Will you Cross the Threshold for Me?

Generic Side-Channel Assisted Chosen-Ciphertext Attacks on NTRU-based KEMs

Prasanna Ravi[1]

Martianus Frederic Ezerman[1]

Shivam Bhasin[1]

Anupam Chattopadhyay[1]

Sujoy Sinha Roy[2]

1. *NTU Singapore*
2. *TU Graz, Austria*

TCHES 2022, 21st Sep, 2022

Outline

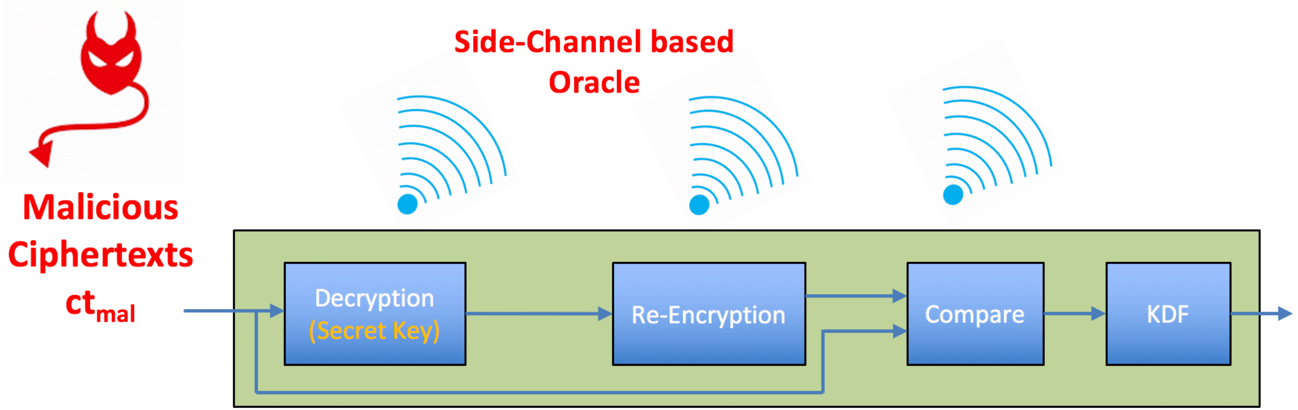
- ❑ **Motivation**
- ❑ Background: NTRU-based KEMs
- ❑ Side-Channel Assisted Chosen-Ciphertext Attacks:
  - ❑ Plaintext Checking (PC) Oracle-based SCA
  - ❑ Decryption Failure (DF) Oracle-based SCA
  - ❑ Full Decryption (FD) Oracle-based SCA
- ❑ Conclusion

Motivation

- Two categories of Lattice-based KEMs:
  - Learning With Errors/Rounding (LWE/LWR)
  - N^{th} order Truncated polynomial Ring Unit (NTRU)

- Lattice-based KEMs were heavily scrutinized by Side-Channel Analysis, particularly LWE/LWR-based KEMs.

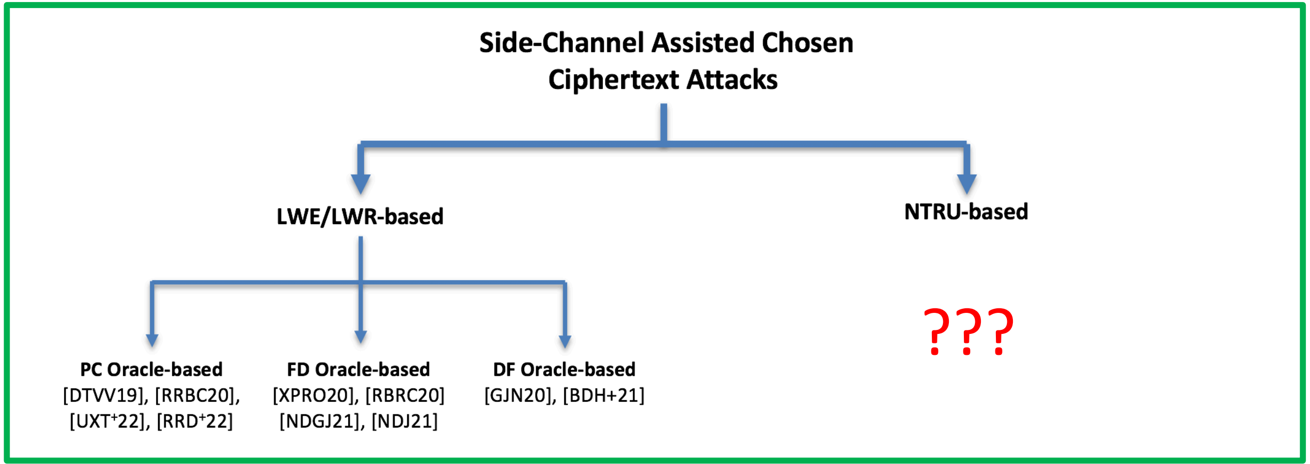
- Major Category of Attacks:
  - SCA Assisted Chosen-Ciphertext Attacks (SCA Assisted CCA)
  - [DTV+19, RRC+20, RBR+20, XPR+20, GJN20, BDH+21, RRD+22, … ,]



Motivation

- **Main Features of SCA Assisted CCA:**
  - Fairly Generic - Exploits inherent algorithmic properties of the scheme
  - Minimal/No knowledge of implementation (HW/SW)
  - Arguably the “Easiest SCA” on lattice-based KEMs

- There are three different flavours of SCA Assisted CCA
  - Plaintext-Checking (**PC**) Oracle-based SCA
  - Decryption-Failure (**DF**) Oracle-based SCA
  - Full-Decryption (**FD**) Oracle-based SCA



Motivation

- **Questions:**
  - Are similar attacks **possible** on NTRU-based KEMs?
  - If so, are NTRU-based KEMs more **easy/difficult** to be attacked compared to LWE/LWR-based KEMs?

- **Our Contributions:**
  - We propose generic SCA assisted CCA on NTRU-based KEMs
    - NTRU (Finalist) and NTRU Prime (Alternate Finalist)
  - **Approximately same effort** to break NTRU-based KEMs compared to LWE/LWR-based KEMs
  - **No. of Queries/Traces:** Few hundred to Few thousand chosen-ciphertext queries
  - Attack works for all parameters for NTRU and NTRU Prime with 100% success rate
  - Experimental Validation using EM side-channel on *pqm4* library on the ARM Cortex-M4 microcontroller

Our Contribution

Side-Channel Assisted Chosen Ciphertext Attacks

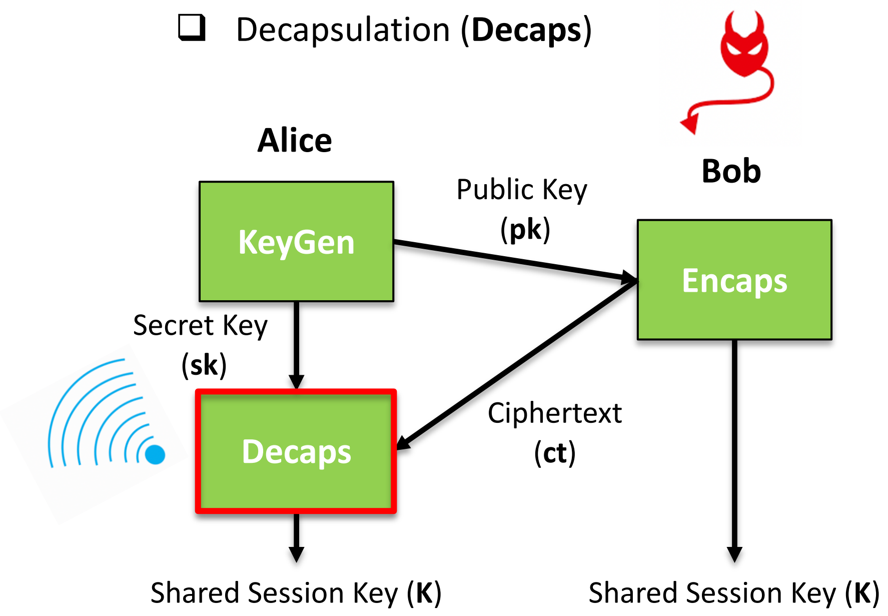
- **LWE/LWR-based**
  - **PC Oracle-based** [DTVV19], [RRBC20], [UXT$^+$22], [RRD$^+$22]
  - **FD Oracle-based** [XPRO20], [RBRC20] [NDGJ21], [NDJ21]
  - **DF Oracle-based** [GJN20], [BDH+21]
- **NTRU-based**
  - **PC Oracle-based** **[This Work]**
  - **FD Oracle-based** **[This Work]**
  - **DF Oracle-based** **[This Work]**

