NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

On Threat of Hardware Trojan To Post-Quantum Lattice-based Schemes: A Key Recovery Attack on SABER and beyond

Prasanna Ravi[1,2], Suman Deb[2]
Anubhab Baksi[1], Anupam Chattopadhyay[1,2]
Shivam Bhasin[1], Avi Mendelson[3]

1. Temasek Labs, NTU Singapore
2. SCSE, NTU Singapore
3. Technion, Israel

SPACE-2021, 11th Dec, 2021

Motivation

- We are now in the **third** and **final** round of the NIST's standardization process for **P**ost-**Q**uantum **C**ryptography (PQC) [DAA+20]

Type	Digital Signatures (DSs)	Key Encapsulation Mechanisms (KEMs)	Main Finalists	Alternate Finalists
Lattice Based	**2**	**3**	**5**	**2**
Code-Based	-	1	1	2
Multivariate	1	-	1	1
Hash-Based	-	-	-	2
Isogeny based	-	-	-	1
Others	-	-	-	0
Total	3	4	7	8

[DAA+20] Moody, Dustin, Gorjan Alagic, Daniel C. Apon, David A. Cooper, Quynh H. Dang, John M. Kelsey, Yi-Kai Liu et al. "Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process." (2020).

Motivation

- Standardization of PQC will spur wide-scale adoption in commercial devices and applications.
- Given the urgent need towards transition to PQC, we can expect prominent use of 3rd Party IP (3PIP) cores implementing PQC in real-world systems.
- In a 3PIP setting, **Hardware Trojans (HT)** naturally become a potent attack vector to break practical implementations of PQC.
- We perform the first study of susceptibility of PQC based Key Encapsulation Mechanisms (KEMs) to Hardware Trojans.
- **Main Target**: Lattice-based KEMs based on the Learning With Error/Rounding (LWE/R) problem.
- **Main Takeaway:** LWE/LWR-based KEMs contain inherent algorithmic properties which can be exploited to perform HT-based attacks.

Outline

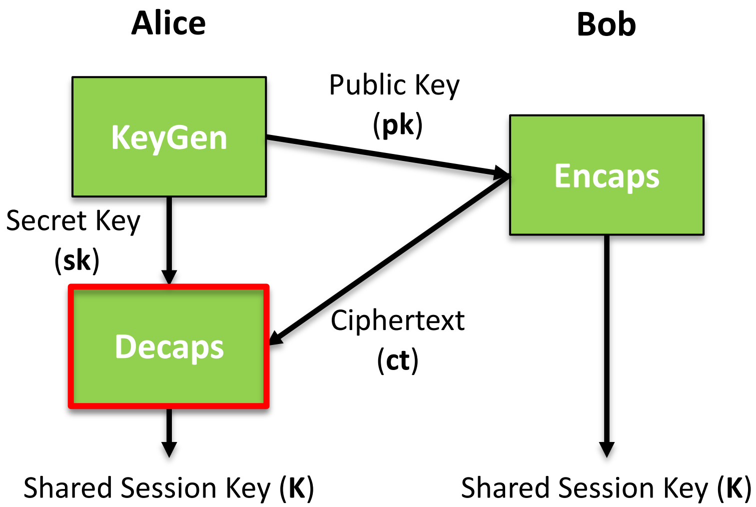
- ❑ Motivation
- ❑ Background:
  - ❑ Lattice-based KEMs (LWE/LWR-based Problem)
  - ❑ Chosen-Ciphertext Attack on LWE/LWR-based KEMs
- ❑ HT assisted Chosen-Ciphertext Attack
- ❑ HT Design Methodology
- ❑ Experimental Results
- ❑ Conclusion

Outline

- Motivation
- Background:
  - Lattice-based KEMs (LWE/LWR-based Problem)
  - Chosen-Ciphertext Attack on LWE/LWR-based KEMs
- HT assisted Chosen-Ciphertext Attack
- HT Design Methodology
- Experimental Results
- Conclusion

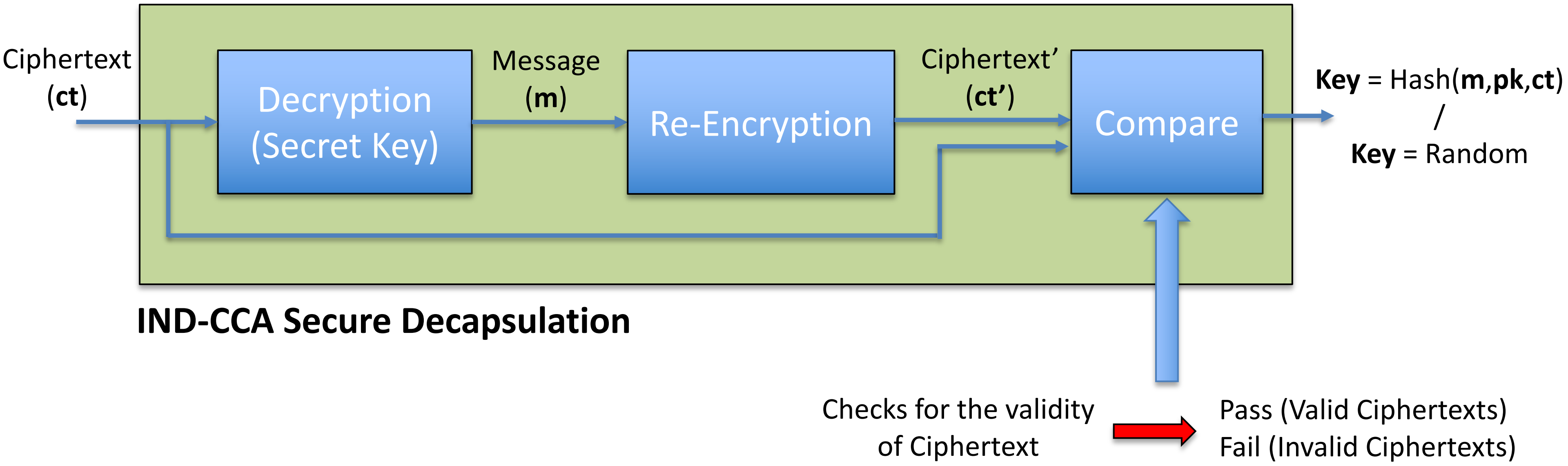
Key Encapsulation Mechanisms (KEMs)

- 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**).
- KEMs can be used in protocols such as TLS to perform key exchange for encrypted communication.
- **Kyber** and **Saber** are two main finalists for KEMs in the NIST PQC standardization process.
- Compromise of **sk** leads to recovery of all session keys (**K**).

