RSA Conference2018
San Francisco | April 16 – 20 | Moscone Center

SESSION ID: CRYP-W14

CRYPTANALYSIS AGAINST SYMMETRIC-KEY SCHEMES WITH ONLINE CLASSICAL QUERIES AND OFFLINE QUANTUM COMPUTATIONS

Akinori Hosoyamada

Researcher
NTT Secure Platform Laboratories

Cryptanalysis against Symmetric-Key Schemes with Online Classical Queries and Offline Quantum Computations

Akinori Hosoyamada Yu Sasaki

NTT Secure Platform Laboratories

Outline

- Backgrounds
- Classical Online-Offline MITM attacks
- MITM attacks with Online Classical Queries and Offline Quantum Computations
- Applications
- Summary

Outline

- **Backgrounds**
- Classical Online-Offline MITM attacks
- MITM attacks with Online Classical Queries and Offline Quantum Computations
- Applications
- Summary

Quantum attacks on Symmetric Key Schemes

- **Some Symmetric key schemes are also broken in poly-time by quantum computers** <u>in some specific situations</u>

- Even-Mansour
- Chaskey
- Minalpher-MAC
- Full-state keyed sponge

- CBC-like MAC
- PMAC-like MAC
- LightMAC
- 3R-Feistel
- LRW, XEX, XE
- Chaskey-B

Quantum attacks on Symmet

Depends on attack models

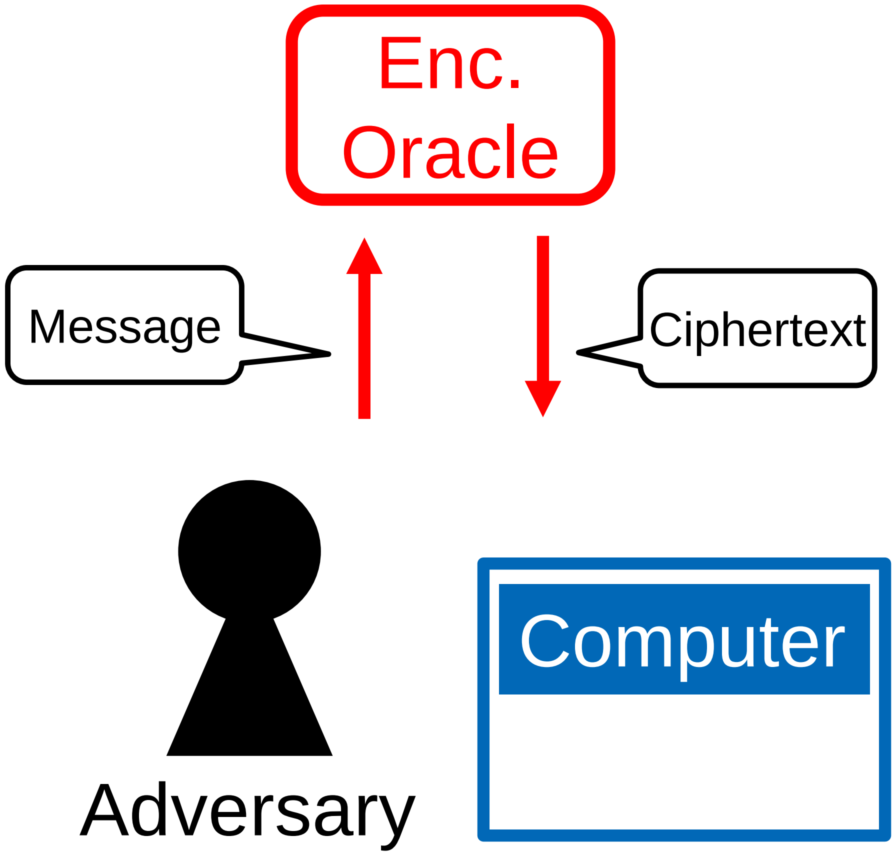
- **Some Symmetric key schemes are also broken in poly-time by quantum computers <u>in some specific situations</u>**

- Even-Mansour
- Chaskey
- Minalpher-MAC
- Full-state keyed sponge

- CBC-like MAC
- PMAC-like MAC
- LightMAC
- 3R-Feistel
- LRW, XEX, XE
- Chaskey-B

