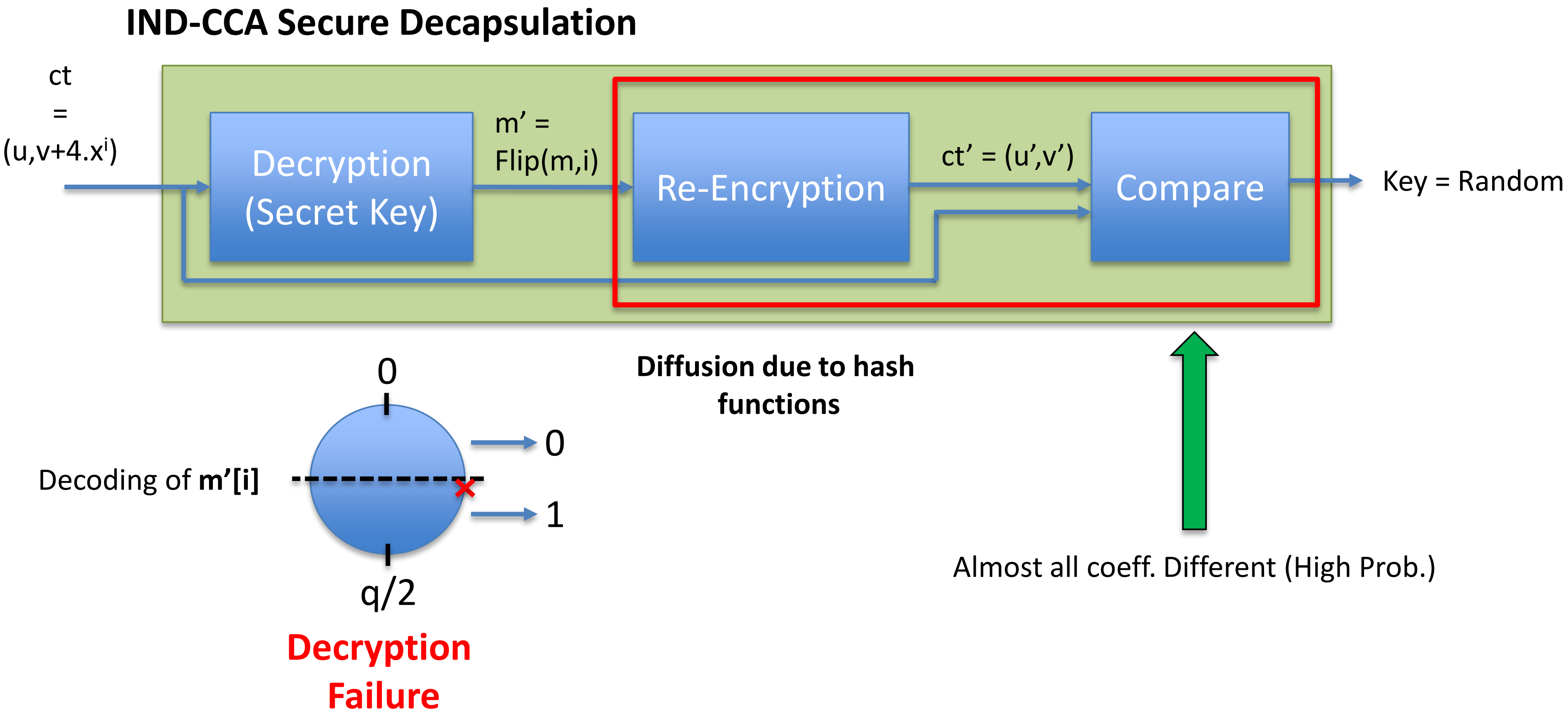
Decryption Failure (DF) Oracle-based SCA:



Decryption Failure (DF) Oracle-based SCA:

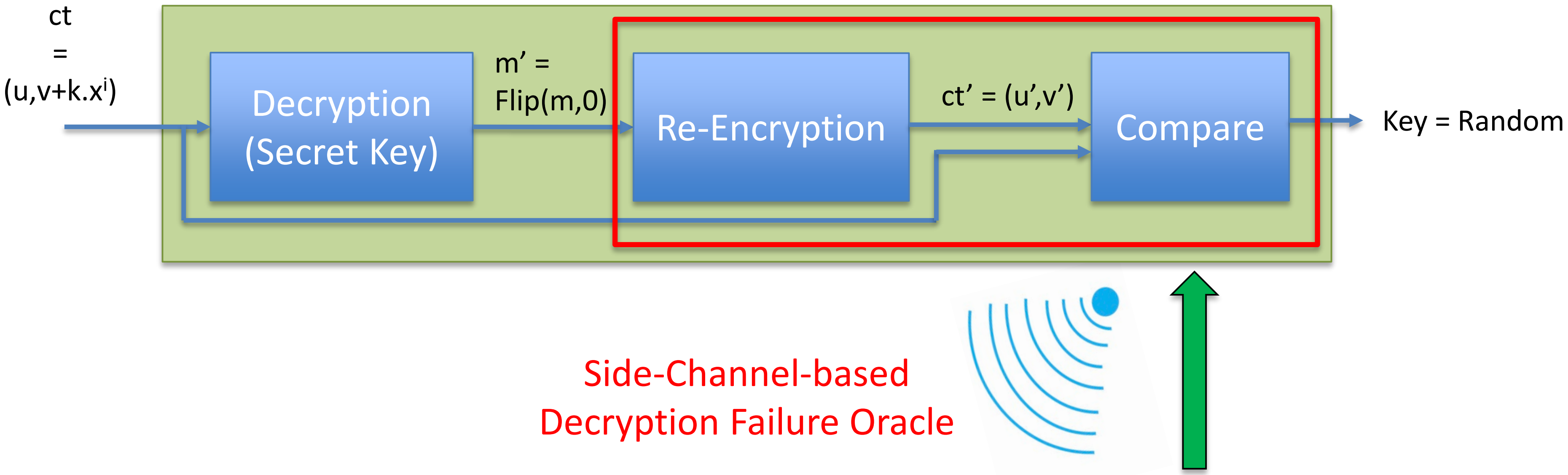


Decryption Failure (DF) Oracle-based SCA:



Decryption Failure (DF) Oracle-based SCA:

IND-CCA Secure Decapsulation



If (**No Decryption Failure**):
Only one coeff. Is different
If (**Decryption Failure**):
Almost All Coeffs. are different (High Prob.)

Decryption Failure (DF) Oracle-based SCA:

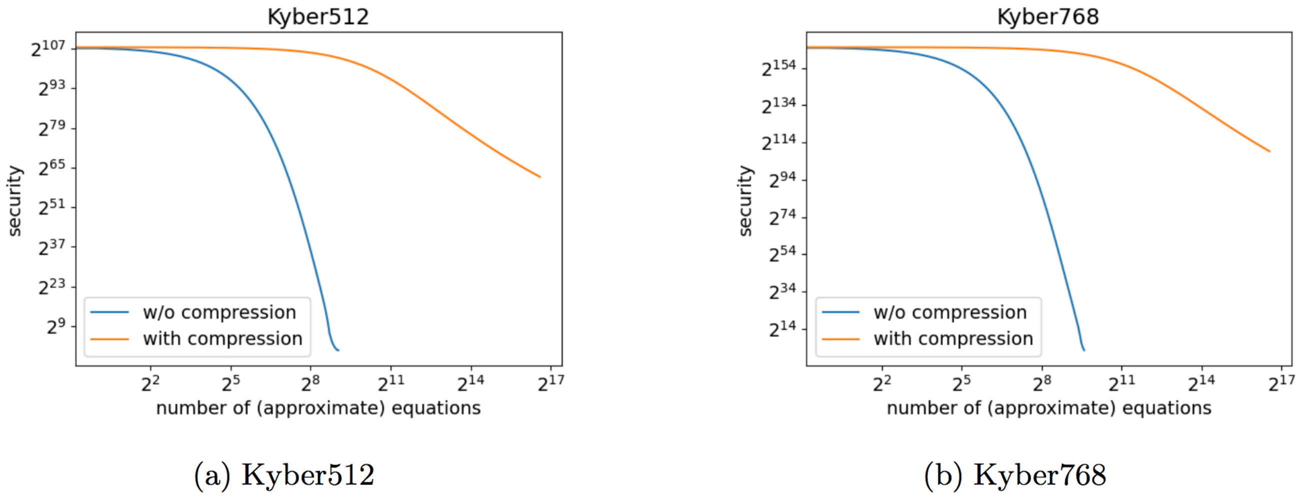
- Guo et al. [GJN20] presented the first DF oracle-based attack in SCA context on Frodo KEM:
  - **Timing Attack** on Non-Constant Time Comparison