Decapsulation in Lattice-based KEMs



IND-CCA Secure Decapsulation

Outline

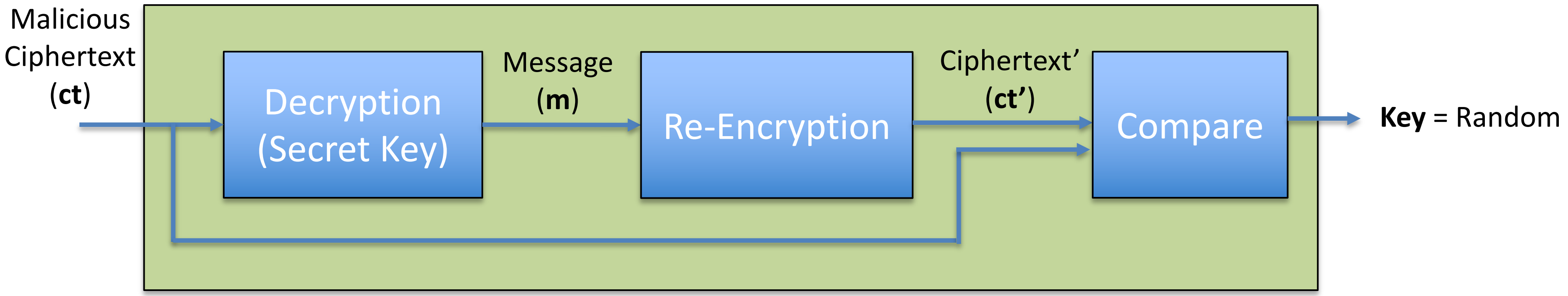
- ❑ Motivation
- ❑ **Background:**
  - ❑ Lattice-based KEMs (LWE/LWR-based Problem)
  - ❑ **Practical Chosen-Ciphertext Attacks on LWE/LWR-based KEMs**
- ❑ HT assisted Chosen-Ciphertext Attack
- ❑ HT Design Methodology
- ❑ Experimental Results
- ❑ Conclusion

Practical Chosen Ciphertext Attacks (CCA)

Side-Channel-based Plaintext Checking (PC) Oracle

Ciphertext	Message
CT1	M2'
CT2	M3'
CT3	M0'

Full Recovery of **sk**



IND-CCA Secure Decapsulation

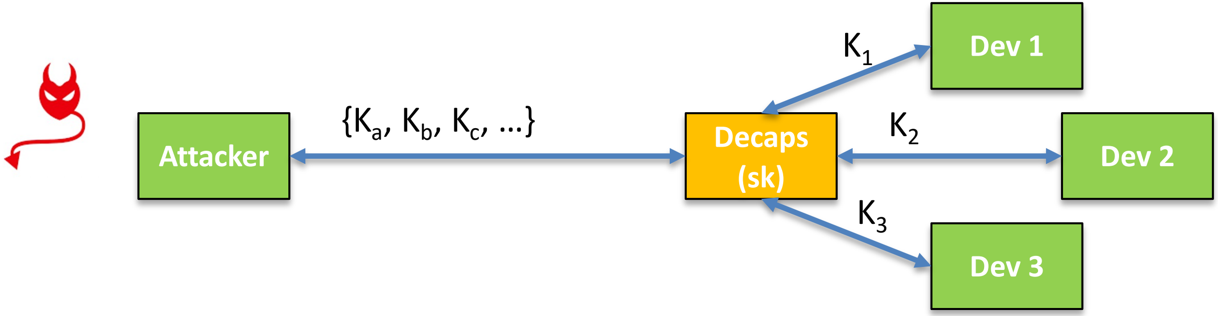
Practical Chosen Ciphertext Attacks (CCA)

- There have been several Side-Channel assisted CCAs on LWE/LWR-based KEMs [DTVV19, RRBC20, XPRO20, NDGJ21]
- These attacks utilize **side-channel** as a PC oracle to obtain information about the decrypted message **m**.
- **Main Question**: In a 3PIP setting, can we utilize HTs to instantiate a PC oracle for key recovery?
- In this work, we propose the first **Hardware Trojan assisted CCA** on LWE/LWR-based KEMs.

[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.
[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.
[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.
[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.

Adversary Model

- Attacker sells malicious 3PIP core implementing PQC based KEM (**KeyGen**, **Encaps**, **Decaps**).
- HT is only implemented in **Decaps** procedure.
- **Decaps** uses static keys (**pk**,**sk**) for key exchange (Generated on device / Installed by user).
- **Attacker's Target**: Recover static secret key (**sk**) used by **Decaps.**
- **Modus Operandi**: Attacker tries to query the **Decaps** procedure with chosen-ciphertexts and uses session keys (**K**) to recover **sk.**



Outline

- ❑ Motivation
- ❑ Background:
  - ❑ Lattice-based KEMs (LWE/LWR-based Problem)
  - ❑ Chosen-Ciphertext Attack on LWE/LWR-based KEMs
- ❑ **HT assisted Chosen-Ciphertext Attack**
- ❑ HT Design Methodology
- ❑ Experimental Results
- ❑ Conclusion

Chosen Ciphertext Attack [RRBC20]

- Build structured ciphertexts (ct = **u**,**v**)
- (**u**,**v**) – two polynomials ($\mathbf{u}_i$ denotes i^{th} coeff of **u**)
- $\mathbf{s}_i$ denotes i^{th} coeff of secret key polynomial **s**

Chosen u | k | 0 | 0 | 0

Chosen v | p | 0 | 0 | 0

Binary Distinguisher for s_0

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

m | $f(s_0)$ | 0 | 0 | 0

Ciphertext (ct) → Decryption (Secret Key) → Message (m)

Chosen Ciphertext Attack [RRBC20]