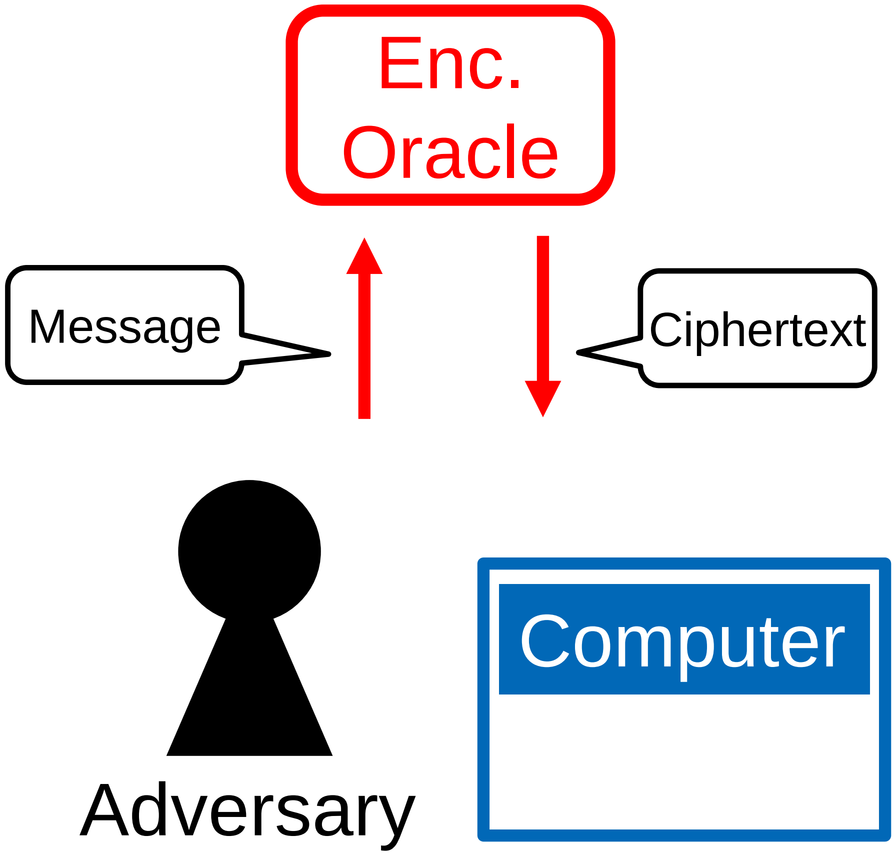
Attack Models

Chosen Plaintext Attack

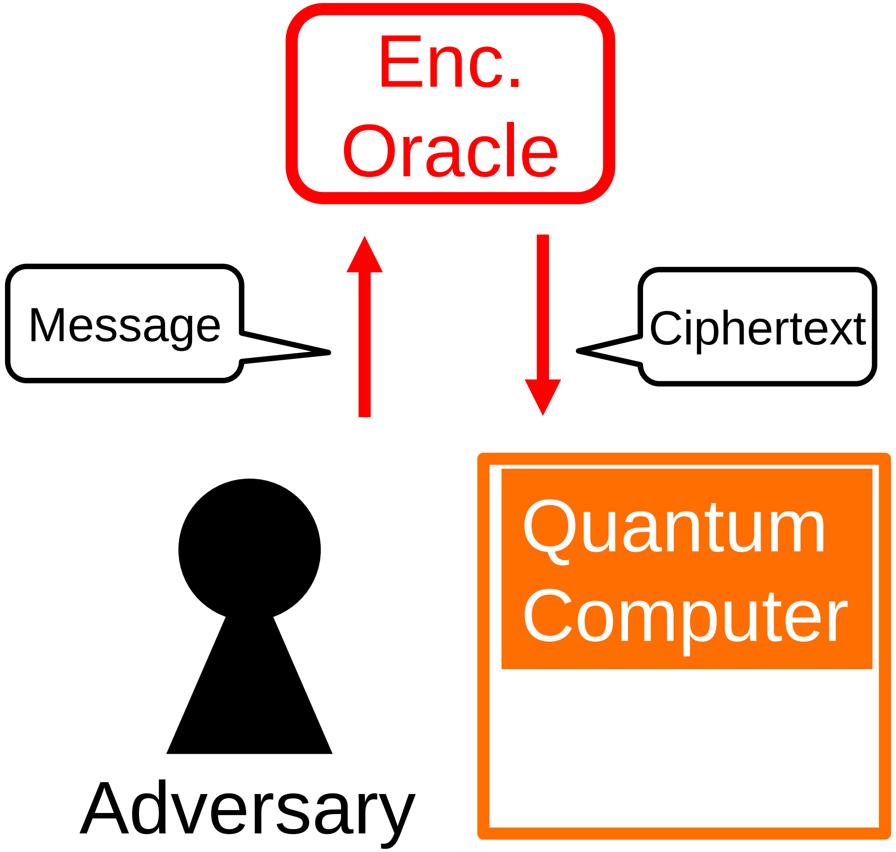


Attack Models

Chosen Plaintext Attack

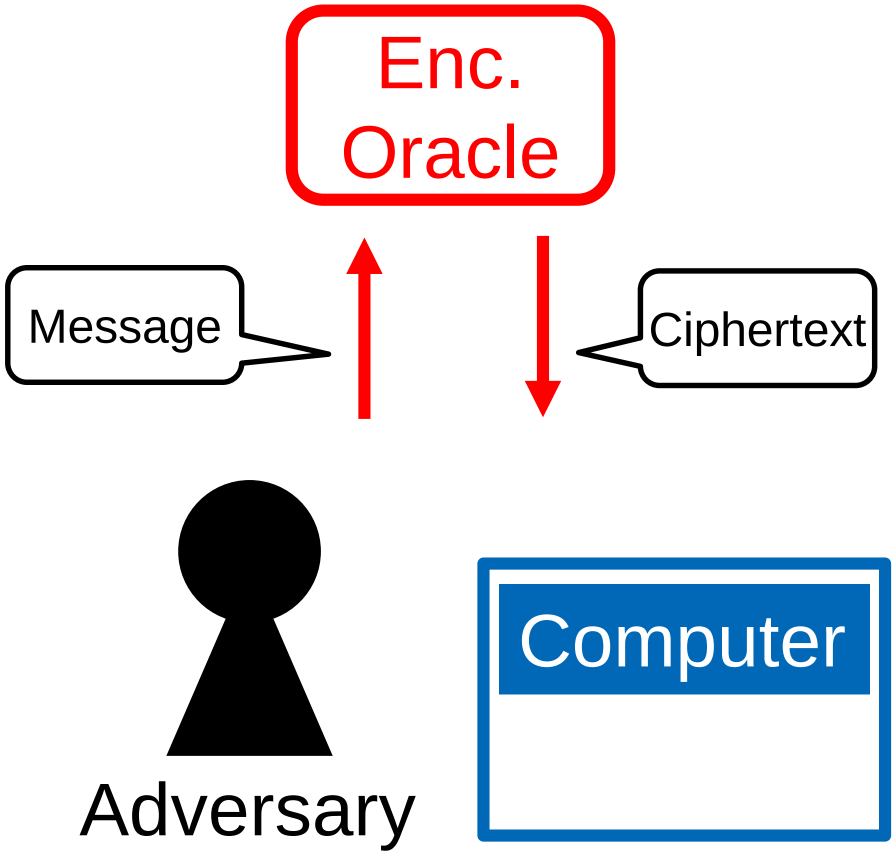


Chosen Plaintext Attack
Q1 model, classical query

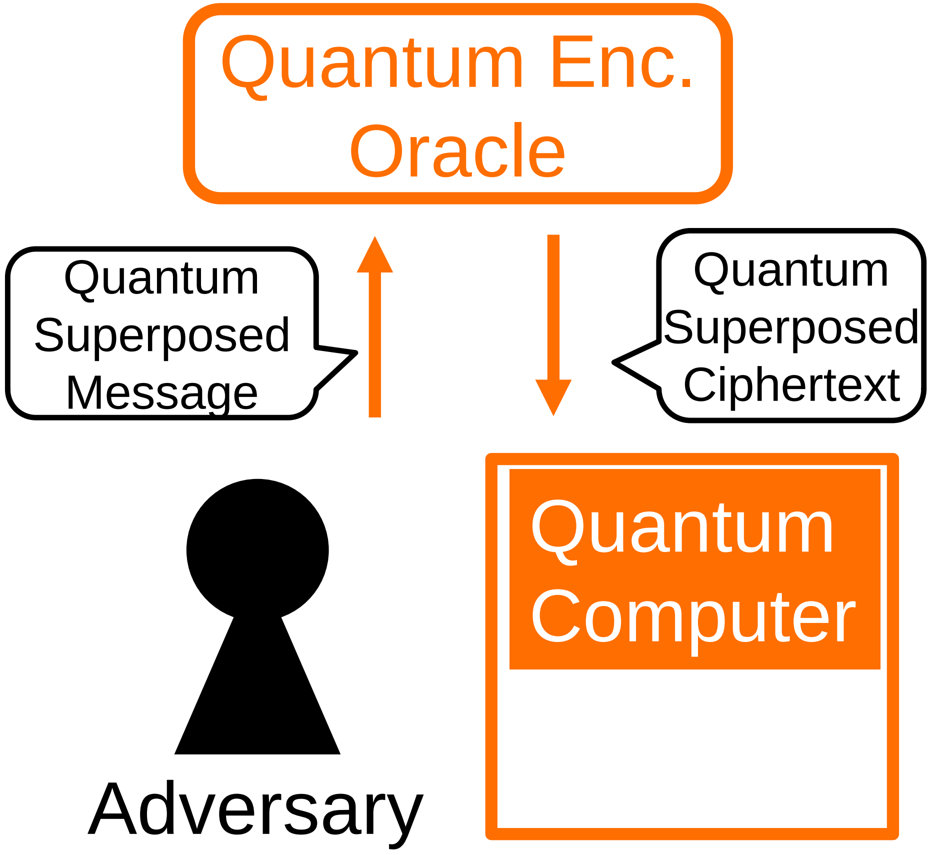


Attack Models

Chosen Plaintext Attack



Chosen Plaintext Attack
Q2 model, quantum query



Poly-time attacks are in Q2 model (quantum superposition query attack)

$\mathrm{Class}^{\mathrm{Q2}}_{\mathrm{Poly}}$: $O(n)$ quantum queries

- Even-Mansour
- Chaskey
- Minalpher-MAC
- Full-state keyed sponge
- CBC-like MAC
- PMAC-like MAC
- LightMAC
- 3R-Feistel
- LRW, XEX, XE
- Chaskey-B

Poly-time attacks are in Q2 model (quantum superposition query attack)

$\mathrm{Class}^{\mathrm{Q2}}_{\mathrm{Poly}}$: $O(n)$ quantum queries

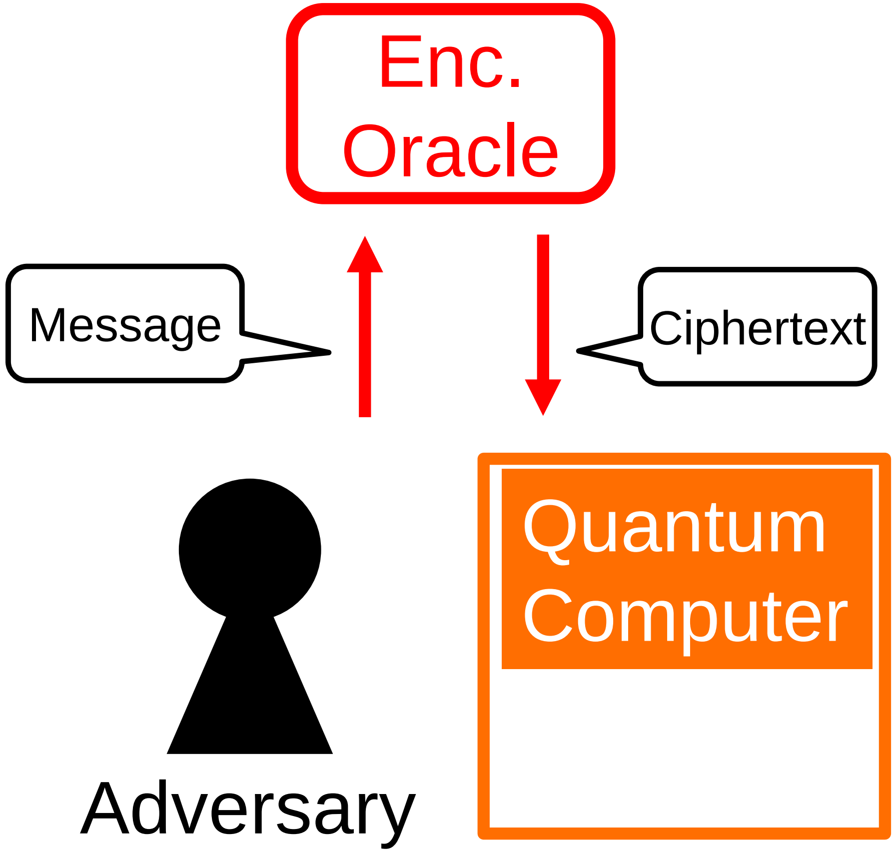
- Even-Mansour
- Chaskey
- Minalpher-MAC
- Full-state keyed sponge
- CBC-like MAC
- PMAC-like MAC
- LightMAC
- 3R-Feistel
- LRW, XEX, XE
- Chaskey-B