    If(**Decryption Failure**)
    Comparison immediately aborts (Lesser Time)
    Else if(**No Decryption Failure**)
    Comparison only aborts at i^{th} coeff. (More Time)

  - 2^{30} decapsulation queries for full secret key recovery (incl. retries to get cleaner timing signal)
- Several approaches known for efficient masked ciphertext comparison (Oder et al. [OSPG18] and Bache et al. [BPO+20])
- For efficiency, they **unmask results of partial checks** (under notion that they are non-leaky).
- Unmasking result of partial checks leaks information about decryption failures - Bhasin et al. [BDH+21]

[OSPG18] Oder, Tobias, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. "Practical CCA2-secure and masked ring-LWE implementation." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018): 142-174.
[BPO+20] Bache, Florian, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu. "High-Speed Masking for Polynomial Comparison in Lattice-based KEMs." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020): 483-507.
[BDH+21] Bhasin, Shivam, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. "Attacking and Defending Masked Polynomial Comparison for Lattice-Based Cryptography."

Decryption Failure (DF) Oracle-based SCA:

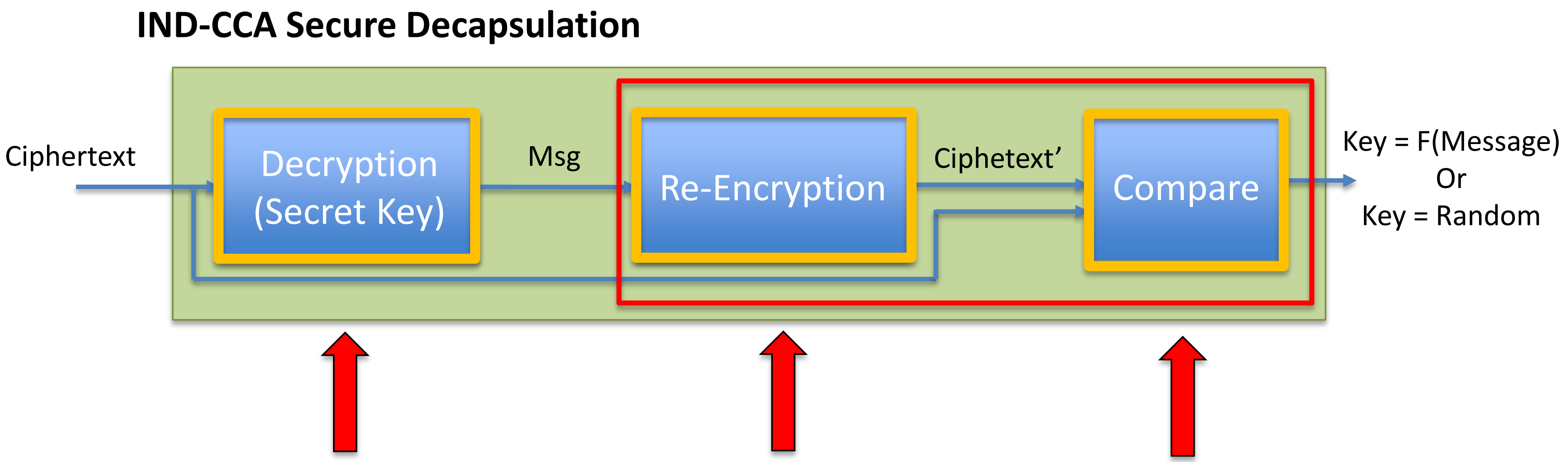


Security of Kyber512 and Kyber768 in function of the number of (approximate) equations retrieved.
Source: Bhasin et al. [BDH+21]

- ❑ **Takeaway:**
  - ❑ Implement **Constant-time Comparison**
  - ❑ Masked Implementation: **Do not unmask partial checks**

[BDH+21] Bhasin, Shivam, Jan-Pieter D’Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. "Attacking and Defending Masked Polynomial Comparison for Lattice-Based Cryptography."

Classification of SCA of LWE/LWR-based PKE/KEMs:



D'Anvers et al. [DTVV19]
Ravi et al. [RRBC20]
Ravi et al. [RBRC20]
Ngo et al. [NDGJ21]

Ravi et al. [RRBC20]
Xu et al. [XPRO20]
Sim et al. [SKL+20]
Amiet et al. [ACLZ20]

Guo et al. [GJN20]
Bhasin et al. [BDH+21]

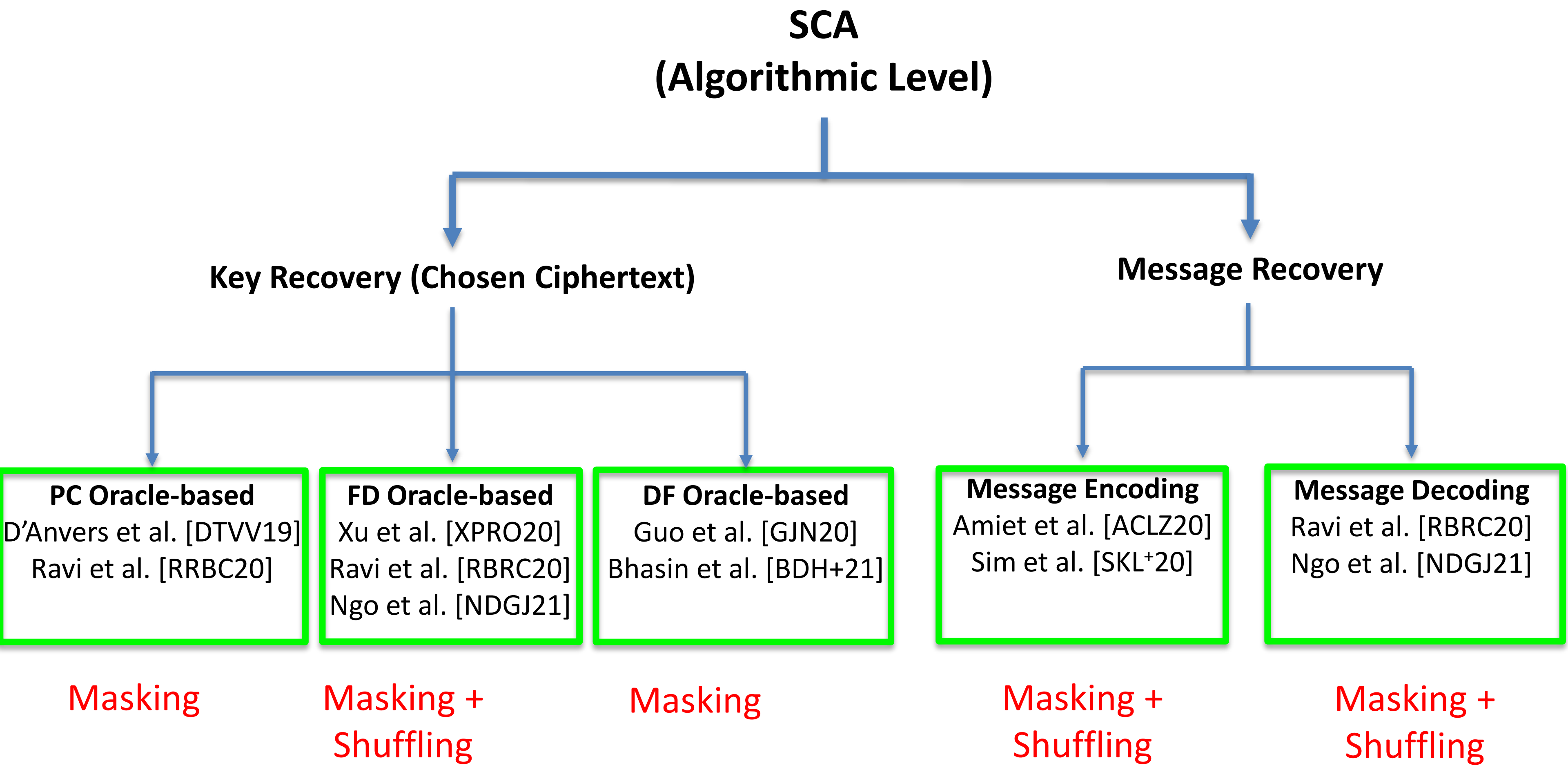
Classification of SCA of LWE/LWR-based PKE/KEMs:

SCA
(Algorithmic Level)

- **Key Recovery (Chosen Ciphertext)**
  - **PC Oracle-based**
    - D’Anvers et al. [DTVV19]
    - Ravi et al. [RRBC20]
  - **FD Oracle-based**
    - Xu et al. [XPRO20]
    - Ravi et al. [RBRC20]
    - Ngo et al. [NDGJ21]
  - **DF Oracle-based**
    - Guo et al. [GJN20]
    - Bhasin et al. [BDH+21]
- **Message Recovery**
  - **Message Encoding**
    - Amiet et al. [ACLZ20]
    - Sim et al. [SKL+20]
  - **Message Decoding**
    - Ravi et al. [RBRC20]
    - Ngo et al. [NDGJ21]

More Efficient

Classification of SCA of LWE/LWR-based PKE/KEMs:



Outline

- Background:
    - Learning With Error (LWE) Problem
    - LWE/LWR-based PKE framework
- Overview of side-channel attacks:
    - Algorithmic-level
    - Implementation-level
- **Overview of masking countermeasures:**
- Conclusions and future works:

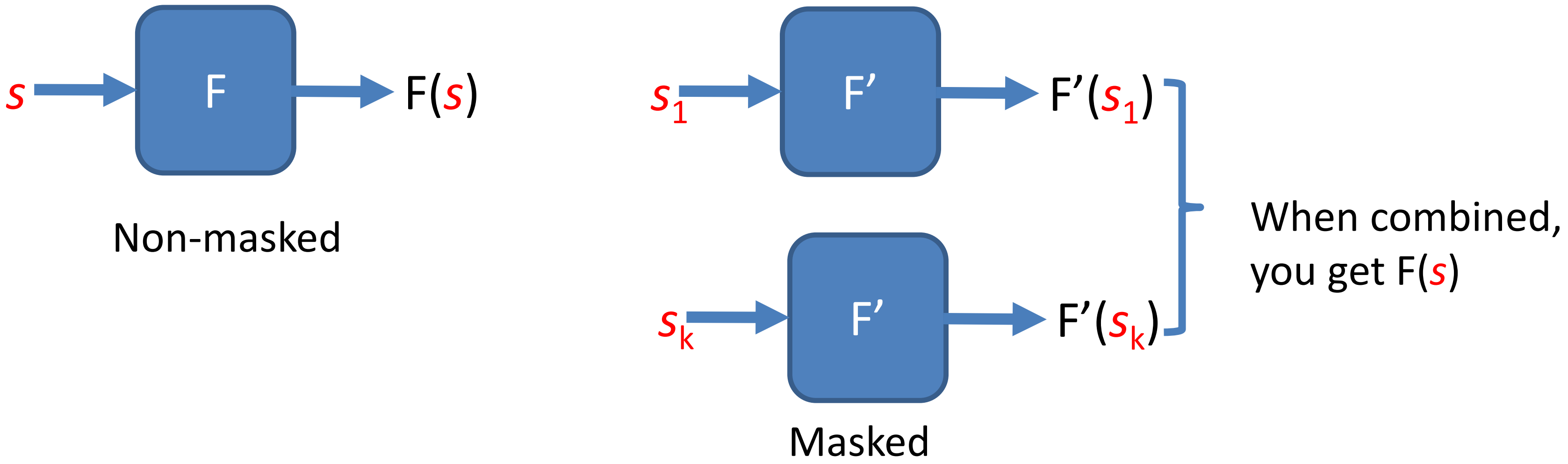
What is masking countermeasure?

- Countermeasure against differential power analysis (DPA)

- Randomizes computation by splitting secret data into random shares

$$s = s_1 + s_2 + s_3 + \ldots + s_k$$

- No information about s can be obtained by observing a proper subset



Arithmetic and Boolean shares

- Two common ways of splitting a secret into shares
- Boolean shares: secret bit is split in GF(2)

$$s = s_1 \oplus s_2 \oplus s_3 \oplus \ldots \oplus s_k \quad \text{mod } 2$$

… applicable to words (vector of bits)

- Arithmetic shares: secret is split in GF(p) where p>2

$$s = s_1 + s_2 + s_3 + \ldots + s_k \quad \text{mod } p$$

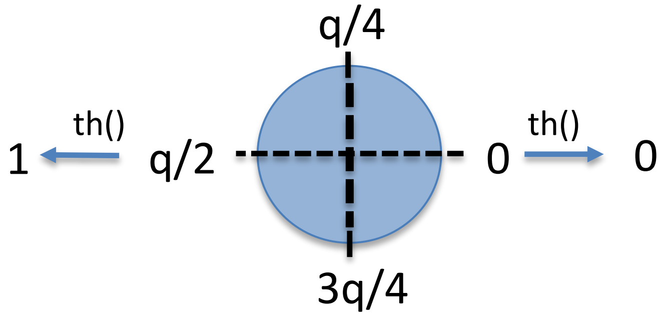
E.g., $7 = 8 + 10 \text{ mod } 11$

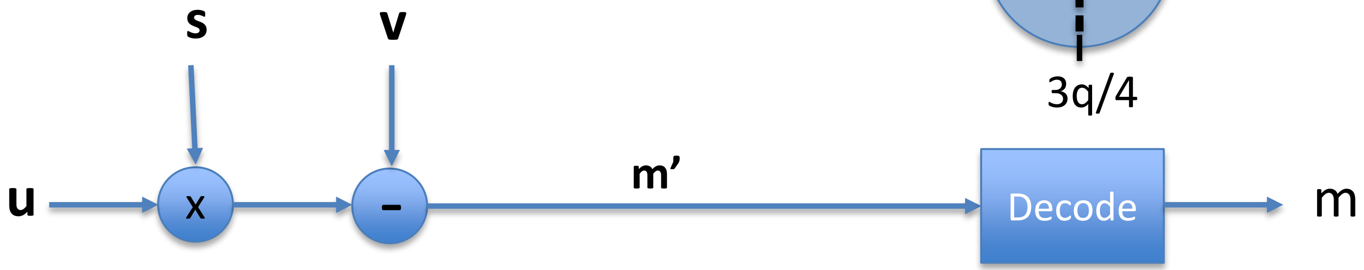
- Some cryptographic algorithms require working with both types

How to apply masking to lattice-based PKE?