- Build structured ciphertexts (ct = **u**,**v**)
- (**u**,**v**) – two polynomials ($\mathbf{u}_i$ denotes i^{th} coeff of **u**)
- $\mathbf{s}_i$ denotes i^{th} coeff of secret key polynomial **s**

Chosen u | 0 | k | 0 | 0

Chosen v | p | 0 | 0 | 0

Binary Distinguisher for s_1

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

m | $f(s_1)$ | 0 | 0 | 0

Ciphertext (**ct**) → Decryption (Secret Key) → Message (**m**)

Chosen Ciphertext Attack [RRBC20]

- Build structured ciphertexts (ct = **u**,**v**)
- (**u**,**v**) – two polynomials ($\mathbf{u}_i$ denotes i^{th} coeff of **u**)
- $\mathbf{s}_i$ denotes i^{th} coeff of secret key polynomial **s**

Chosen u	0	0	k	0
Chosen v	p	0	0	0

Binary Distinguisher for s_2

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

m	$f(s_2)$	0	0	0

Chosen Ciphertext Attack [RRBC20]

- Build structured ciphertexts (ct = **u**,**v**)
- (**u**,**v**) – two polynomials ($\mathbf{u}_i$ denotes i^{th} coeff of **u**)
- $\mathbf{s}_i$ denotes i^{th} coeff of secret key polynomial **s**

Chosen u: 0 | 0 | k | 0

Chosen v: p | 0 | 0 | 0

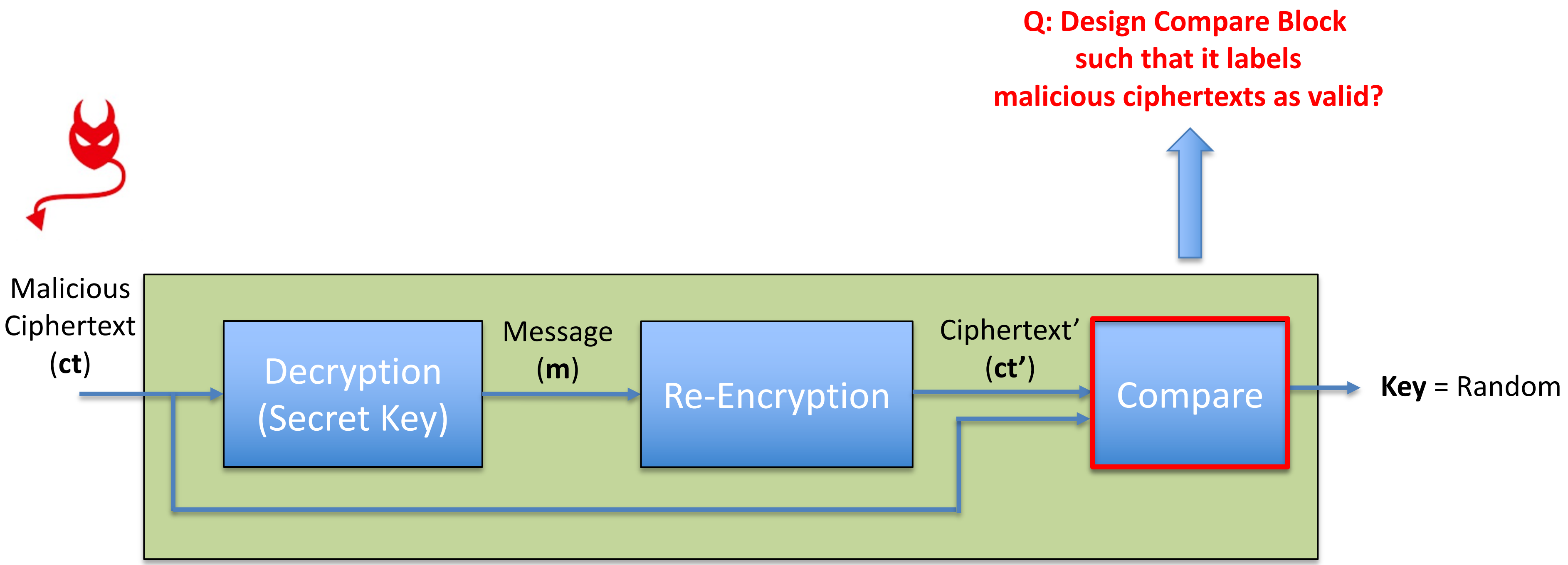
Q: How can attacker obtain information about m (Oracle)?

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

m: $f(s_2)$ | 0 | 0 | 0

Ciphertext (**ct**) → Decryption (Secret Key) → Message (**m**)

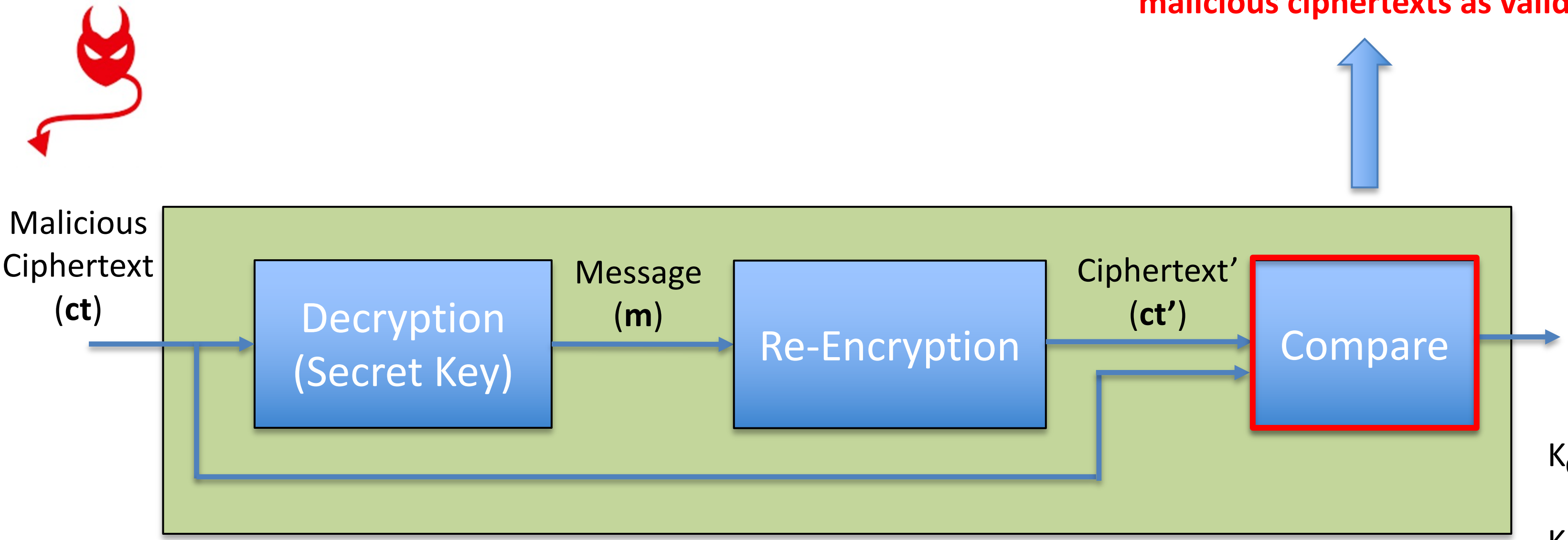
Chosen Ciphertext Attack [RRBC20]