- Poly-time attacks can be realized in Q2 model
- Q2 model is theoretically interesting
- However, Q1 is more realistic model than Q2
- Q1 should receive much more attention…

Attack Models

We focus on Q1 model

Chosen Plaintext Attack
Q1 model, classical query



Quantum Hardware Models

- **If hardware becomes large, architecture significantly affects running time of algorithms**

a. <u>free communication model</u> [Ber09,BB17]

any qubit can interact with any qubit

b. <u>realistic communication model</u> [Ber09,BB17]

a qubit can interact with only near qubits

c. <u>independent small processors without communication</u>

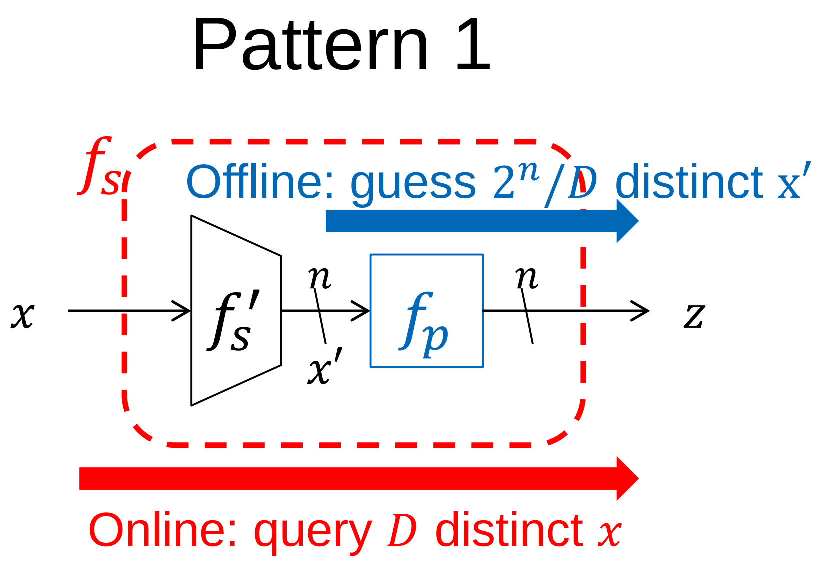
Outline

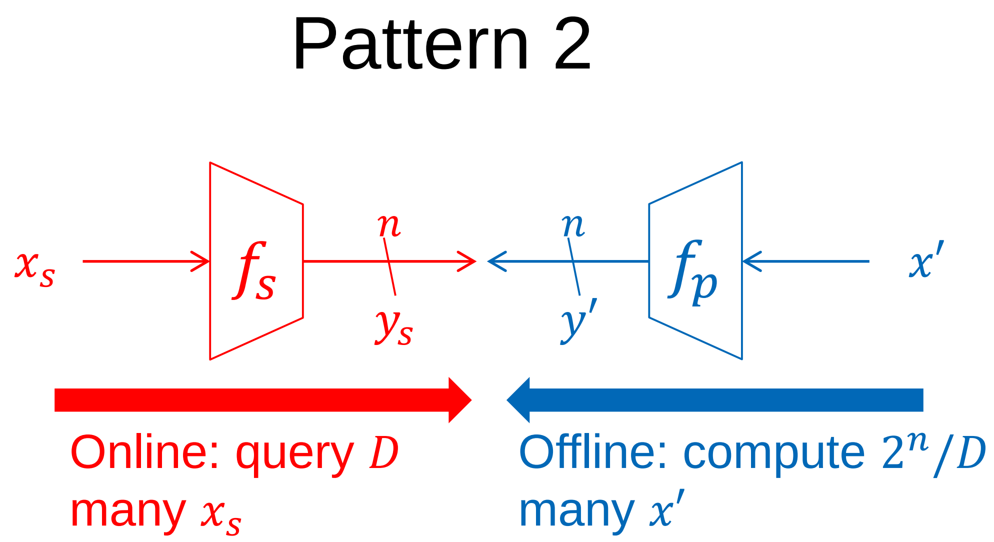
- Backgrounds
- Classical Online-Offline MITM attacks
- MITM attacks with Online Classical Queries and Offline Quantum Computations
- Applications
- Summary

Outline

- Backgrounds
- Classical Online-Offline MITM attacks
- MITM attacks with Online Classical Queries and Offline Quantum Computations
- Applications
- Summary

Classical Online-Offline MITM attacks



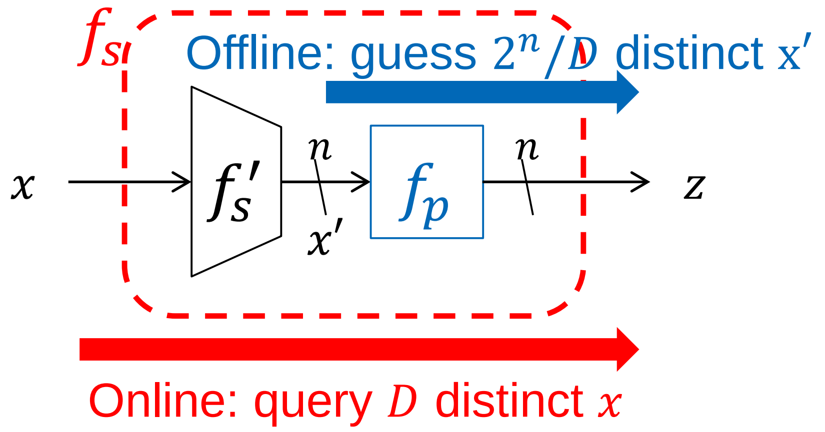


<u>Conditions to Apply On-Off MITM</u>

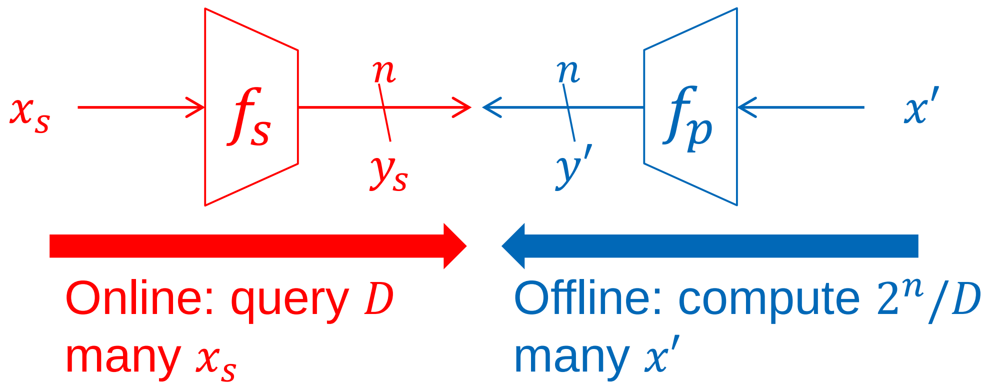
1. f_s can be calculated only by making *online* queries (f_s has some secret information)
2. f_p can be calculated *offline*
3. If we find $\boldsymbol{x}, \boldsymbol{x}'$ **s.t.** $\boldsymbol{f_s(x) = f_p(x')}$, then we can get some secret information on a crypto scheme