Ring LWE-based PKE (IND-CPA)

- **Decryption:**
  - **Input:** ct = (**u**, **v**), sk = **s**
  - **Output:** m





$$\mathbf{m'} = \mathbf{v} - \mathbf{u.s} = \mathbf{Enc}(m) + \mathbf{e}_{\text{small}}$$

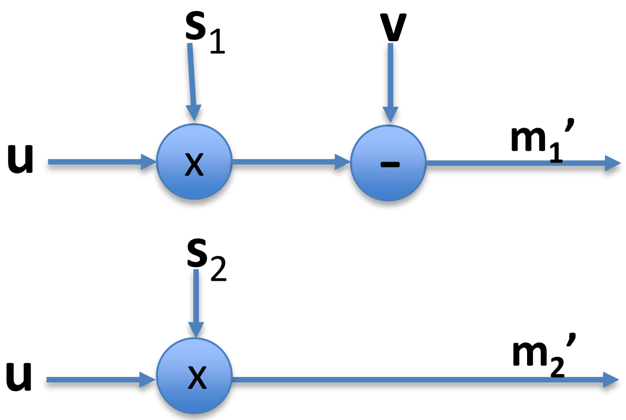
Note: ct = (**u**, **v**) is controlled by attacker

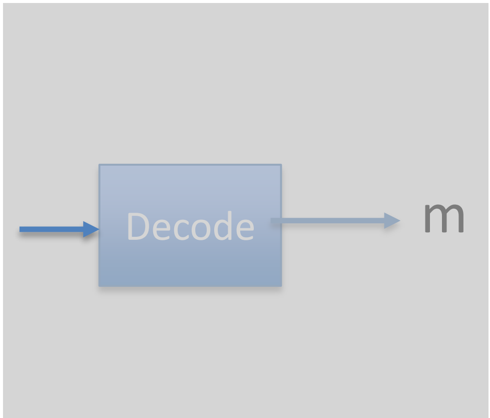
Masking Idea: Split **s** into random shares and randomize computation

1st Order Masking for IND-CPA PKE

- Step1: Split **s** into two arithmetic shares

$$\mathbf{s} = \mathbf{s}_1 - \mathbf{s}_2 \bmod q$$



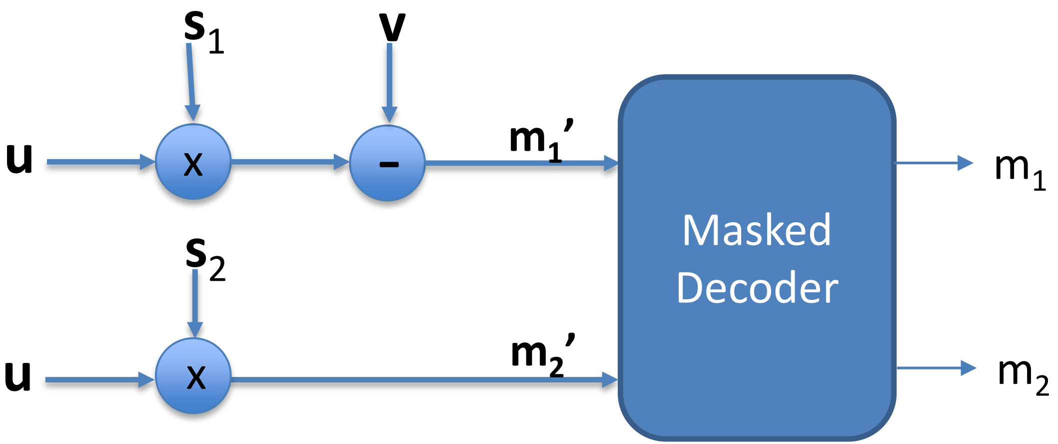


$\mathbf{m_1' = v - u.s_1}$
$\mathbf{m_2' = u.s_2}$

How to compute decoding on two shares?

Easy to check $\mathbf{m_1' + m_2' = v - u.s = m'}$

Masked Decoding



What we want:

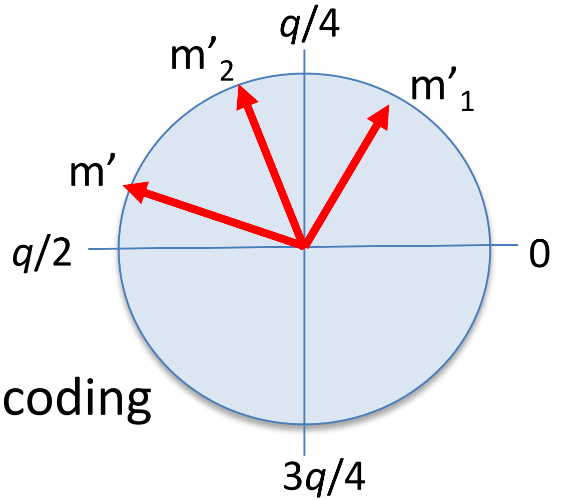
1. Compute mask-message pair (m_1, m_2) s.t. $m = m_1 + m_2 \bmod 2$
2. No combination of the two input shares $\mathbf{m_1'}$ and $\mathbf{m_2'}$

There are several approaches to design masked decoders

Masked Decoder of [RRVV15]

- Observation: Only a few most significant bits of the shares are helpful to perform threshold decoding

- Example:
  If $0 < m'_1 < q/4$ and $q/4 < m'_2 < q/2$
  then $q/4 < m' < 3q/2$
  $\rightarrow$ th(m') = 1



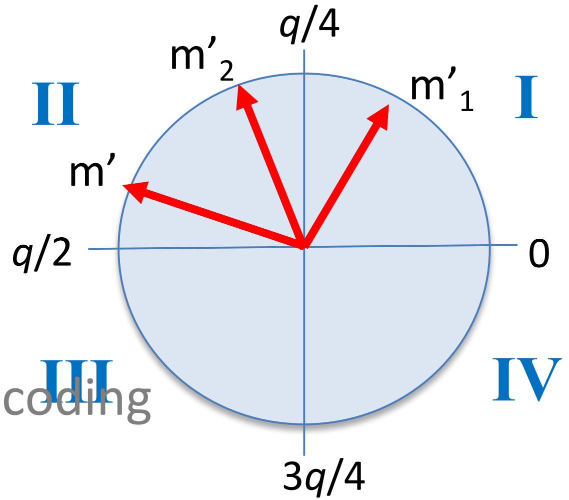
- This observation is used to simplify masked decoding

[RRVV15] O. Reparaz, S. S. Roy, F. Vercauteren, I. Verbauwhede. "A Masked Ring-LWE Implementation". CHES 2015.