IND-CCA Secure Decapsulation

Chosen Ciphertext Attack [RRBC20]

Q: Design Compare Block such that it labels malicious ciphertexts as valid?



IND-CCA Secure Decapsulation

Key = Hash(**m**,**pk**,**ct**)

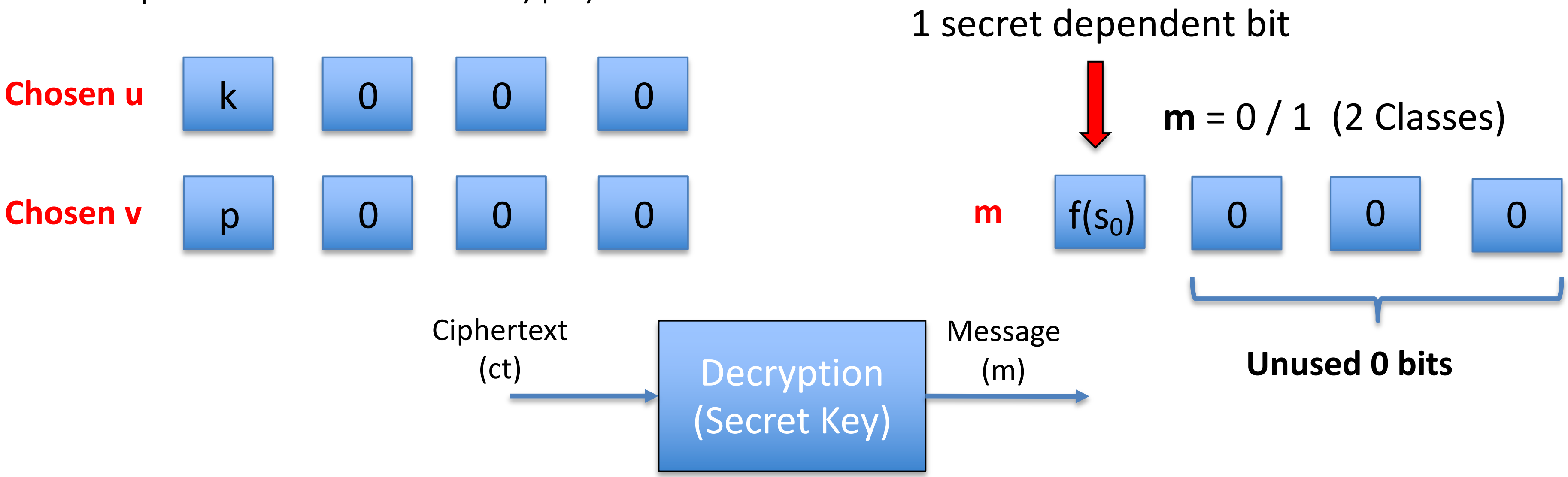
K_0: Hash(**0**,**pk**,**ct**)
or
K_1: Hash(**1**,**pk**,**ct**)

Chosen Ciphertext Attack [RRBC20]

- If session key **K** is used to encrypt a message (say "HELLO"), attacker can guess the value of **K** (**K_0**/**K_1**) from the encrypted data and thus deduce **m** = 0/1.
- In each query, attacker obtains 1-bit information about **m** and thus, 1-bit info. about **sk**.
- For recommended parameter sets of **Saber**, full key recovery is possible in ≅ **2.09k** queries.
- For recommended parameter sets of **Kyber**, full key recovery is possible in ≅ **1.76k** queries.
- The attack requires a few thousand queries for key recovery.
- We propose an improved attack variant that reduces the query complexity.

Parallelized Chosen Ciphertext Attack

- Build handcrafted ciphertexts (ct = **u**,**v**)
- (**u**,**v**) – two polynomials ($\mathbf{u}_i$ denotes i^{th} coeff of **u**)
- $\mathbf{s}_i$ denotes i^{th} coeff of secret key polynomial **s**



Parallelized Chosen Ciphertext Attack

- Build handcrafted ciphertexts (ct = **u**,**v**)
- (**u**,**v**) – two polynomials ($\mathbf{u}_i$ denotes i^{th} coeff of **u**)
- $\mathbf{s}_i$ denotes i^{th} coeff of secret key polynomial **s**

In a single query, attacker can gain info. about 2 secret coeffs.