Classical Online-Offline MITM attacks

Pattern 1



Pattern 2



Attack Procedure

Goal: To find x, x' s.t. $f_s(x) = f_p(x')$ (then we can get some information)

1. Online phase: Collect D pairs $(x_1, f_s(x_1)), \ldots, (x_D, f_s(x_D))$
2. Offline phase: Find x' s.t. $f_p(x') = f_s(x_i)$ for some $1 \leq i \leq D$

(D-multi-target preimage search on f_p)

Step 2 requires time $T = 2^n/D$ (tradeoff: $T \cdot D = 2^n$)

NTT

Classical Online-Offline MITM attacks

Example:

$$D = 2^{n/2} \Rightarrow T = 2^{n/2}$$
$$D = 2^{n/3} \Rightarrow T = 2^{2n/3}$$

Attack Procedure

Goal: To find x, x' s.t. $f_s(x) = f_p(x')$ (then we can get some information)

1. Online phase: Collect D pairs $(x_1, f_s(x_1)), \ldots, (x_D, f_s(x_D))$
2. Offline phase: Find x' s.t. $f_p(x') = f_s(x_i)$ for some $1 \leq i \leq D$
   (D-multi-target preimage search on f_p)

Step 2 requires time $T = 2^n/D$ (tradeoff: $T \cdot D = 2^n$)

Classical Online-Offline MITM attacks

Pattern 1 Pattern 2

Attack Procedure

Example:

$$D = 2^{n/2} \Rightarrow T = 2^{n/2}$$
$$D = 2^{n/3} \Rightarrow T = 2^{2n/3}$$

Some schemes claim security up to $T \approx 2^{2n/3}$ (BBB security) by limiting D to be $< 2^{n/3}$

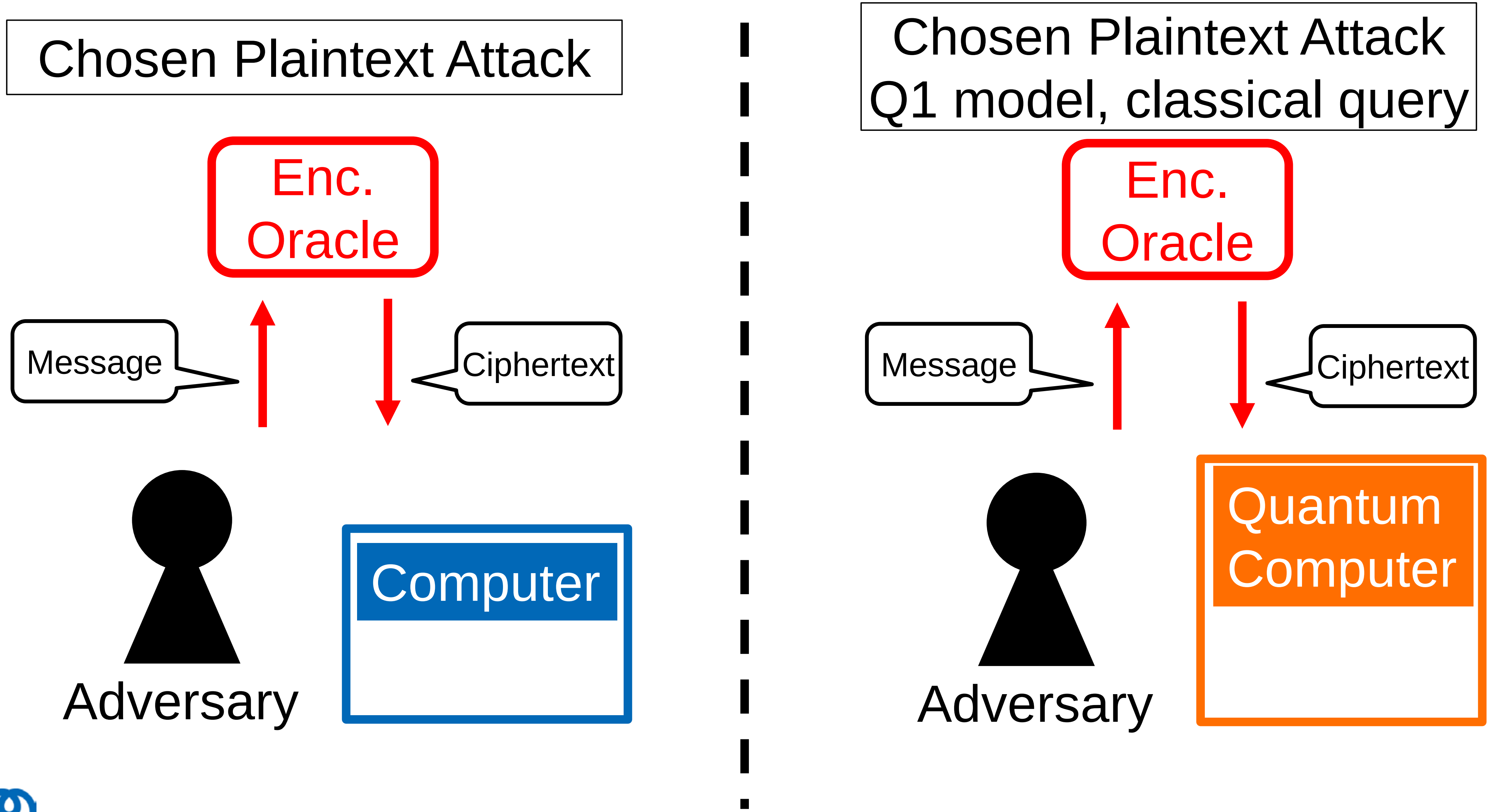
Outline

- Backgrounds
- Classical Online-Offline MITM attacks
- MITM attacks with Online Classical Queries and Offline Quantum Computations
- Applications
- Summary

Outline

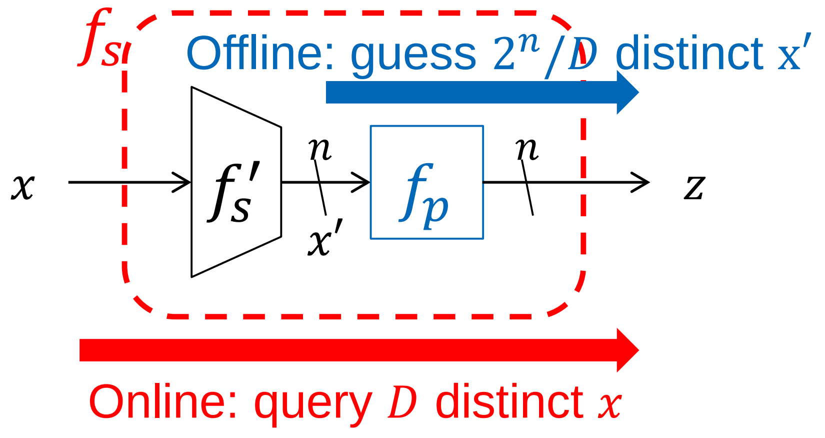
- Backgrounds
- Classical Online-Offline MITM attacks
- **MITM attacks with Online Classical Queries and Offline Quantum Computations**
- Applications
- Summary