Outline

- ❑ Motivation
- ❑ **Background: NTRU-based KEMs**
- ❑ Side-Channel Assisted Chosen-Ciphertext Attacks:
  - ❑ Plaintext Checking (PC) Oracle-based SCA
  - ❑ Decryption Failure (DF) Oracle-based SCA
  - ❑ Full Decryption (FD) Oracle-based SCA
- ❑ Conclusion

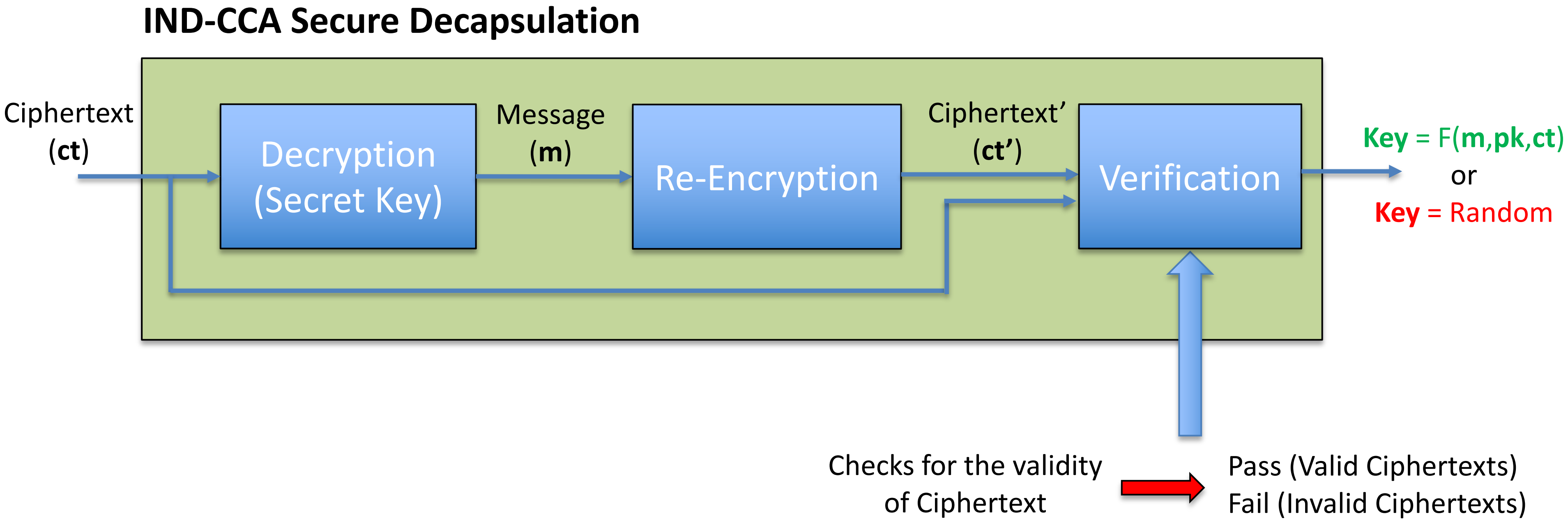
Key Encapsulation Mechanisms (KEMs)

- KEM is a cryptographic primitive used to derive a shared key between two untrusted parties.
- **Three Procedures**:
  - Key Generation (**KeyGen**)
  - Encapsulation (**Encaps**)
  - Decapsulation (**Decaps**)

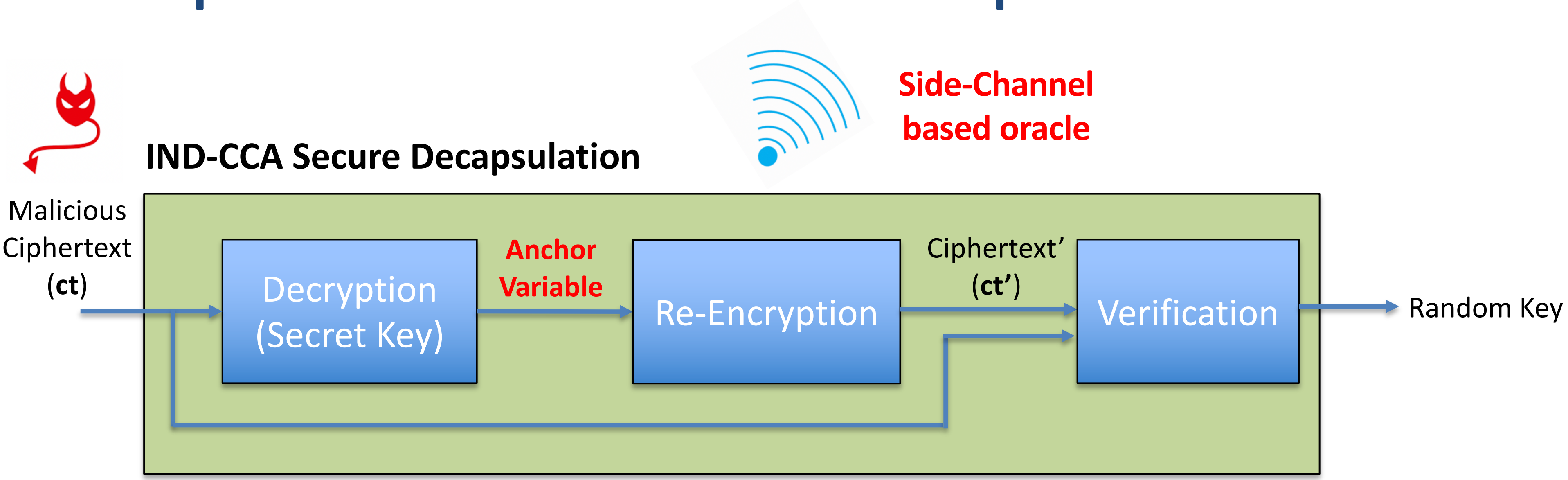


- Alice can reuse her keypair (**pk**, **sk**) to generate multiple session keys (**K**).
- **Static Key Setting**
- Compromise of **sk** leads to recovery of all session keys (**K**).

Decapsulation in Lattice-based KEMs



Decapsulation: SCA-based Chosen-Ciphertext Attacks

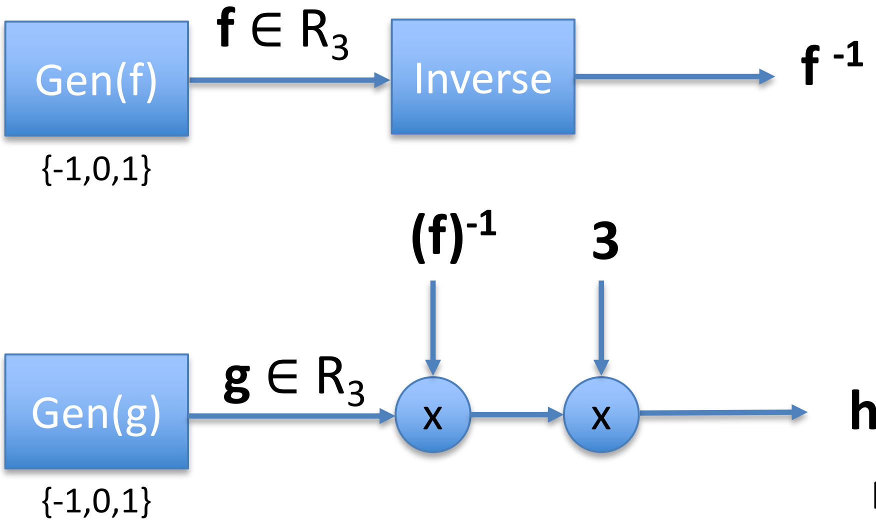


Key Idea:

- Build ciphertexts in order to control value of **secret dependent anchor variable**
- Use side-channels as **oracle** to recover information about anchor variable
- Key Recovery

IND-CPA secure NTRU PKE (Simplified)

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



(NTRU Instance)