Chosen u	k	0	0	0
Chosen v	p	p	0	0

m = 0 / 1 / 2 / 3 (4 Classes)

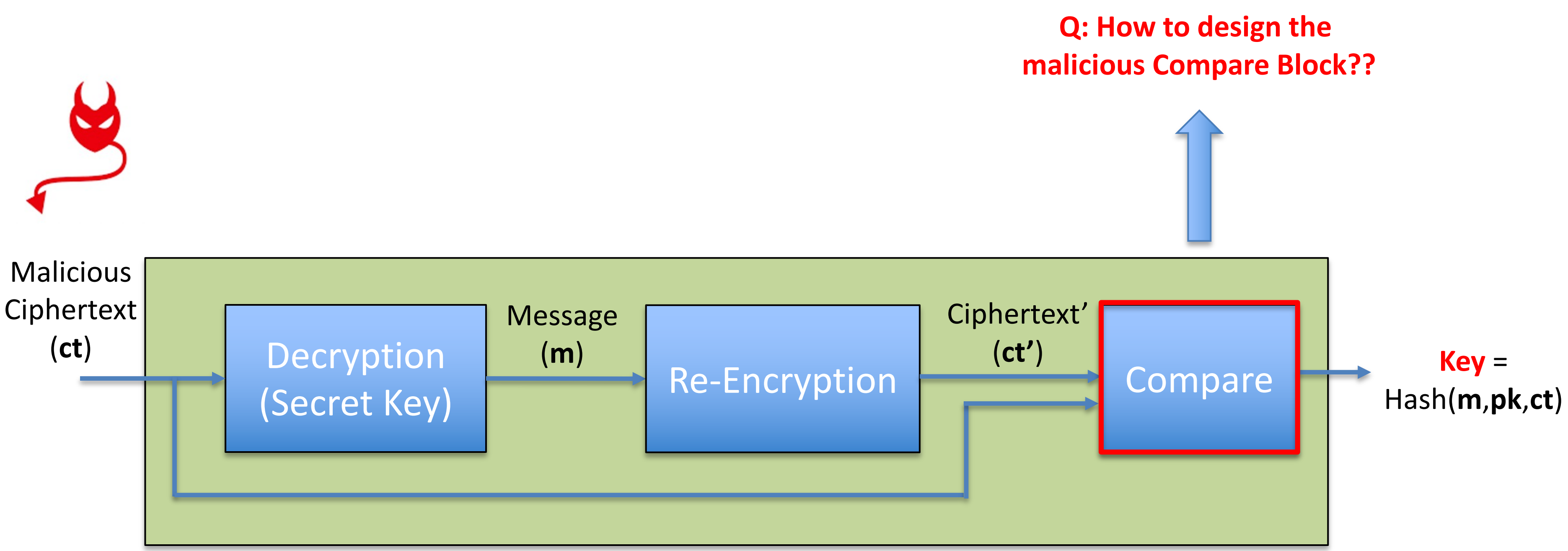
m	$f(s_0)$	$f(s_1)$	0	0

Ciphertext (ct) → Decryption (Secret Key) → Message (m)

Parallelized Chosen Ciphertext Attack

- We can have configurable number of secret dependent message bits (**t**)
- Let Query complexity for original attack (binary PC oracle) = **K**
- If we have **t** secret dependent message bits, key recovery can be done in (**K/t**) queries (reduction by factor **t**).
- For each query, attacker has to perform $\mathbf{2^t}$ computations (offline) to guess **t** bits of the message.
- If **t** = 32,
  - Saber: 192 queries and 2^{39} offline computational complexity
  - Kyber: 96 queries and 2^{38} offline computational complexity
- Attacker can choose **t** depending upon his/her computational constraints.

HT assisted Chosen Ciphertext Attack



IND-CCA Secure Decapsulation

Outline

- ❑ Motivation
- ❑ Background:
  - ❑ Lattice-based KEMs (LWE/LWR-based Problem)
  - ❑ Chosen-Ciphertext Attack on LWE/LWR-based KEMs
- ❑ HT assisted Chosen-Ciphertext Attack
- ❑ **HT Design Methodology**
- ❑ Experimental Results
- ❑ Conclusion

HT Design Methodology

- **Requirement:**
  - **Valid** ciphertexts should be labelled as **valid**
  - **Invalid** ciphertexts should be labelled as **invalid**
  - **Malicious** ciphertexts should be labelled as **valid**

- **Idea 1:** Can we design a HT that triggers on ciphertext input?
  - Use structure of input chosen **ct’s** for identification
  - **Problem**: But, chosen **ct’s** have very low entropy. Accidental triggering during functional testing.

- We exploit another algorithmic property of LWE/LWR-based KEMs: **Ciphertext Malleability**

- We exploit **Ciphertext Malleability** to design the HT trigger mechanism

Exploiting Ciphertext Malleability for HT Trigger

Chosen u	k	0	0	0	0	0	0	0	0
Chosen v	p	p	0	0	0	0	0	0	0
m	$f(s_0)$	$f(s_1)$	0	0	0	0	0	0	0

Unused 0 bits

- ❑ Can we use the unused bits of **m** to embed a trigger pattern?
- ❑ **Bit-Flip Property** [RBRC20]: Adding C to i^{th} coefficient of **v** ($\mathbf{v_i}$) , flips i^{th} bit of **m** ($\mathbf{m_i}$)

[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.

Exploiting Ciphertext Malleability for HT Trigger

Chosen u	k	0	0	0	0	0	0	0	0
Chosen v	p	p	C	0	0	0	0	0	0
m	$f(s_0)$	$f(s_1)$	1	0	0	0	0	0	0

Unused 0 bits

- Can we use the unused bits of **m** to embed a trigger pattern?
- **Bit-Flip Property** [RBRC20]: Adding C to i^{th} coefficient of **v** ($\mathbf{v}_i$) , flips i^{th} bit of **m** ($\mathbf{m}_i$)

[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.