Attack Models

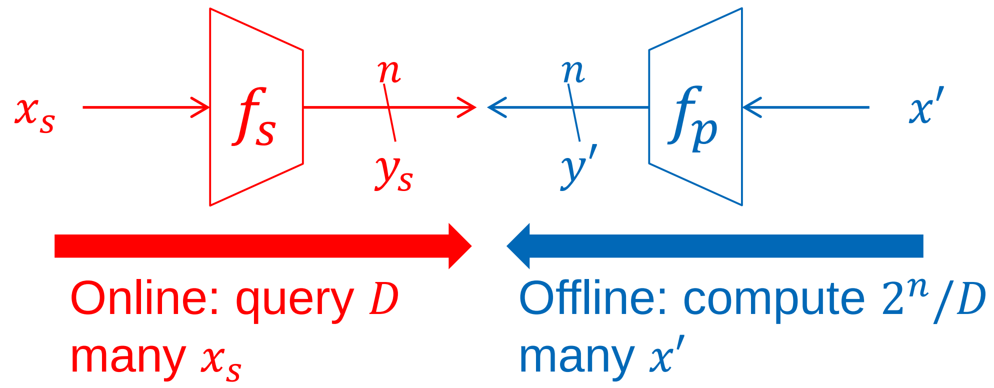


MITM attacks : Online Classical Queries and Offline Quantum Computations

Pattern 1



Pattern 2



Attack Procedure

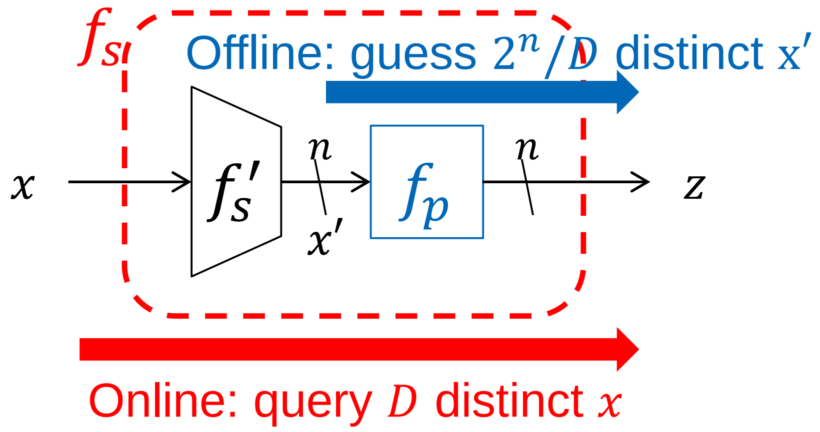
Goal: To find x, x' s.t. $f_s(x) = f_p(x')$ (then we can get some information)

1. Online phase: Collect D pairs $(x_1, f_s(x_1)), \dots, (x_D, f_s(x_D))$
2. Offline phase: Find x' s.t. $f_p(x') = f_s(x_i)$ for some $1 \le i \le D$ (D-multi-target preimage search on f_p)

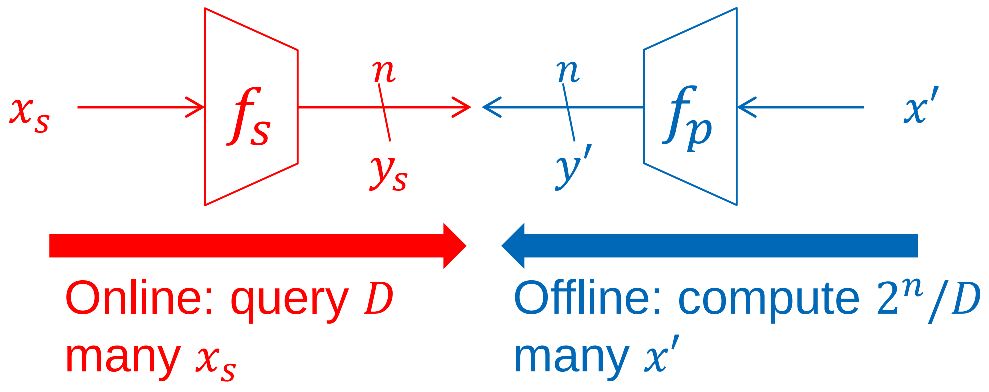
Step 2 requires time $T = 2^n/D$ (tradeoff: $T \cdot D = 2^n$)

MITM attacks : Online Classical Queries and Offline Quantum Computations

Pattern 1



Pattern 2



Attack Procedure

Goal: To find x, x' s.t. $f_s(x) = f_p(x')$ (then we can get some information)

1. Online phase: Collect D pairs $(x_1, f_s(x_1)), \ldots, (x_D, f_s(x_D))$
2. Offline phase: Find x' s.t. $f_p(x') = f_s(x_i)$ for some $1 \le i \le D$
   (D-multi-target preimage search on f_p)

Step 2 can be accelerated!! (we obtain new tradeoff!!)

MITM attacks : Online Classical Queries and Offline Quantum Computations

Some schemes claim security up to $T \approx 2^{2n/3}$ (BBB security) by limiting D to be $< 2^{n/3}$

(D-multi-target preimage search on f_p)

Step 2 can be accelerated!! (we obtain new tradeoff!!)

MITM attacks : Online Classical Queries and Offline Quantum Computations

f_S

x

x'

On

ute $2^n/D$

Some schemes claim security up to $T \approx 2^{2n/3}$
(BBB security) by limiting D to be $< 2^{n/3}$

But

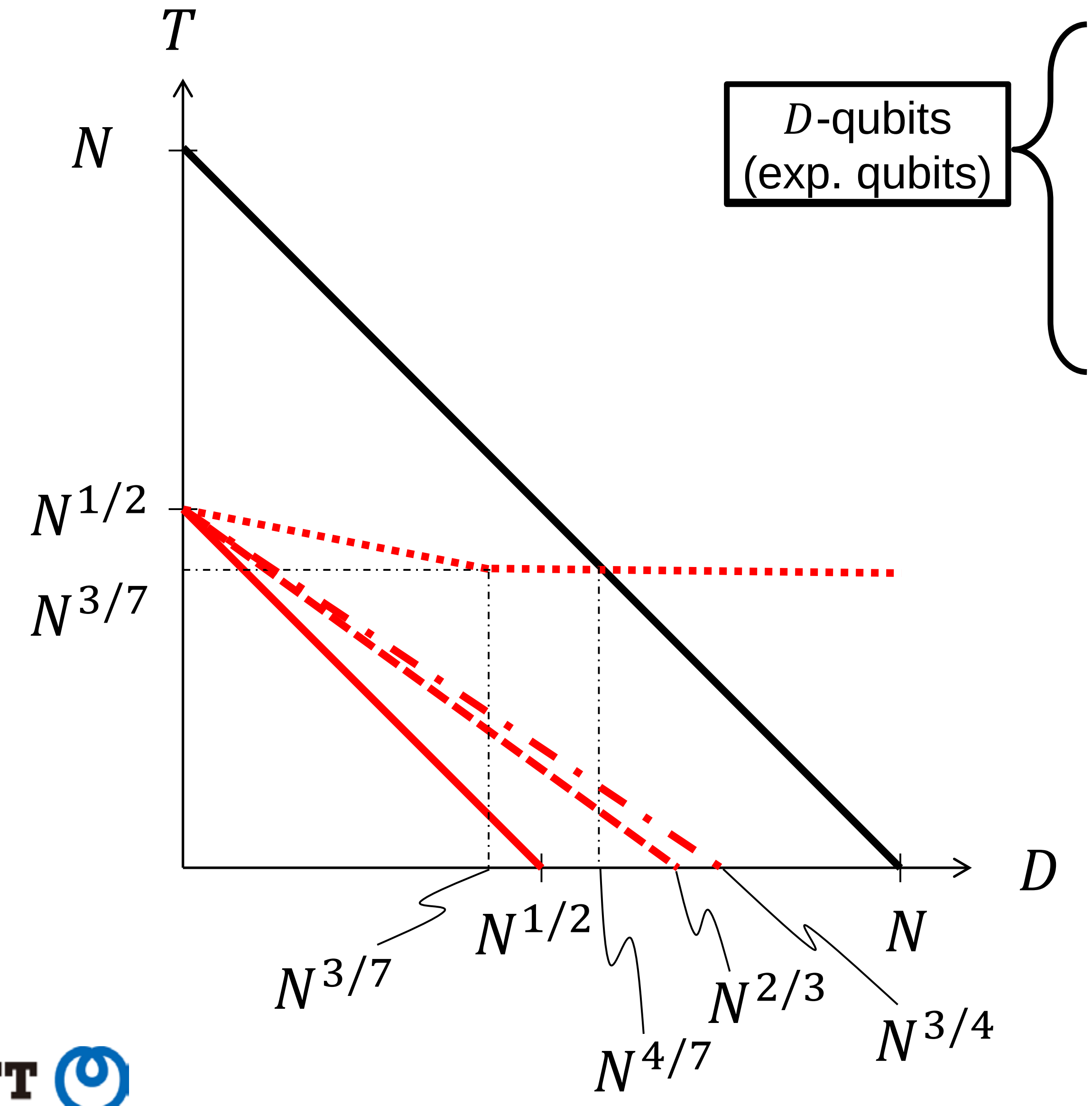
Those claims are broken in the quantum settings
due to new tradeoffs

2

(D-multi-target preimage search on f_p)

Step 2 can be accelerated!! (we obtain new tradeoff!!)