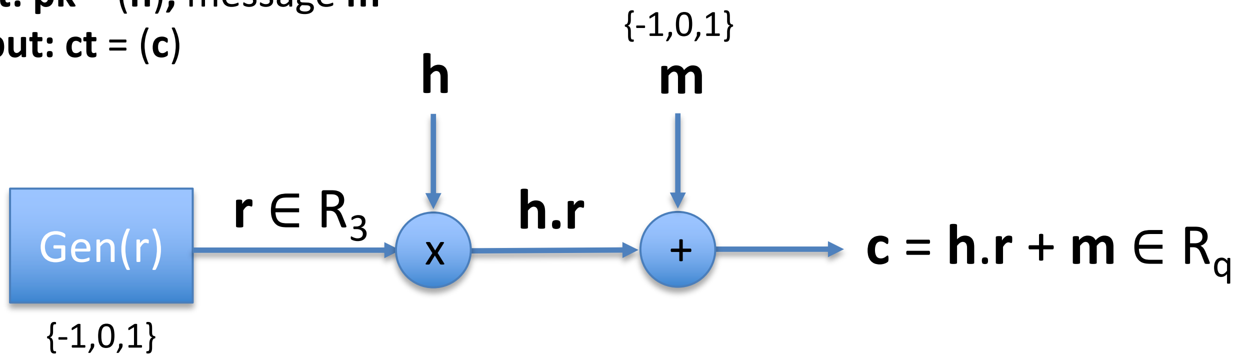
$\mathbf{h} = 3.g\ .\ (f)^{-1} \in R_q$

Public Key (**pk**): (**h**)

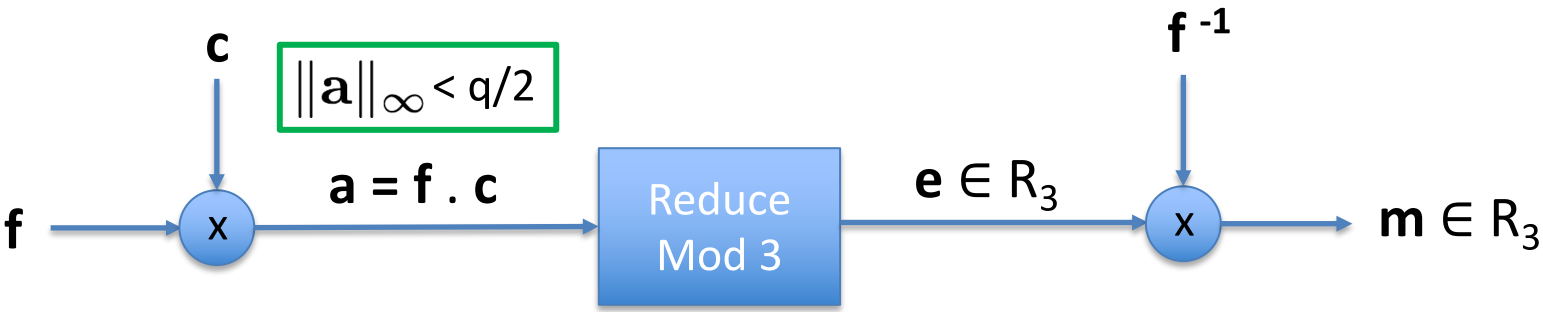
Secret Key (**sk**): (**g**,**f**)

IND-CPA secure NTRU PKE (Simplified)

- **Encryption:**
  - **Input: pk** = (**h**), message **m**
  - **Output: ct** = (**c**)



- **Decryption:**
  - **Input:** ct = (**h**), sk = (**f**, **g**)
  - **Output:** m



Outline

- ❑ Motivation
- ❑ Background: NTRU-based KEMs
- ❑ **Side-Channel Assisted Chosen-Ciphertext Attacks:**
  - ❑ **Plaintext Checking (PC) Oracle-based SCA**
  - ❑ Decryption Failure (DF) Oracle-based SCA
  - ❑ Full Decryption (FD) Oracle-based SCA
- ❑ Conclusion

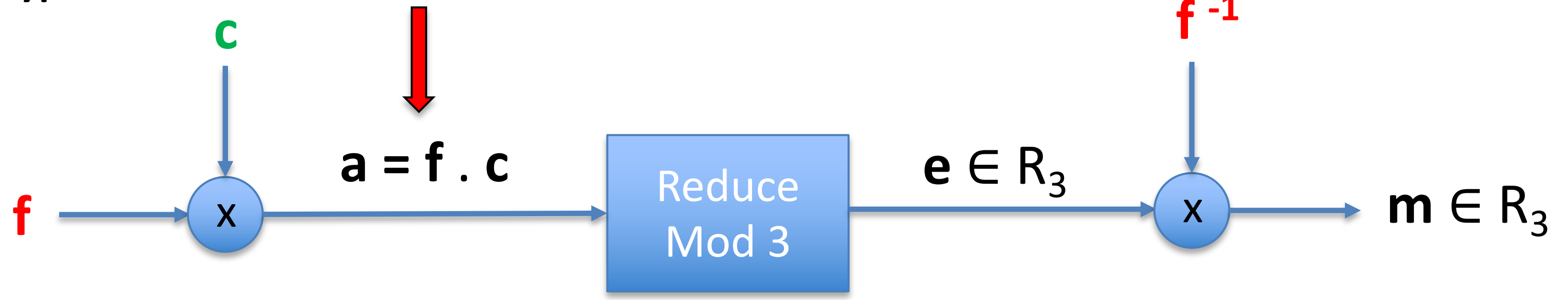
Plaintext Checking (PC) Oracle-based SCA

- Inspired from classical chosen-ciphertext attack on NTRU PKE by Jaulmes and Joux in Crypto 2000

- **Two Phases**:
  - **Pre-Processing Phase**: Search for a base ciphertext ($\mathbf{c}_{\mathbf{base}}$)
    - Leakage upon decryption reveals critical information about secret key
  - **Key-Recovery Phase**:
    - Use $\mathbf{c}_{\mathbf{base}}$ to build attack ciphertexts ($\mathbf{c}_{\mathbf{attack}}$), whose leakage enables key recovery

Pre-Processing Phase: Search for c_{base}

- Decryption:



Chosen c = $(k_1 \cdot \mathbf{t_1}) + (k_2 \cdot \mathbf{t_2} \cdot \mathbf{h})$

$\mathbf{t_1} = x^{i1} + x^{i2} + x^{i3} + \ldots + x^{im}$

$\mathbf{t_2} = x^{j1} + x^{j2} + x^{j3} + \ldots + x^{jn}$

(i1, i2, i3, … , im), (j1, j2, j3, …, jn)

randomly chosen indices

$(k_1, k_2) \in 3\mathbb{Z}$

$\mathbf{a} = \mathbf{f} \cdot \mathbf{c}$

$= 3k_1 \cdot \mathbf{t_1} \cdot \mathbf{f} + k_2 \cdot \mathbf{t_2} \cdot \mathbf{g}$

$\mathbf{t_1} \cdot \mathbf{f} = (x^{i1} \cdot \mathbf{f}) + (x^{i2} \cdot \mathbf{f}) + \ldots + (x^{im} \cdot \mathbf{f})$

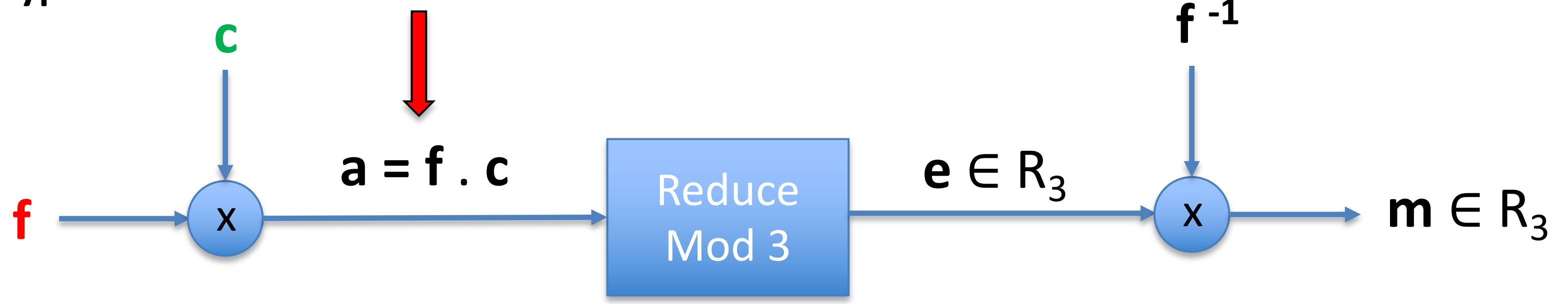
$= \mathrm{Rot}(f, i_1) + \mathrm{Rot}(f, i_2) + \ldots + \mathrm{Rot}(f, i_m)$