Exploiting Ciphertext Malleability for HT Trigger

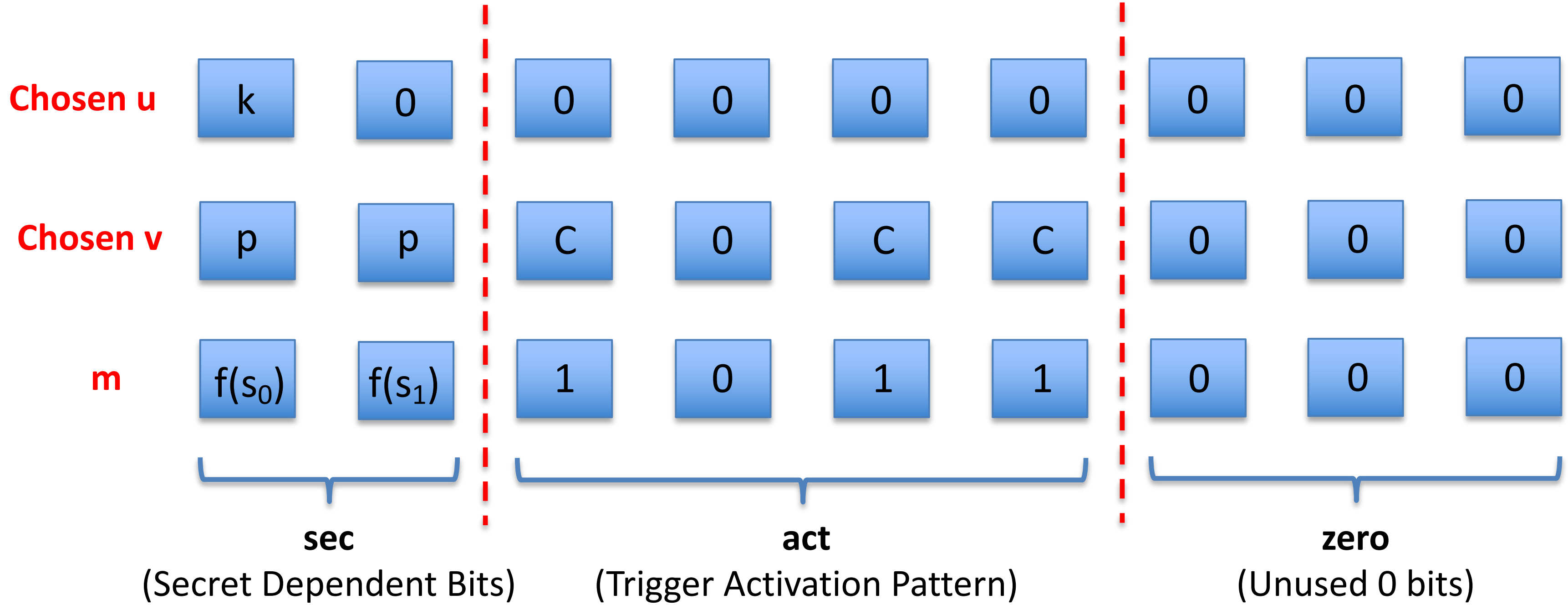
Chosen u	k	0	0	0	0	0	0	0	0
Chosen v	p	p	C	0	C	C	0	0	0
m	$f(s_0)$	$f(s_1)$	1	0	1	1	0	0	0

Unused 0 bits

- ❑ Can we use the unused bits of **m** to embed a trigger pattern?
- ❑ **Bit-Flip Property** [RBRC20]: Adding C to i^{th} coefficient of **v** (**v_i**) , flips i^{th} bit of **m** (**m_i**)

[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.

Exploiting Ciphertext Malleability for HT Trigger



- We can build ciphertexts which decrypt to **m** of the form: (**sec** | **act** | **zero**)
- Chosen-Ciphertext Technique + Ciphertext Malleability -> Algorithmic properties of LWE/LWR-based KEMs

HT Trigger

- For attacker's chosen ciphertexts, **act** portion of **m** contains trigger activation pattern (**trig_pattern)**

- **Logic:**
  - Compares the **act** portion of **m** with fixed **trig_pattern**
  - If comparison succeeds, incoming ciphertext is the attacker's ciphertext => **Activate HT**

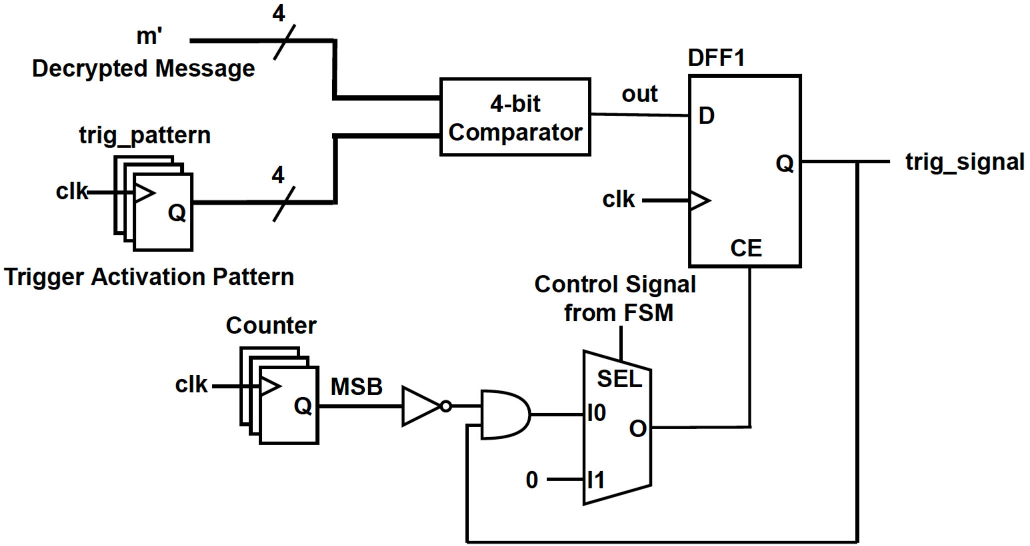
- Attacker should choose A (**trig_pattern**) so as to have a negligible trigger activation probability

- If len(**act**) = A, trigger activation probability = 2^{-A}

HT Trigger

- **Target Implementation**: Open-source implementation of Saber by Roy and Basso [RB20]

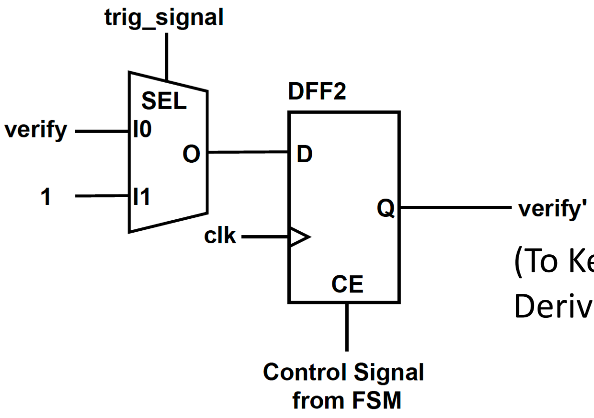
- The message **m** (from decryption procedure) is generated in a serial fashion - 4 bits at a time.