MITM attacks in the quantum settings: 4 new tradeoffs between T and D



- Free communication model, $D^2 \cdot T^2 = N$ [BB17]
- Realistic communication model, $D^{3/2} \cdot T^2 = N$ [BB17]
- Any (independent small processors), $D^4 \cdot T^6 = N^3$ [CNPS17]
- Poly-qubits, $D \cdot T^6 = N^3$[CNPS17]
- (Classical, $D \cdot T = N$)

Offline phase of MITM attack is accelerated by quantum multi-target preimage search

Outline

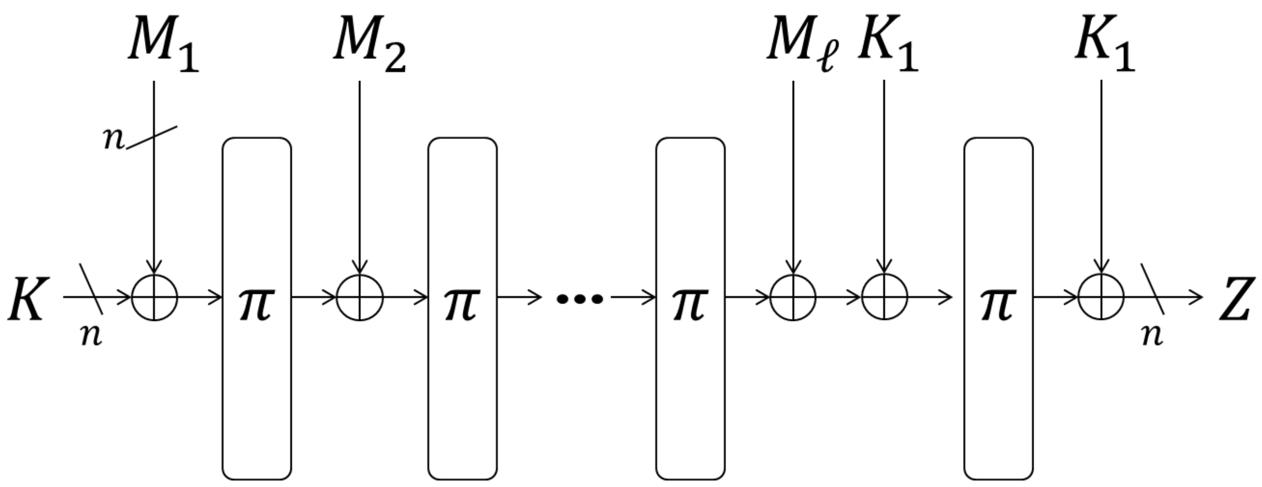
- **Backgrounds**
- **Classical MITM attacks**
- **MITM attacks with Online Classical Queries and Offline Quantum Computations**
- **Applications**
- **Summary**

Outline

- Backgrounds
- Classical MITM attacks
- MITM attacks with Online Classical Queries and Offline Quantum Computations
- **Applications**
- Summary

Attack on Chaskey

- **Chaskey[Mou15] is a lightweight MAC**



Classical online-offline MITM attack
can be applied to Chaskey
(Classical tradeoff: $T \cdot D = 2^{128}$ if n=128)

Attack on Chaskey

- **Chaskey[Mou15] is a lightweight MAC**

Chaskey claims 80-bit security
by restricting D to be $< 2^{48}$
It claims security up to $\mathrm{T} \approx 2^{80}$
(n is set as n=128)

Classical online-offline MITM attack
can be applied to Chaskey
(Classical tradeoff: $T \cdot D = 2^{128}$ if n=128)

Attack on Chaskey

- If $D < 2^{48}$ queries are allowed, then attack complexity becomes…

	T	D	Q	M
Classical	2^{80}	2^{48}	–	2^{48}
Case 1a (Exp. qubits, free communication)	2^{32}	2^{32}	2^{32}	2^{32}
Case 1b (Exp. qubits, realistic communication)	2^{37}	2^{37}	2^{37}	2^{37}
Case 1c (Exp. qubits, any communication)	2^{39}	2^{39}	2^{39}	2^{39}
Case 2 (Poly. qubits)	2^{56}	2^{48}	(2^{7})	2^{16}

T is overwhelmingly smaller than 2^{80} of classical attack

NTT

Applications to various schemes

$\mathrm{Class}^{\mathrm{Q2}}_{\mathrm{Poly}}$: $O(n)$ quantum queries

- Even-Mansour
- Chaskey
- Minalpher-MAC
- Full-state keyed sponge

- CBC-like MAC
- PMAC-like MAC
- LightMAC
- 3R-Feistel
- LRW, XEX, XE
- Chaskey-B

Applications to various schemes

$\text{Class}^{\text{Q2}}_{\text{Poly}}$: $O(n)$ quantum queries

- Even-Mansour
- Chaskey
- Minalpher-MAC
- Full-state keyed sponge

- CBC-like MAC
- PMAC-like MAC
- LightMAC
- 3R-Feistel
- LRW, XEX, XE
- Chaskey-B

$\text{Class}^{\text{Q1}}_{\text{Exp}}$: below $O(2^{n/2})$ classical queries

Applications to various schemes

$\mathrm{Class}^{\mathrm{Q2}}_{\mathrm{Poly}}$: $O(n)$ quantum queries

- TDR
- McOE-X
- H^2MAC, LPMAC
- Keyed sponge
- KMAC

- Even-Mansour
- Chaskey
- Minalpher-MAC
- Full-state keyed sponge

- CBC-like MAC
- PMAC-like MAC
- LightMAC
- 3R-Feistel
- LRW, XEX, XE
- Chaskey-B

$\mathrm{Class}^{\mathrm{Q1}}_{\mathrm{Exp}}$: below $O(2^{n/2})$ classical queries

Applications to various schemes

$\text{Class}^{\text{Q2}}_{\text{Poly}}$: $O(n)$ quantum queries

- TDR
- McOE-X
- H^2MAC, LPMAC
- Keyed sponge
- KMAC

- Even-Mansour
- Chaskey
- Minalpher-MAC
- Full-state keyed sponge

- CBC-like MAC
- PMAC-like MAC
- LightMAC
- 3R-Feistel
- LRW, XEX, XE
- Chaskey-B

$\text{Class}^{\text{Q1}}_{\text{Exp}}$: below $O(2^{n/2})$ classical queries

Others

- FX-constructions

Outline

- Backgrounds
- Classical MITM attacks
- MITM attacks with Online Classical Queries and Offline Quantum Computations
- Applications
- Summary

Outline

Innovative R&D by NTT

- Backgrounds
- Classical MITM attacks
- MITM attacks with Online Classical Queries and Offline Quantum Computations
- Applications
- Summary

Summary

- MITM attack with Online Classical and Offline quantum computations (quantum attack in Q1 model)
- New tradeoffs between D and T in 4 models
- Some existing schemes claim BBB security on T by limiting the maximum number of D following the classical tradeoff DT=N, but such claims are broken by our attacks

Thank you!!

E-mail: hosoyamada.akinori@lab.ntt.co.jp

#RSAC

RSA Conference 2018

San Francisco | April 16 – 20 | Moscone Center

NOW MATTERS

SESSION ID: CRYP-W14

IMPROVING STATELESS HASH-BASED SIGNATURES

Guillaume Endignoux

Software Engineer
Google

Improving Stateless Hash-Based Signatures

CT-RSA 2018

Jean-Philippe Aumasson[1], Guillaume Endignoux[2]

Wednesday 18th April, 2018

[1]Kudelski Security

[2]Work done while at Kudelski Security and EPFL

Hash-based signatures

What are hash-based signatures?

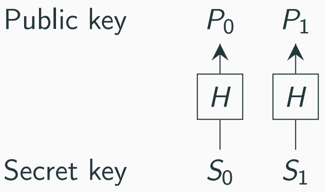
- Good hash functions are hard to invert = *preimage-resistance*.
- We can use this property to create signature schemes[1].