Masked Decoder of [RRVV15]

- Observation: Only a few most significant bits of the shares are helpful to perform threshold decoding

- Example:
  If $0 < m'_1 < q/4$ and $q/4 < m'_2 < q/2$
  then $q/4 < m' < 3q/2$
  → $th(m') = 1$



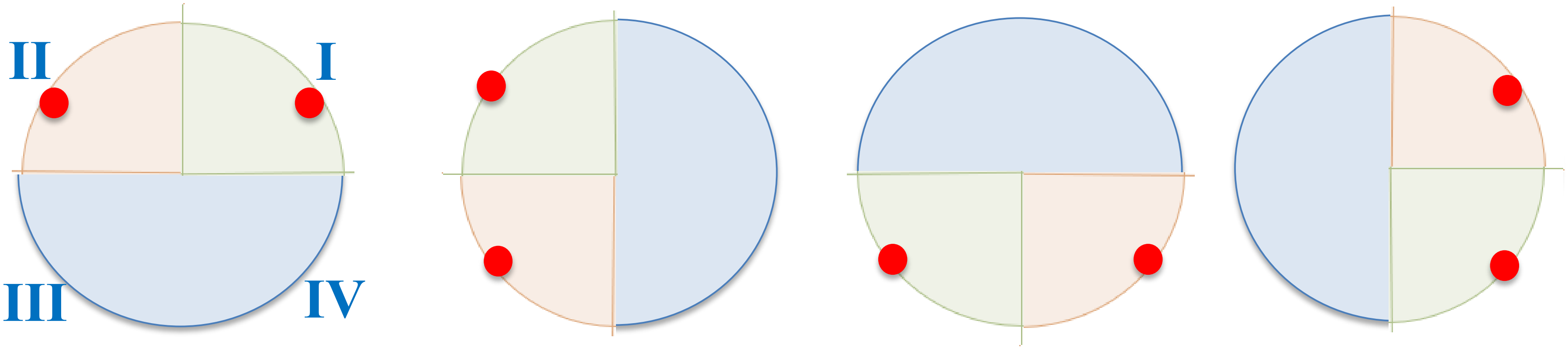
- This observation is used to simplify masked decoding

quad() function is used to output quadrant of a share.

[RRVV15] O. Reparaz, S. S. Roy, F. Vercauteren, I. Verbauwhede. "A Masked Ring-LWE Implementation". CHES 2015.

Masked Decoder of [RRVV15]

Quad-based decoding **works** if two shares are in adjacent quadrants.



Otherwise, this approach fails.

Solution proposed in [RRVV15]: Refresh shares and try again.

1. Take a constant δ_i from a table
2. $m'_1 := m'_1 - \delta_i$
3. $m'_2 := m'_2 + \delta_i$
4. Check if they are in adjacent quadrants

Iterated a fixed number of times

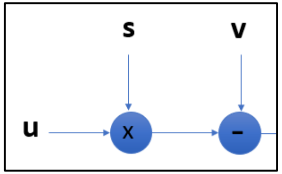
Results: Masked ring-LWE PKE (IND-CPA) [RRVV15]

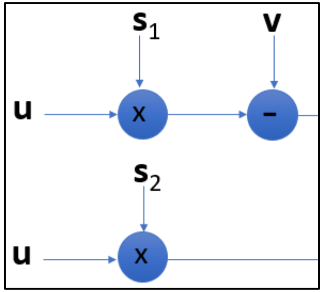
Masking overhead: ~2.7 times more cycles in HW (FPGA)
~5.8 times more cycles in SW (ARM M4

Decryption failure increases.

Reasons behind increased computation time:

1. Polynomial arithmetic cost doubles





2x computation

Unprotected vs Protected

2. Iterative ‘quad-based decoding’ increases the cost further

More Efficient Masked Decoder by [OSPG18]

- *Assume* that m'_1 and m'_2 are Boolean shares (instead of arithmetic)

  i.e., $m' = m'_1 \oplus m'_2$

- Naturally, $\mathrm{MSb}(m') = \mathrm{MSb}(m'_1) \oplus \mathrm{MSb}(m'_2)$

- Hence, $\mathrm{th}(m') = \mathrm{th}(m'_1) \oplus \mathrm{th}(m'_2)$

→ Masked decoding becomes an easy operation in this setting

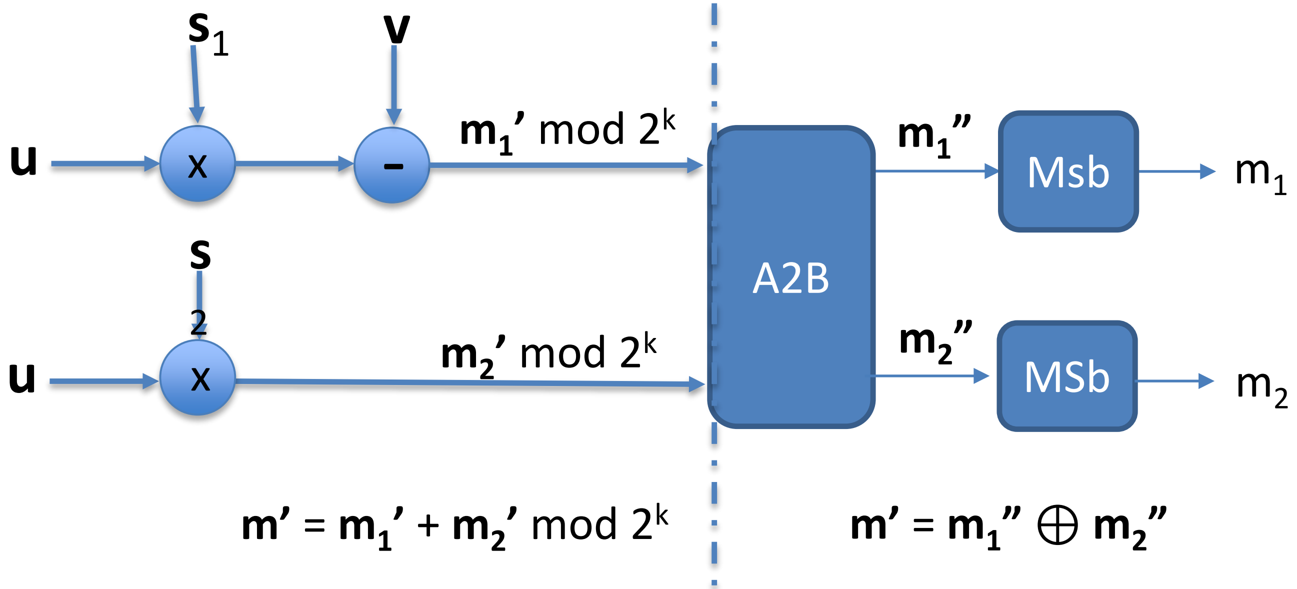
Can we realize this for ring/mod LWE/LWR?

Idea in [OSPG18]: Arithmetic to Boolean conversion (A2B)

[OSPG18] T. Oder, T. Schneider, T. Pöppelmann, T. Güneysu.
"Practical CCA2-Secure and Masked Ring-LWE Implementation". TCHES 2018

Masked Decoder with A2B approach [OSPG18]