[RB20] Roy, Sujoy Sinha, and Andrea Basso. "High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020): 443-466.

HT Payload

- Modifies the output of the ciphertext comparison block – Labels attacker's ciphertexts as valid
- 1-bit switch (1-bit Multiplexer)
- True output of Ciphertext Comparision = *verify* (Pass – 1/ Fail – 0)

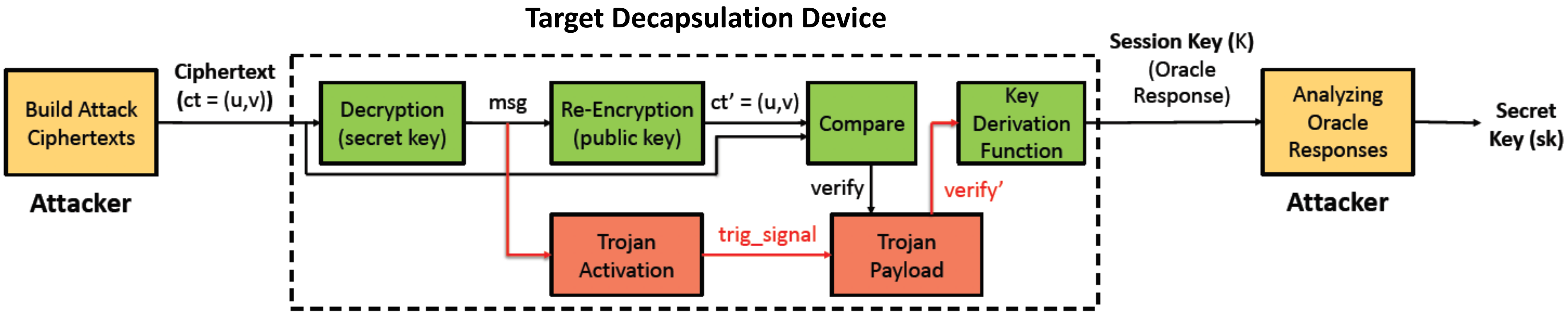


(To Key
Derivation Block)

HT operation for different Ciphertext Inputs

Type of Ciphertext	*trig_signal*	*verify*	*verify'*
Valid	0	1	1
Invalid	0	0	0
Malicious	**1**	**0**	**1**

Attack Flow of HT assisted CCA



Improved HT Design: Unified Trigger and Payload

HT operation for different Ciphertext Inputs

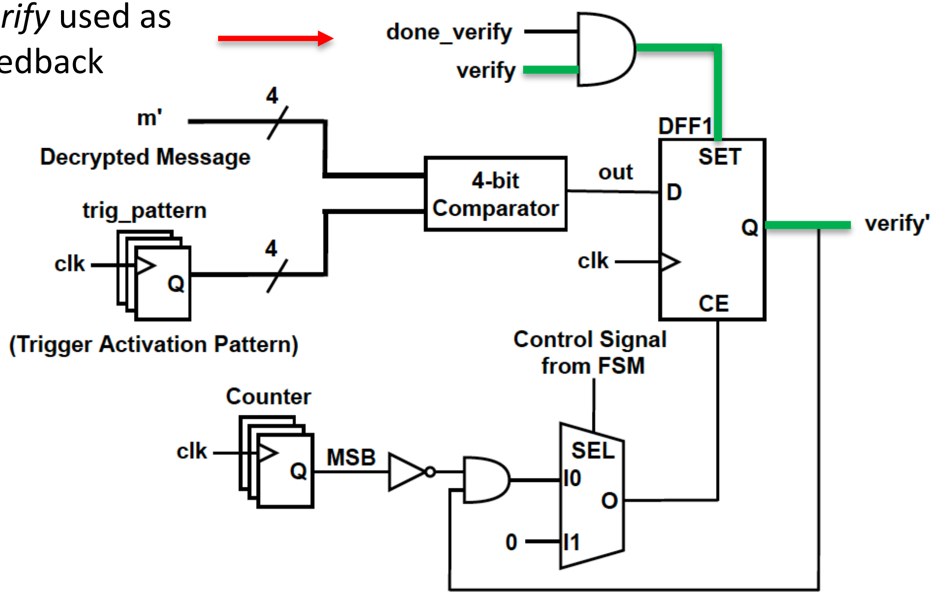
Type of Ciphertext	*trig_signal*	*verify*	*verify'*
Valid	0	1	1
Invalid	0	0	0
Malicious	1	0	1

Same HT payload for valid and Malicious Ciphertexts

- **Idea**: HT can be activated even for valid ciphertexts
- **Advantage**: HT also participates in normal operation of target (Improves HT's stealthiness)
- *verify* (output of ciphertext comparison) is used as a feedback signal to HT
- **Result**: Unified HT Trigger and Payload

Improved HT Design: Unified Trigger and Payload

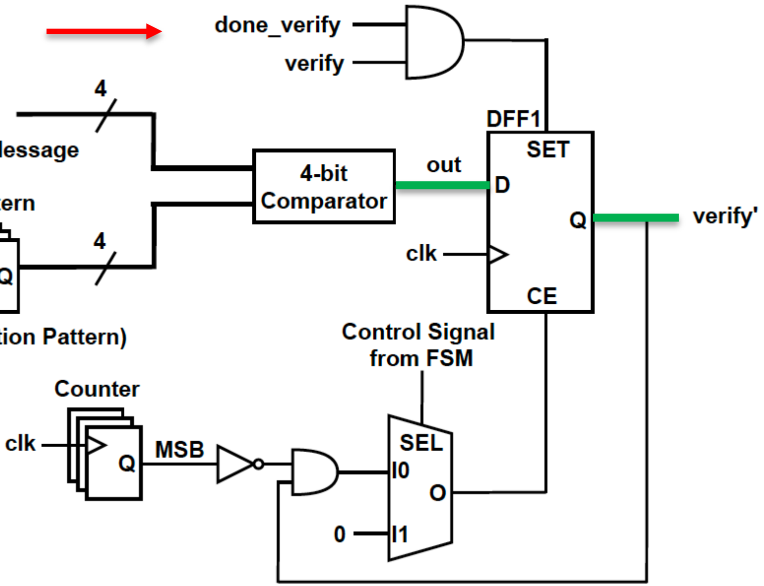
verify used as feedback



Valid Ciphertexts (Normal Op.)