[1] Whitfield Diffie and Martin E. Hellman. *New directions in cryptography.* 1976

Hash-based signatures

What are hash-based signatures?

- Good hash functions are hard to invert = *preimage-resistance*.
- We can use this property to create signature schemes[1].



First step: scheme to sign 1-bit message.

- Key generation: commit to 2 secrets with H
- Sign bit b: reveal $\sigma = S_b$
- Verify signature σ: compare $H(\sigma)$ with P_b

[1] Whitfield Diffie and Martin E. Hellman. *New directions in cryptography.* 1976

Hash-based signatures

Second step: sign n-bit message $\Rightarrow$ n copies of the previous scheme.

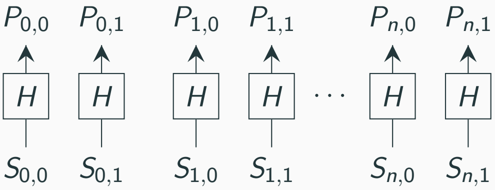


Figure 1: Lamport signatures.

Hash-based signatures

Second step: sign n-bit message $\Rightarrow$ n copies of the previous scheme.

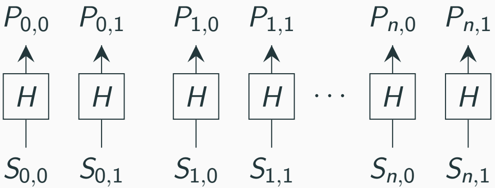


Figure 1: Lamport signatures.

However, this is a **one-time** signature scheme.

Hash-based signatures

More constructions:

- **WOTS** (Winternitz one-time signatures) = compact version of the n-bit message scheme.
- **Merkle trees** = *stateful* multiple-time signatures.
- **HORS** = *stateless* few-time signatures.
- **HORST** = HORS with Merkle tree.

Hash-based signatures

SPHINCS = stateless many-time signatures (up to 2^{50} messages).

- Hyper-tree of WOTS signatures $\approx$ certificate chain
- Hyper-tree of height $H = 60$, divided in 12 layers of {Merkle tree + WOTS}

Sign message M:

- Select index $0 \leq i < 2^{60}$
- Sign M with i-th HORST instance
- Chain of WOTS signatures.

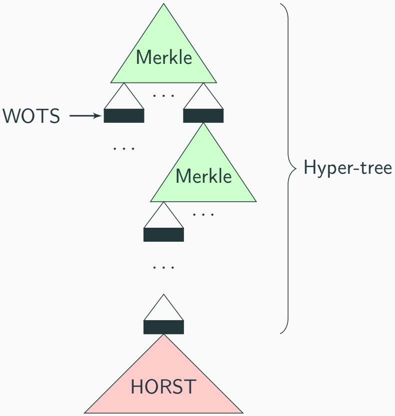


Figure 2: SPHINCS.

Hash-based signatures

Hash-based signatures in a nutshell:

- Post-quantum security well understood $\Rightarrow$ **Grover's algorithm**: preimage-search in $O(2^{n/2})$ instead of $O(2^n)$ for n-bit hash function.
- Signature size is quite large: 41 KB for SPHINCS (stateless), 8 KB for XMSS (stateful).

Contributions

We propose improvements to **reduce signature size** of SPHINCS:

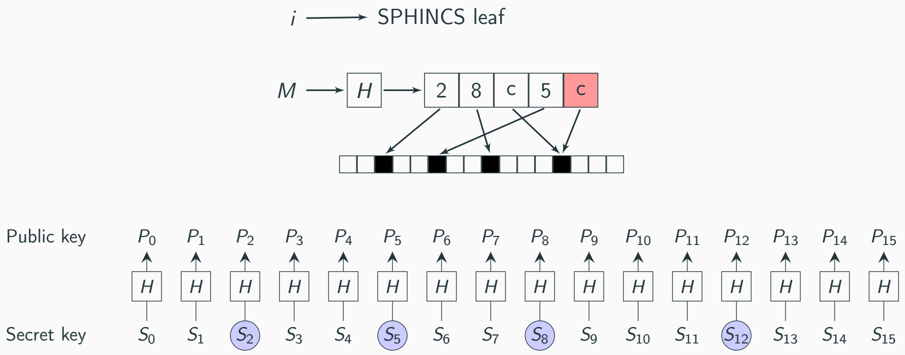
- PRNG to obtain a random subset (PORS)
- Octopus: optimized multi-authentication in Merkle trees
- Secret key caching
- Non-masked hashing

PRNG to obtain a random subset

From HORS to PORS

Sign a message M with HORS:

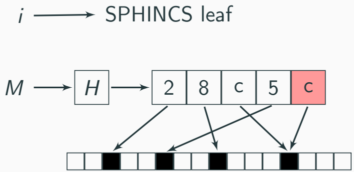
- Hash the message $H(M) = 28\text{c}5\text{c}...$
- Split the hash to obtain indices $\{2, 8, c, 5, c, \ldots\}$ and reveal values $S_2, S_8, \ldots$



From HORS to PORS

Sign a message M with HORS:

- Hash the message $H(M) = $ 28c5c...
- Split the hash to obtain indices $\{2, 8, c, 5, c, \ldots\}$ and reveal values $S_2, S_8, \ldots$



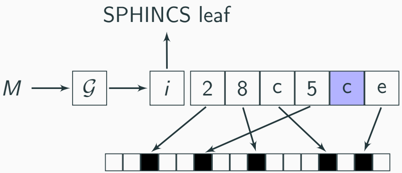
Problems:

- Some indices may be the same $\Rightarrow$ fewer values revealed $\Rightarrow$ lower security...
- Attacker is free to choose the hyper-tree index $i \Rightarrow$ larger attack surface.

From HORS to PORS

PORS = PRNG to obtain a random subset.

- Seed a PRNG from the message.
- Generate the hyper-tree index.
- Ignore duplicated indices.



Significant security improvement for the same parameters!

From HORS to PORS

Advantages of PORS:

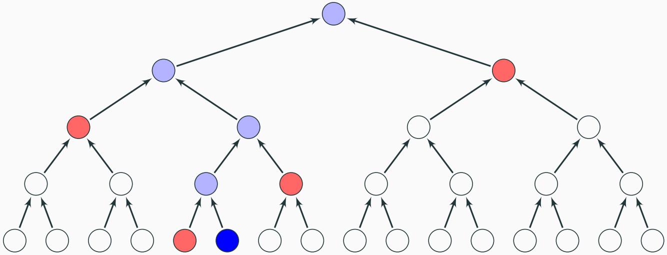
- Significant security improvement for the same parameters!
- Smaller hyper-tree than SPHINCS for same security level $\Rightarrow$ Signatures are **4616 bytes** smaller.
- Performance impact of PRNG vs. hash function is negligible $\Rightarrow$ For SPHINCS, generate only 32 distinct values.

Octopus: multi-authentication in Merkle trees

Octopus

Merkle tree of height h = compact way to authenticate any of 2^h values.

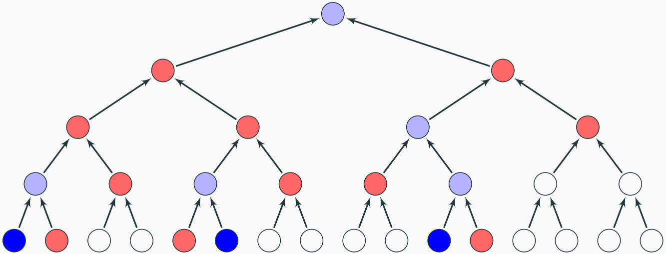
- Small public value = root
- Small proofs of membership = h authentication nodes



Octopus

How to authenticate k values?

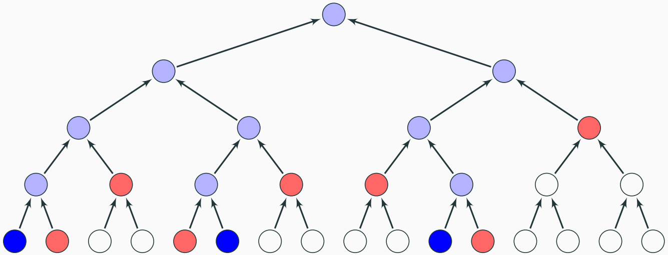
- Use k independent proofs $= kh$ nodes.
- This is suboptimal! Many redundant values...