= Sum of Rotations of **f**

$\mathrm{absmax}((\mathbf{t_1}.\mathbf{f})[i]) = m$

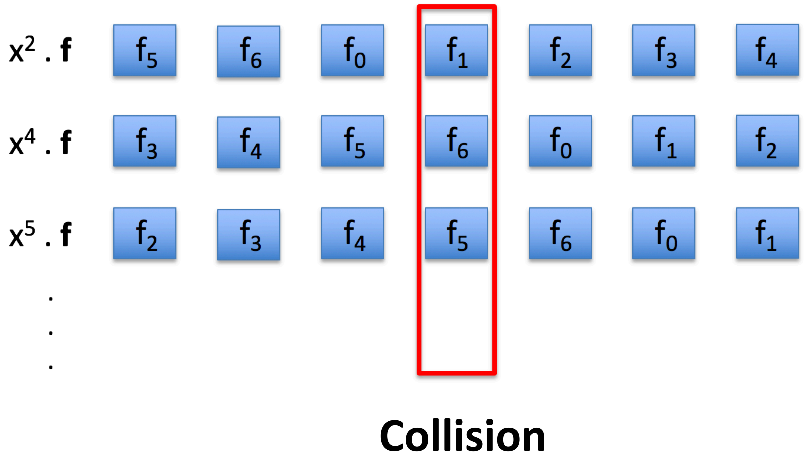
Pre-Processing Phase: Search for c_{base}

- Decryption:



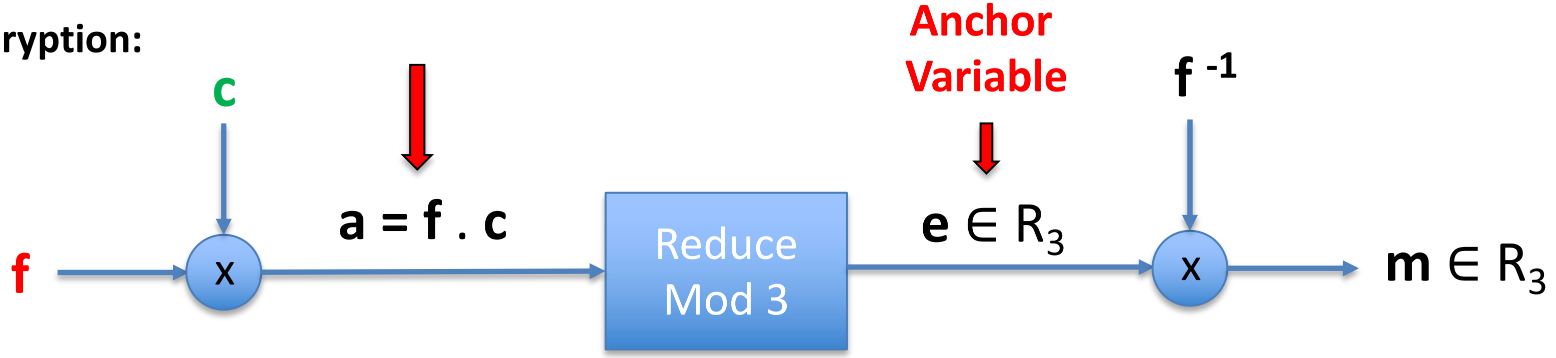
$$t_1 . f = (x^{i1} . f) + (x^{i2} . f) + \dots + (x^{im} . f)$$
$$= Rot(f,i_1) + Rot(f,i_2) + \dots + Rot(f,i_m)$$
$$= \text{Sum of Rotations of } \mathbf{f}$$

$$absmax((t_1.f)[i]) = m$$



Pre-Processing Phase: Search for c_{base}

- **Decryption:**



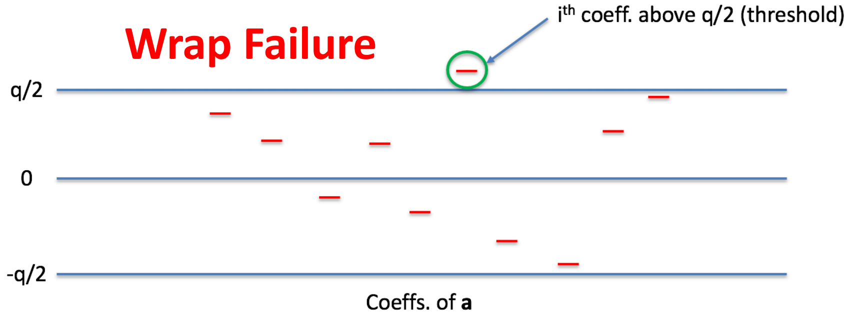
$\mathbf{a} = \mathbf{f} \cdot \mathbf{c}$

$= 3k_1 \cdot \mathbf{t_1 \cdot f} + k_2 \cdot \mathbf{t_2 \cdot g}$

absmax($\mathbf{a}$[i]) = $(3k_1 \cdot m + k_2 \cdot n)$ → Collision(**f,g,i**)

Choose (m, n, k_1, k_2) such that:

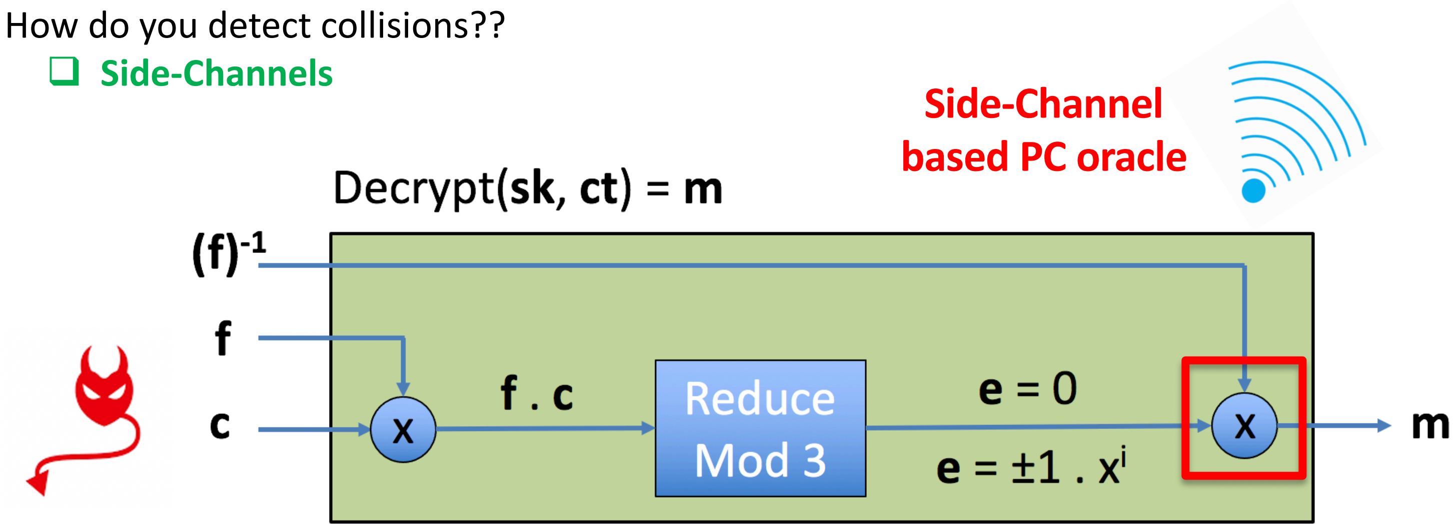
- $(3k_1 \cdot m + k_2 \cdot N) > q/2$
- Maximize Prob(Single Collision)



If single collision: $\mathbf{e = +/- x^i}$
If no collision: $\mathbf{e = 0}$

Pre-Processing Phase: Detect Collisions for c_{base}

- How do you detect collisions??
  - **Side-Channels**



Pre-Processing Phase: Detect Collisions for c_{base}

- Two Class Classification: **Welch's t-test** for Collision Detection

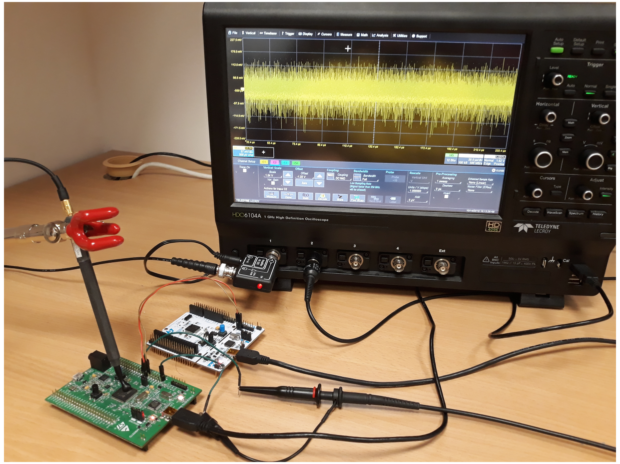
- Decapsulate zero ciphertext **c** = 0 (**e** = 0) : T_o (n = 10 executions)