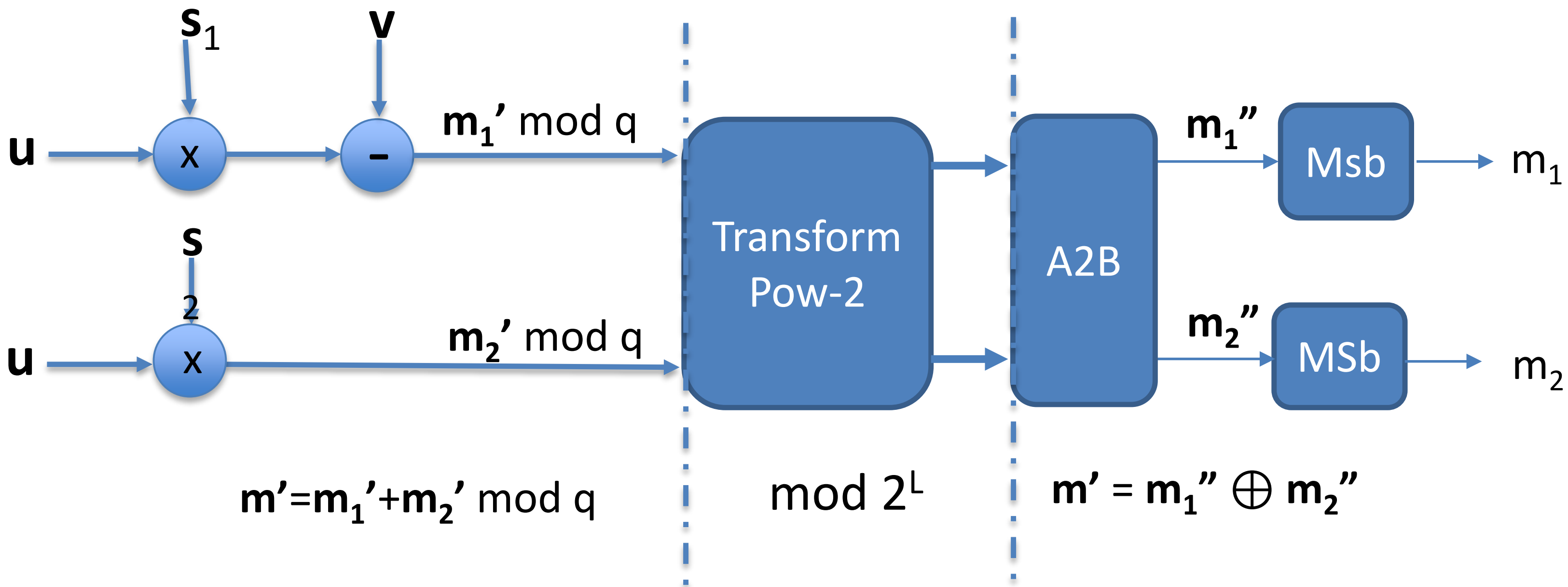
Assume that $\mathbf{m_1'}$ and $\mathbf{m_2'}$ are in (mod 2^k) for some k



A2B requires inputs to be modulo power-of-2

Masked Decoder with A2B approach [OSPG18]

Assume that $\mathbf{m_1'}$ and $\mathbf{m_2'}$ are in (mod q) where $q \neq 2^k$



An additional block “Transform-Power-of-2” is needed [OSPG18]

[OSPG18] T. Oder, T. Schneider, T. Pöppelmann, T. Güneysu.
"Practical CCA2-Secure and Masked Ring-LWE Implementation". TCHES 2018

Masking implementation: Case study for Saber KEM

- Saber uses Module LWR problem

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \\ \vdots \\ e_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ \vdots \\ b_m \end{pmatrix} \text{ mod q}$$

LWE

-vs-

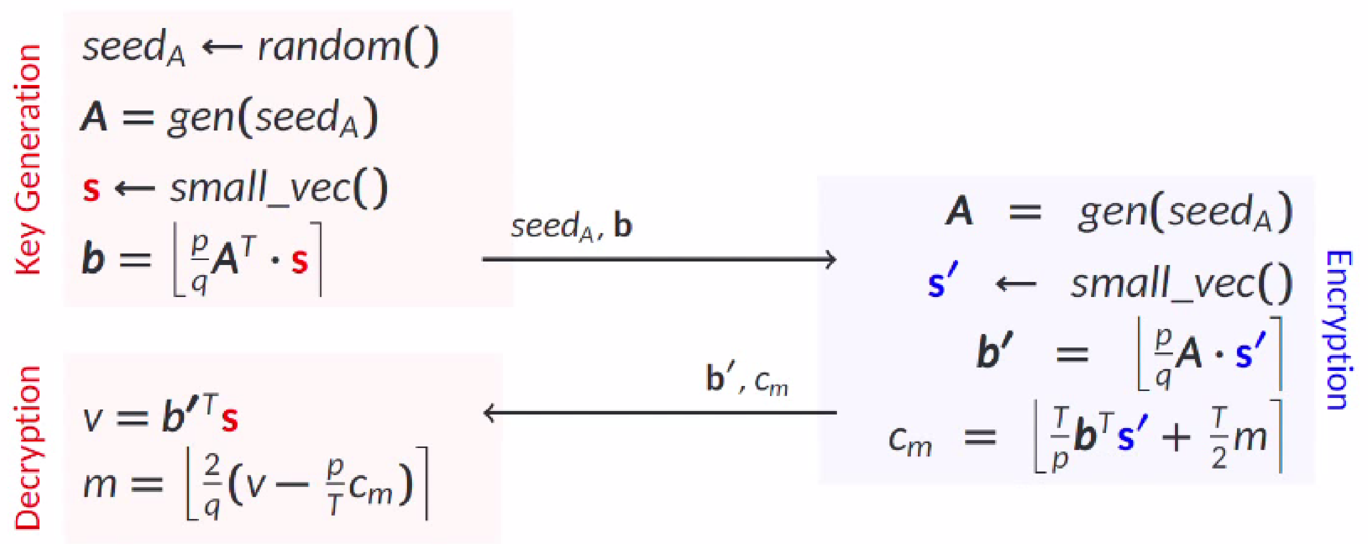
$$\left\lfloor \frac{p}{q} \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} \right\rceil = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ \vdots \\ b_m \end{pmatrix} \text{ mod p}$$

LWR

- No explicit noise generation.

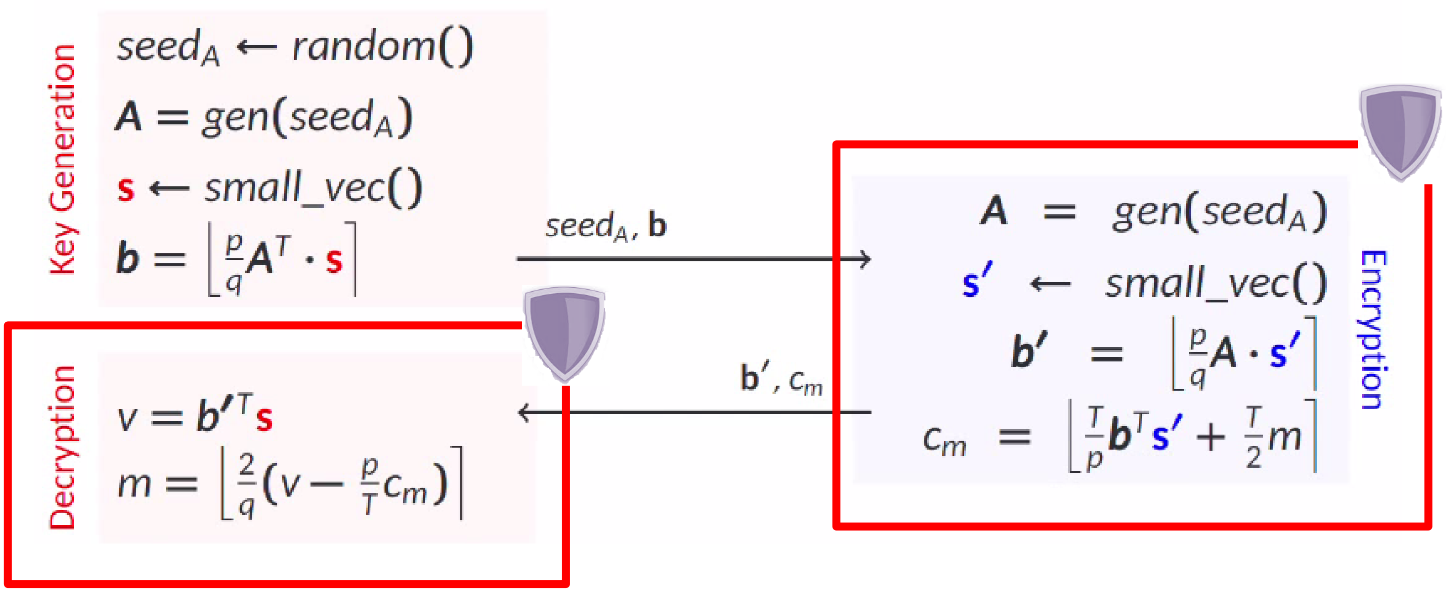
- Saber uses power-of-2 moduli $p=2^{10}$ and $q=2^{13}$
  → Rounding becomes bit-shift