Improved HT Design: Unified Trigger and Payload

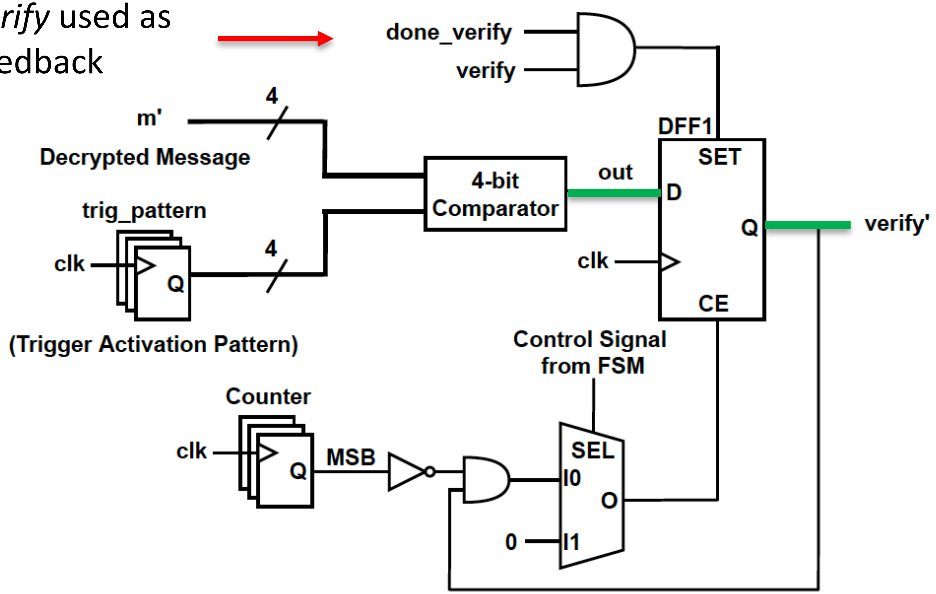
verify used as feedback



Invalid Ciphertexts (Normal Op.)

Improved HT Design: Unified Trigger and Payload

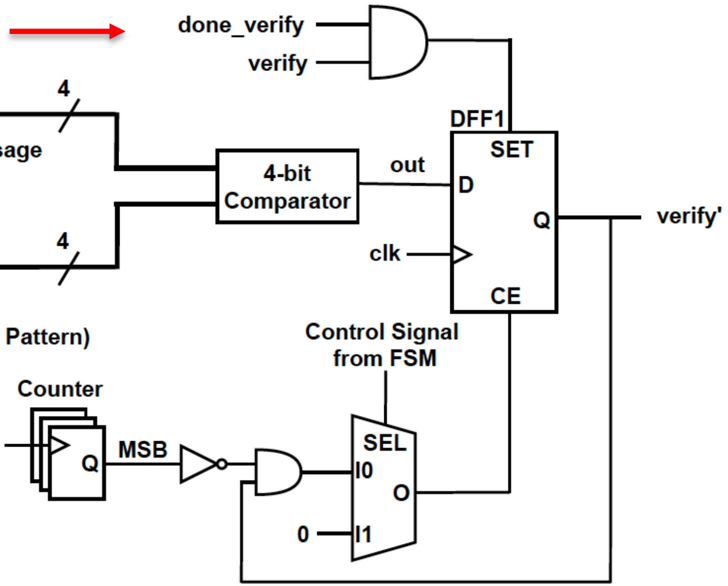
verify used as feedback



Malicious Ciphertexts (Attacker Op.)

Improved HT Design: Unified Trigger and Payload

verify used as feedback



- **Advantages (Anti-Detection):**
  - No Dormant Signals
  - HT participation in normal operation
  - Sequential arrival of trigger vector
  - Non Trivial leakage of secret Key

Outline

- ❑ Motivation
- ❑ Background:
  - ❑ Lattice-based KEMs (LWE/LWR-based Problem)
  - ❑ Chosen-Ciphertext Attack on LWE/LWR-based KEMs
- ❑ HT assisted Chosen-Ciphertext Attack
- ❑ HT Design Methodology
- ❑ **Experimental Results**
- ❑ Conclusion

Experimental Results

- We implemented the HT for implementations of Saber for the Zynq UltraScale FPGA (xczu9eg-vb1156-2-e).

- We implement the HT for different lengths of the trigger activation pattern (32-bit, 64-bit 128-bit)

Implementation	No. (Overhead in %)					
	FFs	**LUTs**	**LUTRAMs**	**BRAMs**	**IOs**	**BUFGs**
HT-Free	9747 (-)	24103 (-)	0	2	189	7
128-bit HT	9843 (0.98%)	24111 (0.03%)	6	2	189	7
64-bit HT	9797 (0.51%)	24110 (0.03%)	0	2	189	7
32-bit HT	9766 (0.19%)	24096 (-0.03%)	0	2	189	7

- **Advantages**:
  - Configurable HT design
  - Low area overhead
  - Generically applicable to several LWE/LWR-based KEMs

Outline

- Motivation
- Background:
  - Lattice-based KEMs (LWE/LWR-based Problem)
  - Chosen-Ciphertext Attack on LWE/LWR-based KEMs
- HT assisted Chosen-Ciphertext Attack
- HT Design Methodology
- Experimental Results
- **Conclusion**

Conclusion

- We present the first HT based key recovery attack for LWE/LWR-based KEMs.
- **Attack Methodology**: Chosen-Ciphertext Attack (+ Ciphertext Malleability)
- Our attack primarily exploits algorithmic properties of LWE/LWR-based KEMs for key recovery
- Full key recovery possible in a **few hundred** to **few thousand** queries to decapsulation device
- **Area overhead** (HW implementation of Saber): 0.98% (FFs) and 0.03% (LUTs)
- Favourable characteristics which could provide strong resistance against several detection techniques.
- For more information, please visit

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