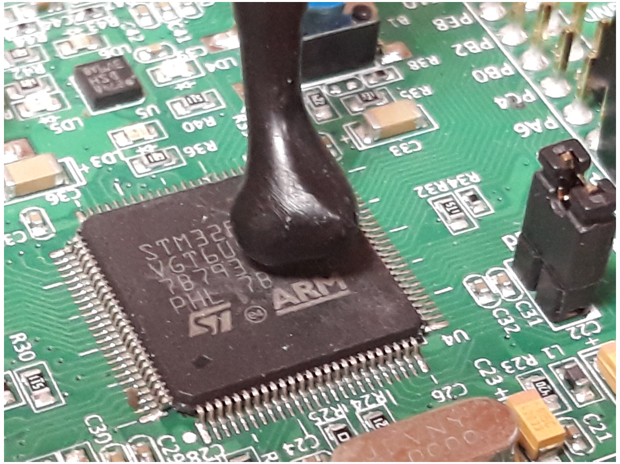
- Decapsulate chosen ciphertext **c'** : T_X (n = 10 executions)

- Compute the Welch's t-test between T_o and T_X

Experimental Setup:

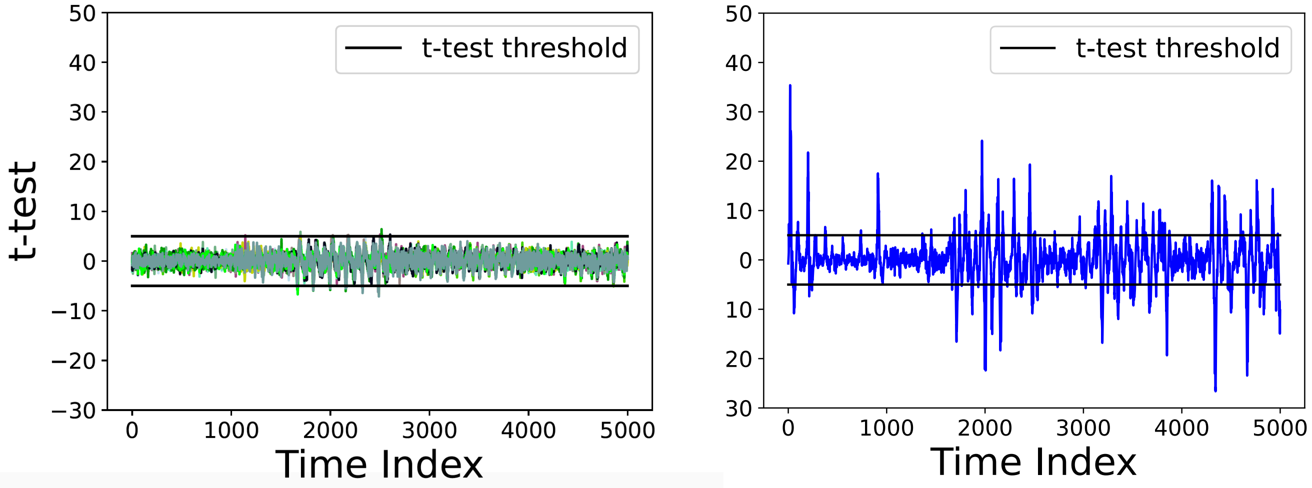
- **Target**: Optimized Implementations of NTRU, NTRU Prime from pqm4 library.
- **Platform**: STM32F407VG MCU based on the 32-bit ARM Cortex-M4 processor (24 MHz).
- **Leakage Acquisition**: EM side-channels using near-field EM probe (500 Msamples/sec)





Pre-Processing Phase: Detect Collisions for c_{base}

- Two Class Classification: **Welch's t-test** for Collision Detection
- Decapsulate zero ciphertext **c** = 0 (**e** = 0) : T_o (n = 10 executions)
- Decapsulate chosen ciphertext **c'** : T_X (n = 10 executions)
- Compute the Welch's t-test between T_o and T_X



(a) No Collision for c'

(b) Single Collision for c'

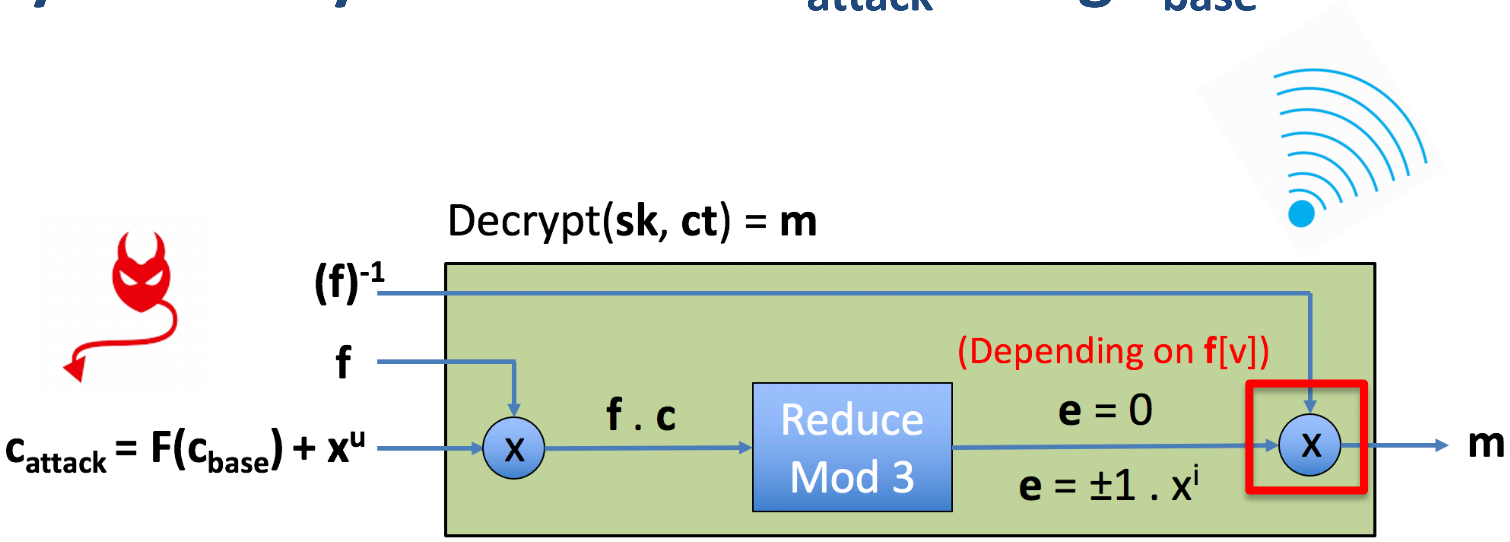
Select features above threshold as PoI

Use PoI to construct template for both classes

RT_O – Class **O**

RT_X – Class **X**

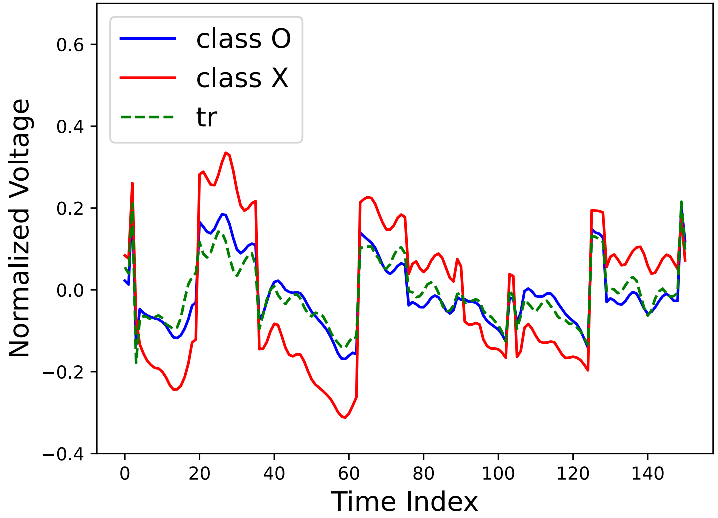
Key Recovery Phase: Build c_{attack} using c_{base}