Saber protocol



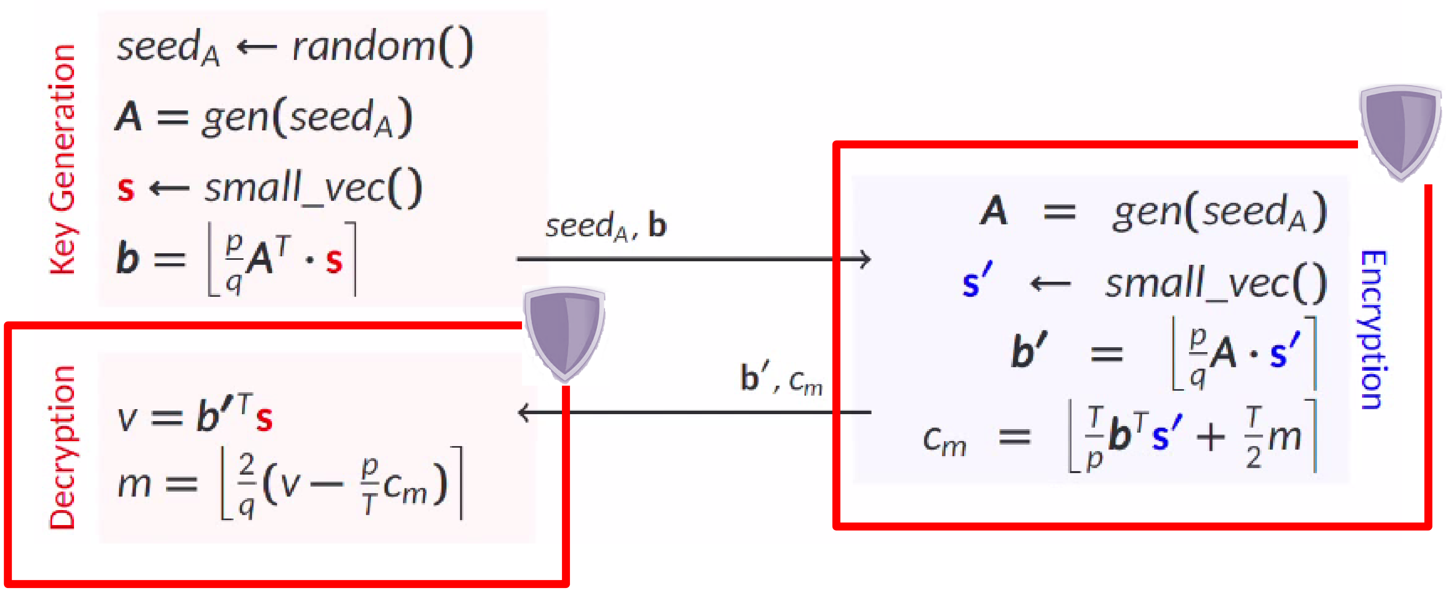
Saber.KEM is obtained via the Fujisaki-Okamoto transform.

Saber KEM with masking



Masking of **decryption + re-encryption + ct comparison**

Saber KEM with masking



Masking of **decryption + re-encryption + ct comparison**

Building blocks that should be protected:

- Polynomial addition and multiplication
- Rounding (i.e., bit-shifting)
- Keccak-based functions: SHA, SHAKE
- Binomial sampling
- Comparison of ciphertexts

Masking of rounding in Saber

- As *p* and *q* are powers-of-2, rounding is bit-shifting in Saber

- Bit-shifting is easy with Boolean shares
  To perform x>>k, shift x_1>>k and x_2>>k where $x = x_1 \oplus x_2$

- However, inputs to rounding are arithmetic shares

  E.g. Output of polynomial arithmetic is rounded

- Idea: Apply A2B transformation before rounding.
  Apply B2A transformation after rounding.

- [BDKBV20] proposes an *optimized implementation* that combines A2B+Shifting+B2A

[BDKBV20] MV. Beirendonck, JP D'Anvers, A. Karmakar, J. Balasch, I. Verbauwhede.
"A Side-Channel Resistant Implementation of SABER", ACM JETC.

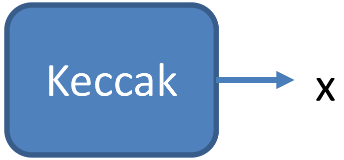
Masking of binomial sampling in Saber

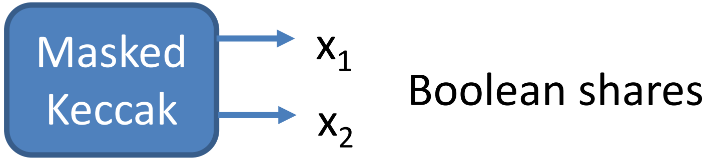
- Binomial sampling: Pseudo-random strings x and y as inputs. Produces

  Sample z = HammingWeight(x) - HammingWeight(y)

- Easy to compute on arithmetic shares.

- However, pseudorandom strings are generated by Keccak





- Optimized: [BDKBV20] evaluates ‘half adder/subtractor circuits’ on Boolean shares
  - ➢ Uses bit-slicing to improve performance

Results: 1st order masking of Saber

SW Results (ARM M4) [BDKBV20]

- Masked IND-CCA decapsulation has 2.5x cycle counts as overhead
- Overall masked decapsulation takes < 3M cycles
- Memory requirement increases by 1.84x

What helps masking in Saber?

- Power-of-2 moduli → Easier A2B conversions
- LWR has implicit error → Less error sampling

Preliminary HW Results (Xilinx FPGA)

Ongoing work by A. Basso, L. Prakop, and S. S. Roy

- Masked IND-CCA decapsulation has 2.4x cycle counts as overhead
- Area increase 1.3x

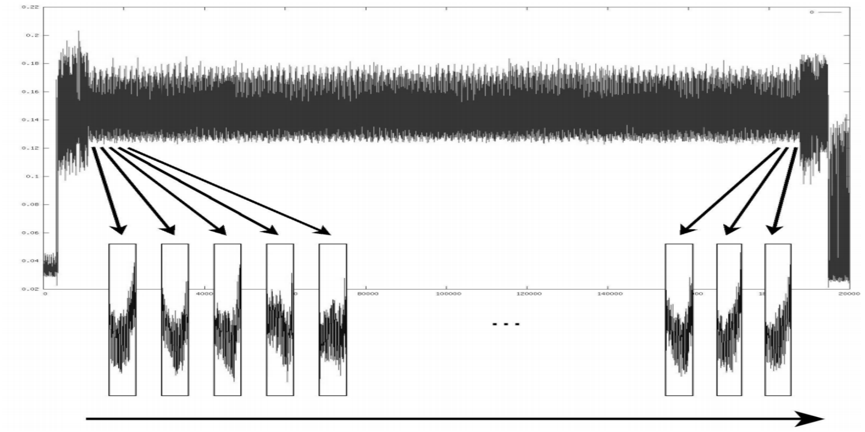
Outline

- ❑ Background:
  - ❑ Learning With Error (LWE) Problem
  - ❑ LWE/LWR-based PKE Framework (Main Focus)
- ❑ **Overview of Side-Channel Attacks:**
  - ❑ Algorithmic-Level
  - ❑ **Implementation-Level**
- ❑ Overview of Side-Channel Countermeasures:
- ❑ Future Works and Conclusion:

Implementation-based SCA on LWE/LWR-based PKE/KEMs

- ❑ **Major Computation Sub-blocks:**
  - ❑ Polynomial/Matrix-Vector Multiplication
  - ❑ Error/Secret Sampler (Gaussian/Sub-Gaussian Distribution)
  - ❑ PRF/PRNG – Extendable Output Function (XOF - (e.g.) SHAKE)

- ❑ **Single-trace key recovery** attacks using power/EM side-channel - Most Potent



Modus Operandi:

- ❑ Partition Trace into sub-traces (sensitive intermediates)

- ❑ Two common ways to extract information:
  - ❑ Horizontal CPA/DPA [CFG+10]
  - ❑ Template Matching and Algebraic techniques (Soft-Analytical SCA [VGS14])

[CFG+10] Clavier, Christophe, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. "Horizontal correlation analysis on exponentiation." In *International Conference on Information and Communications Security*, pp. 46-61. Springer, Berlin, Heidelberg, 2010.