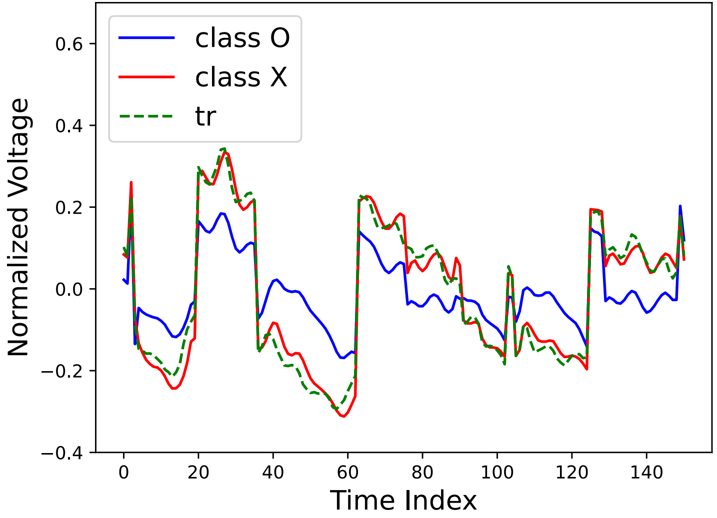
- ❑ Value of **e = 0/± x^i^** (i.e.) No-Collision/Collision depends upon a single coefficient **f**[v]
- ❑ For **f**[v] ∈ {-1,0,1}, we can build a binary distinguisher for every candidate of **f**[v] based on
  - ❑ Collision (Class O) / No-Collision (Class X)
- ❑ Side-Channel templates used to classify a given attack ciphertext as Class O/Class X

Key Recovery Phase: Classify c_{attack} as Class O/Class X

- Given a trace *tr* from decryption of $\mathbf{c}_{attack}$, reduced templates can be used to classify as Class O/Class X.



(a) Class O



(b) Class X

- Single trace classification between Class O/Class X : 100% success rate

Experimental Results: PC Oracle Attack on NTRU

- We successfully validated our attack on all parameters of NTRU.
- **Success Rate**: 100% with trace complexity: 1.8k - 2.9k traces

Scheme	t_{base}	t_{total}	Scheme	t_{base}	t_{total}
ntruhps2048509	70	1791	ntruhps4096821	30	2911
ntruhps2048677	100	2364	ntruhrss701	70	2447

- PC Oracle-based SCA on Kyber [RRCB20, UXT+22]: 1k - 3k traces

Experimental Results: PC Oracle Attack on NTRU Prime

- We successfully validated our attack on all parameters of Streamlined NTRU Prime.

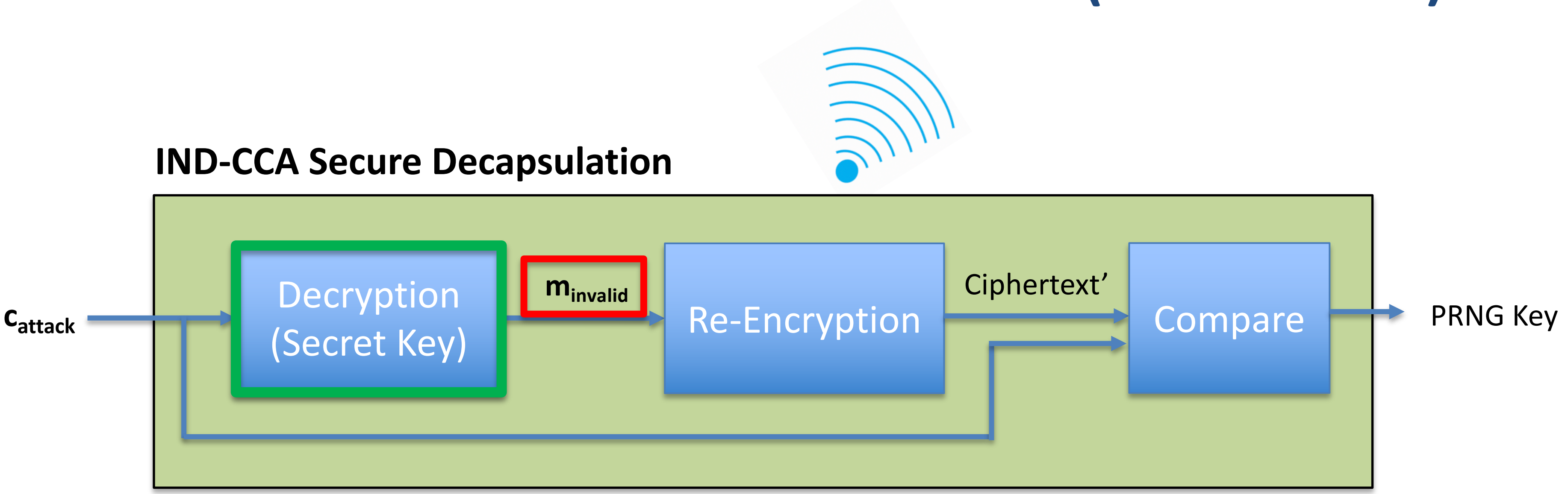
- **Success Rate**: 100% with trace complexity: 3k - 4.6k traces

Scheme	t_{base}	t_{total}	Scheme	t_{base}	t_{total}
sntrup653	420	3005	sntrup953	270	3601
sntrup761	390	3269	sntrup1013	320	4026
sntrup857	420	3731	sntrup1277	240	4688

- PC Oracle-based SCA on Kyber [RRCB20, UXT+22]: 1k - 3k traces

- At no point, does the attacker utilize any information about the implementation

Observations on PC Oracle-based SCA (NTRU Prime)



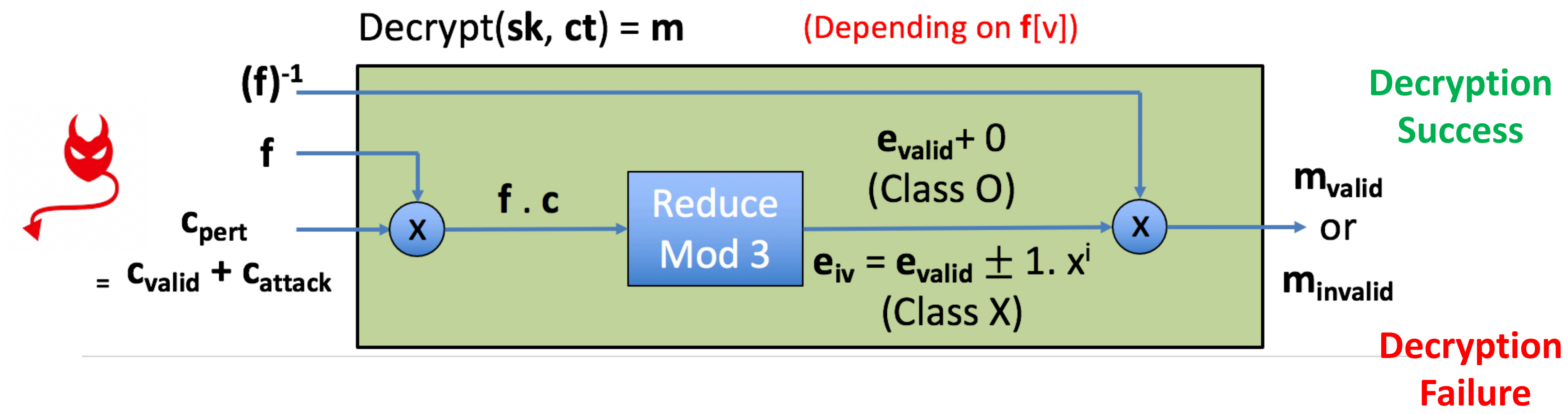
- ❑ Information about **e** (anchor variable) does not propagate beyond decryption
- ❑ NTRU Prime adopts a weight check failure within decryption
  - ❑ which always fails for attack ciphertexts
- ❑ Can we widen the scope of the attack (target side-channel leakage from re-encryption procedure) ??