[VGS14] Veyrat-Charvillon, Nicolas, Benoît Gérard, and François-Xavier Standaert. "Soft analytical side-channel attacks." In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 282-296. Springer, Berlin, Heidelberg, 2014.

Implementation-based SCA on LWE/LWR-based PKE/KEMs

Single Trace Key Recovery Attacks (Implementation Level)

Reported Works	Attack Technique	Target Scheme
School Book Multiplier (Poly Mul./Matrix-Vector Mul.)		
Aysu et al. [ATT+18]	Horizontal DPA (Extend and Prune)	Frodo and NewHope
Bos et al. [BFM+18]	Template Attack (Extend and Prune)	Frodo
Number Theoretic Transform (Poly Mul.)		
Primas et al. [PPM17]	Template Attack (SASCA)	Generic LWE/LWR-based PKE
Pessl et al. [PP20]	Template Attack (SASCA)	Generic LWE/LWR-based PKE
SHAKE (PRNG)		
Kannwischer et al. [KPP20]	Template Attack (SASCA)	Generic LWE/LWR-based PKE

Implementation-based SCA on LWE/LWR-based PKE/KEMs

- **Advantages:**
  - Single Trace Key Recovery
  - Only Side-Channel information sufficient (No communication with target-device)

- **Disadvantages**:
  - Requires some/complete knowledge of implementation
  - Sensitive to SNR (horizontal noise (jitter))

- **Countermeasures**:
  - **Shuffling** of intermediate operations within single computation [ZBT19, RPBC20]

[ATT+18] Aysu, Aydin, Youssef Tobah, Mohit Tiwari, Andreas Gerstlauer, and Michael Orshansky. "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols." In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 81-88. IEEE, 2018.

[PPM17] Primas, Robert, Peter Pessl, and Stefan Mangard. "Single-trace side-channel attacks on masked lattice-based encryption." In *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 513-533. Springer, Cham, 2017.

[PP19] Pessl, Peter, and Robert Primas. "More practical single-trace attacks on the number theoretic transform." In *International Conference on Cryptology and Information Security in Latin America*, pp. 130-149. Springer, Cham, 2019.

[HCY20] Huang, Wei-Lun, Jiun-Peng Chen, and Bo-Yin Yang. "Power analysis on NTRU prime." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020): 123-151.

[KPP20] Kannwischer, M. J., Pessl, P., & Primas, R. (2020). Single-Trace Attacks on Keccak. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, *2020*(3), 243-268.

[BFM+18] Bos, Joppe W., Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. "Assessing the feasibility of single trace power analysis of frodo." In *International Conference on Selected Areas in Cryptography*, pp. 216-234. Springer, Cham, 2018.

[RPBC20] Ravi, Prasanna, Romain Poussier, Shivam Bhasin, and Anupam Chattopadhyay. "On Configurable SCA Countermeasures Against Single Trace Attacks for the NTT." In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pp. 123-146. Springer, Cham, 2020.

[ZBT19] Zijlstra, Timo, Karim Bigou, and Arnaud Tisserand. "FPGA implementation and comparison of protections against SCAs for RLWE." In *International Conference on Cryptology in India*, pp. 535-555. Springer, Cham, 2019.

Outline

- Background:
  - Learning With Error (LWE) Problem
  - LWE/LWR-based PKE framework
- Overview of side-channel attacks:
  - Algorithmic-level
  - Implementation-level
- Overview of masking countermeasures:
- **Conclusions and future works:**

Conclusion:

- We cannot ignore side-channel security of lattice-based schemes
  - Several practical attacks which break only with a very few traces.
- Requirement of more analysis of SCA-protected implementations of lattice-based schemes.
- Scope for improvement in efficiency of masking countermeasures for LWE/LWR-based PKE/KEMs.
- Requirement of new techniques to concretely estimate security after SCA
  - Leaky LWE Estimator (Toolkit: https://github.com/lducas/leaky-LWE-Estimator)

Future Works:

More Attacks

- Scope for algorithmic-level SCA on NTRU:
  - Existing SCA mostly target the polynomial multiplier [ABGV08,MKS+10,WZW13,ZWW13,SMS19,HCY20]
  - Several PC Oracle-based key recovery attacks known for NTRU-based schemes [JJ00, GP07, ZCQ+21, DDS+19]

Countermeasures

- Fully masked implementations
- Scheme-specific countermeasures

References:

[JJ00] Jaulmes, Éliane, and Antoine Joux. "A chosen-ciphertext attack against NTRU." In *Annual International Cryptology Conference*, pp. 20-35. Springer, Berlin, Heidelberg, 2000.

[GP07] Gama, Nicolas, and Phong Q. Nguyen. "New chosen-ciphertext attacks on NTRU." In *International Workshop on Public Key Cryptography*, pp. 89-106. Springer, Berlin, Heidelberg, 2007.

[ZCQ+21] Zhang, Xiaohan, Chi Cheng, Yue Qin, and Ruoyu Ding. "Small Leaks Sink a Great Ship: An Evaluation of Key Reuse Resilience of PQC Third Round Finalist NTRU-HRSS."

[DDS+19] Ding, J., Deaton, J., Schmidt, K., Vishakha, Zhang, Z.: A simple and efficient key reuse attack on ntru cryptosystem (2019), https://eprint.iacr.org/2019/1022

[MKS+10] LEE Mun-Kyu, Jeong Eun Song, and HAN Dong-Guk. Countermeasures against power analysis attacks for the NTRU public key cryptosystem. IEICE transactions on fundamentals of electronics, communications and computer sciences, 93(1):153–163, 2010.

[WZW13] An Wang, Xuexin Zheng, and Zongyue Wang. Power analysis attacks and countermeasures on NTRU-based wireless body area networks. TIIS, 7(5):1094–1107, 2013.

[ZWW13] Xuexin Zheng, An Wang, and Wei Wei. First-order collision attack on protected NTRU cryptosystem. Microprocessors and Microsystems - Embedded Hardware Design, 37(6-7):601–609, 2013.

[ABGV08] AC Atici, Lejla Batina, Benedikt Gierlichs, and Ingrid Verbauwhede. Power analysis on NTRU implementations for RFIDs: First results. In The 4th Workshop on RFID Security, July 9th -11th, Budapest, 2008

[HCY20] Huang, Wei-Lun, Jiun-Peng Chen, and Bo-Yin Yang. "Power analysis on NTRU prime." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020): 123-151.

[SMS19] Schamberger, Thomas, Oliver Mischke, and Johanna Sepulveda. "Practical evaluation of masking for NTRUEncrypt on ARM Cortex-M4." In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 253-269. Springer, Cham, 2019.