Outline

- ❑ Motivation
- ❑ Background: NTRU-based KEMs
- ❑ **Side-Channel Assisted Chosen-Ciphertext Attacks:**
  - ❑ Plaintext Checking (PC) Oracle-based SCA
  - ❑ **Decryption Failure (DF) Oracle-based SCA**
  - ❑ Full Decryption (FD) Oracle-based SCA
- ❑ Conclusion

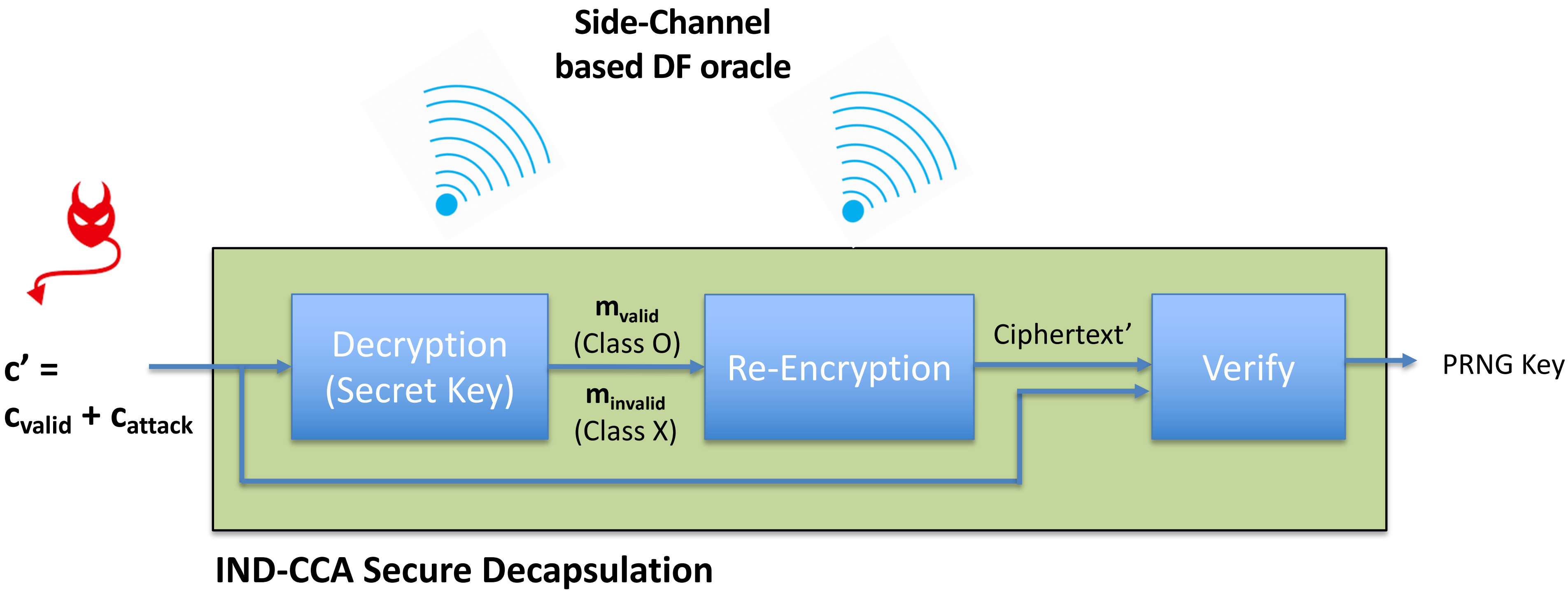
Decryption Failure (DF) Oracle-based SCA

- **Key Idea:** We perturb valid ciphertexts $\mathbf{c}_{valid}$ with the attack ciphertexts $\mathbf{c}_{attack}$ (PC Oracle-based SCA)



Decryption Failure (DF) Oracle-based SCA

- **Key Idea:** We perturb valid ciphertexts $\mathbf{c_{valid}}$ with the attack ciphertexts $\mathbf{c_{attack}}$ (PC Oracle-based SCA)



Experimental Results: DF Oracle-based SCA (NTRU Prime)

- We successfully validated our attack on all parameters of Streamlined NTRU Prime.
- **Success Rate**: 100% with trace complexity: 4k - 5k traces

Scheme	t_{base}	t_{total}	Scheme	t_{base}	t_{total}
sntrup653	1630	4182	sntrup953	760	4436
sntrup761	1650	4566	sntrup1013	740	4603
sntrup857	1200	4631	sntrup1277	410	5287

- DF Oracle-based attack on Kyber [HPP21]: 5k-8k traces

Outline

- ❑ Context
- ❑ Background: NTRU-based KEMs
- ❑ Side-Channel Assisted Chosen-Ciphertext Attacks:
  - ❑ Plaintext Checking (PC) Oracle-based SCA
  - ❑ Decryption Failure (DF) Oracle-based SCA
  - ❑ Full Decryption (FD) Oracle-based SCA
- ❑ **Conclusion**

Conclusion:

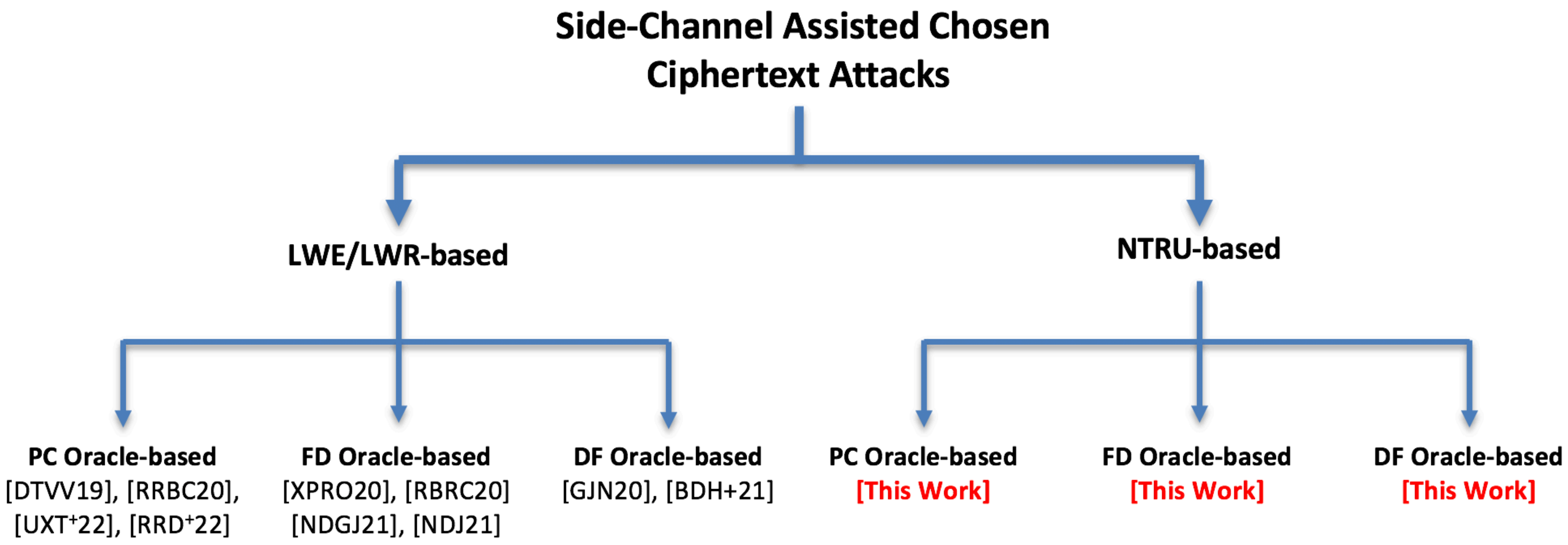
- ❑ We have demonstrated generic SCA assisted CCA on NTRU-based KEMs

Type of Oracle	Oracle Response
Plaintext Checking (**PC**) Oracle	msg = m_0 or m_1
Decryption Failure (**DF**) Oracle	msg = m_{valid} or $m_{invalid}$
Full Decryption (**FD**) Oracle	msg = m

- ❑ **Take-Home Message:** Breaking NTRU KEMs through SCA assisted CCA similar to LWE/LWR-based KEMs
- ❑ Experimental Validation using EM side-channel on the ARM Cortex-M4 microcontroller
- ❑ Our attacks demonstrate the ease of attacking unprotected implementations for key recovery
  - ❑ Implementation Agnostic
  - ❑ Easiest SCA
- ❑ Code Package (including traces) is available at:

https://github.com/PRASANNA-RAVI/SCA_Assisted_CCA_on_NTRU

Thank you!!!


Side-Channel Assisted Chosen Ciphertext Attacks
LWE/LWR-based
NTRU-based
PC Oracle-based
[DTVV19], [RRBC20], [UXT+22], [RRD+22]
FD Oracle-based
[XPRO20], [RBRC20] [NDGJ21], [NDJ21]
DF Oracle-based
[GJN20], [BDH+21]
PC Oracle-based
[This Work]
FD Oracle-based
[This Work]
DF Oracle-based
[This Work]

References

[DTV+19] 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.

[RRC+20] 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.

[XPR+20] 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.

[RBR+20] 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.

References

[NDG+21] 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.

[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.

[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." In IACR-TCHES 2021.

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

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

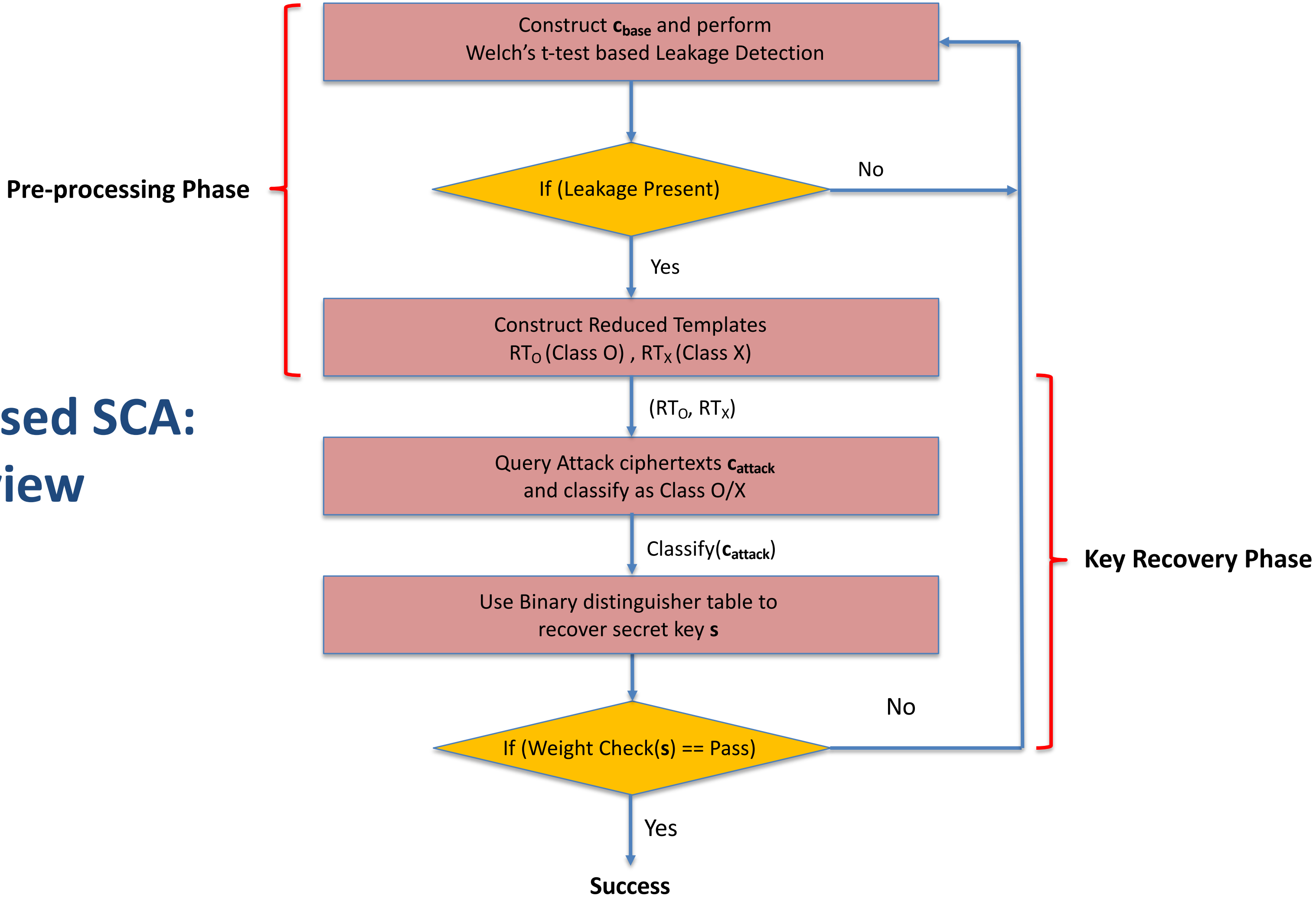
References

[RRD+22] Rajendran, Gokulnath, Prasanna Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. "Pushing the Limits of Generic Side-Channel Attacks on LWE-based KEMs-Parallel PC Oracle Attacks on Kyber KEM and Beyond." *Cryptology ePrint Archive* (2022).

[UXT+22] Ueno, Rei, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. "Curse of re-encryption: A generic power/em analysis on post-quantum kems." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2022): 296-322.

Backup

PC Oracle-based SCA: Attack Overview



More Efficient Key Recovery Attacks

- The PC and DF oracle-based SCA extract binary information (1-bit) about the secret key, thus require thousand of traces to recover the full secret-key.

- Is it possible to extract more than a 1-bit information about sensitive intermediates?

- In LWE/LWR-based schemes, several works [SKL+20, RBR+20, NDG+21] have shown that the message encoding/decoding procedures leak information about all the 256 bits of the sensitive decrypted message.

- Are there similar vulnerabilities present in NTRU-based schemes ?

- Sim et al. [SKL+20] showed that there are similar operations in the NTRU decryption procedure which manipulate single coefficients of the decrypted message, enabling full message recovery in a single trace.

- Such side-channel leakage can be used to instantiate a much more informative oracle to perform efficient key recovery attacks – **Full Decryption (FD) oracle-based SCA**

FD Oracle-based SCA

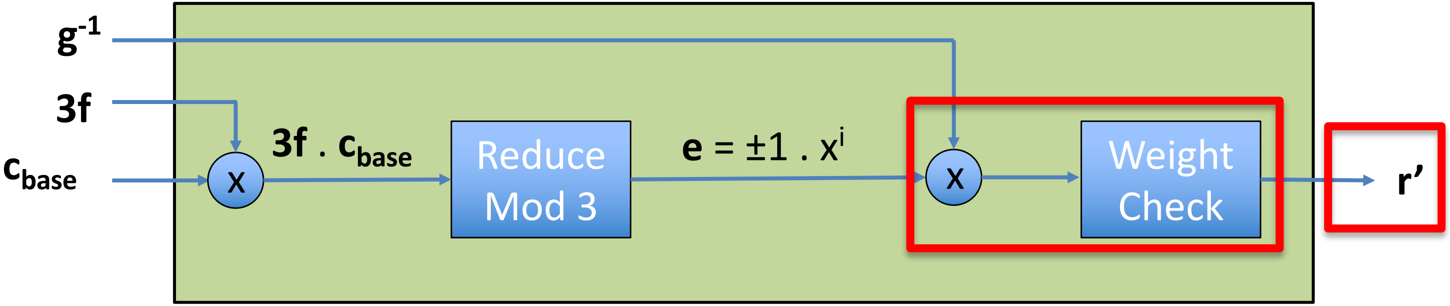
Decrypt(**sk**, **ct**) = **r’**

Secret Key (**sk**): (**f**,**g**)

Ciphertext (**ct**): $\mathbf{c}_{\mathbf{attack}}$

Message (**r’**): **r’**

Side-Channel based FD oracle



- The weight-check operation within the decryption procedure manipulates single coefficients of the decrypted messsage **r’**.

- If SCA leakage can be used to recover complete decrypted message **r’**, then attacker can query the decryption procedure with $\mathbf{c}_{\mathbf{base}}$ and recover the complete secret polynomial **g** as

$$\mathbf{g} = \mathbf{e} \,.\, (\mathbf{r'})^{-1}$$

FD Oracle-based SCA

Trace complexity of FD Oracle-based SCA on NTRU Prime (assuming perfect FD oracle)

Scheme	t_{total}	Scheme	t_{total}
sntrup653	420	sntrup953	270
sntrup761	390	sntrup1013	320
sntrup857	420	sntrup1277	240

Trace complexity of FD Oracle-based SCA on NTRU (assuming perfect FD oracle)

Scheme	t_{total}	Scheme	t_{total}
ntruhps2048509	70	ntruhps4096821	30
ntruhps2048677	100	ntruhrss701	70

- ❑ FD Oracle-based SCA on LWE/LWR-based schemes: 9 traces (Kyber768) [XPR+20], 12 traces (Saber) [NGJ+21]
- ❑ **Key Difference**: No search of $\mathbf{c}_{base}$ required for LWE/LWR-based schemes