Octopus

How to authenticate k values?

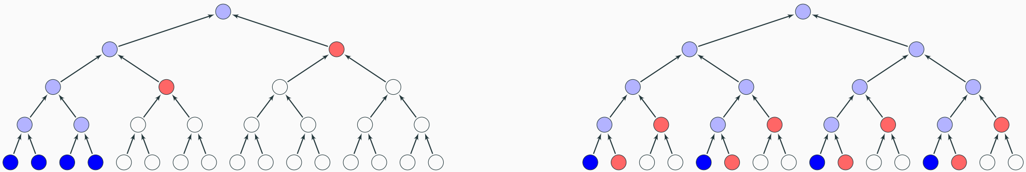
- Optimal solution: compute smallest set of authentication nodes.



Octopus

How many bytes does it save?

- It depends on the shape of the "octopus"!
- Examples for $h = 4$ and $k = 4$: between 2 and 8 authentication nodes.



Theorem

Given a Merkle tree of height h and k leaves to authenticate, the minimal number of authentication nodes n verifies:

$$h - \lceil \log_2 k \rceil \leq n \leq k(h - \lfloor \log_2 k \rfloor)$$

$\Rightarrow$ For $k > 1$, this is always better than the kh nodes for k independent proofs!

Octopus

In the case of SPHINCS, $k = 32$ **uniformly distributed leaves**, tree of height $h = 16$.

In our paper, recurrence relation to compute **average** number of authentication nodes.

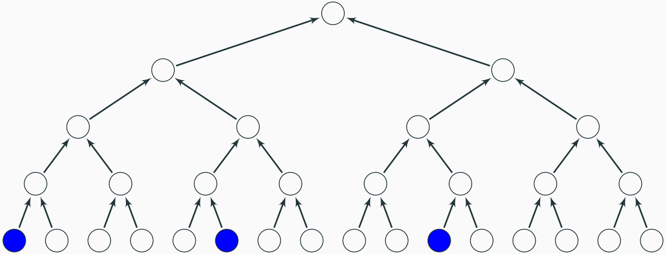
Method	Number of auth. nodes
Independent proofs	512
SPHINCS[2]	384
Octopus (worst case)	352
Octopus (average)	324

$\Rightarrow$ Octopus authentication saves **1909 bytes** for SPHINCS signatures on average.

[2] SPHINCS has a basic optimization to avoid redundant nodes close to the root.

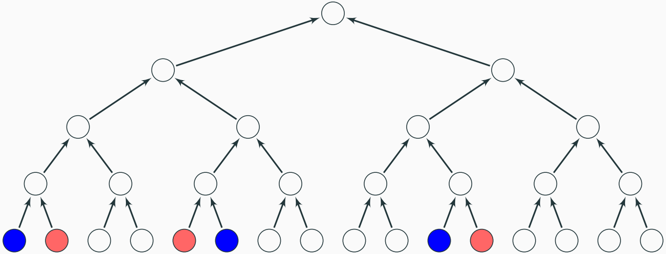
Octopus algorithm

- Bottom-up algorithm to compute the optimal authentication nodes.
- Formal specification in the paper, let's see an example.



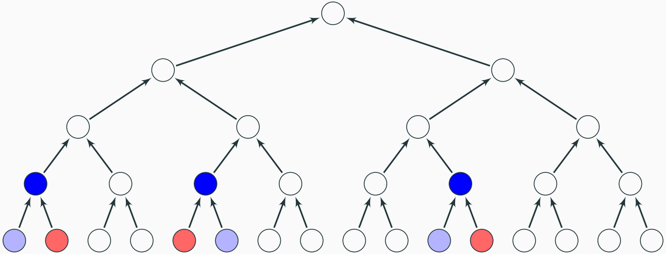
Octopus algorithm

- Bottom-up algorithm to compute the optimal authentication nodes.
- Formal specification in the paper, let's see an example.



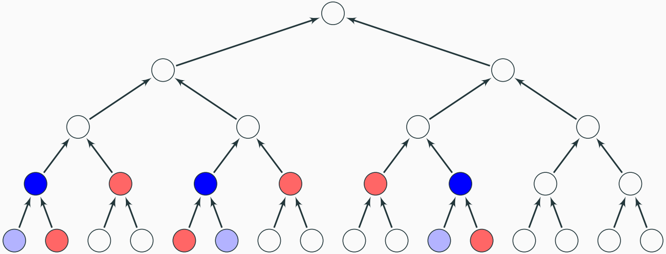
Octopus algorithm

- Bottom-up algorithm to compute the optimal authentication nodes.
- Formal specification in the paper, let's see an example.



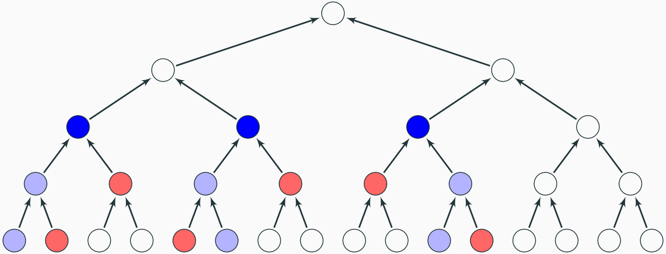
Octopus algorithm

- Bottom-up algorithm to compute the optimal authentication nodes.
- Formal specification in the paper, let's see an example.



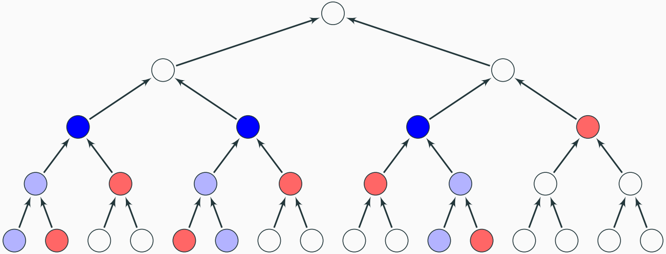
Octopus algorithm

- Bottom-up algorithm to compute the optimal authentication nodes.
- Formal specification in the paper, let's see an example.



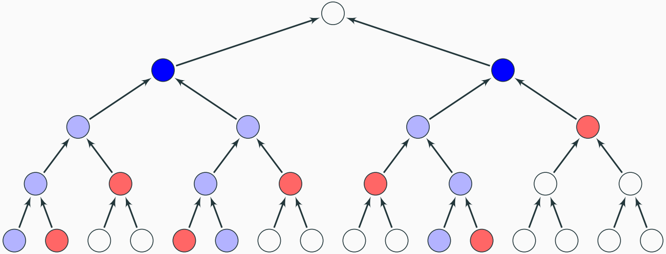
Octopus algorithm

- Bottom-up algorithm to compute the optimal authentication nodes.
- Formal specification in the paper, let's see an example.



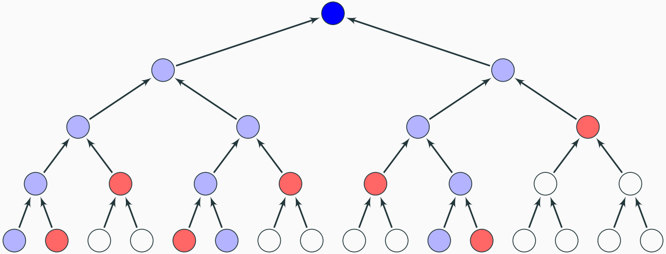
Octopus algorithm

- Bottom-up algorithm to compute the optimal authentication nodes.
- Formal specification in the paper, let's see an example.



Octopus algorithm

- Bottom-up algorithm to compute the optimal authentication nodes.
- Formal specification in the paper, let's see an example.



Conclusion

Take-aways

- Octopus + PORS = great improvement over HORST.
- These modifications are simple to understand $\Rightarrow$ low risk of implementation bugs.
- More improvements in the paper.

Implementation

Two open-source implementations:

- Reference C implementation, proposed for NIST pqcrypto standardization
  `https://github.com/gravity-postquantum/gravity-sphincs`
- Rust implementation with focus on clarity and testing
  `https://github.com/gendx/gravity-rs`

Conclusion

Thank you for your attention!

Secret key caching

WOTS signatures to "connect" Merkle trees are large ($\approx$ 2144 bytes per WOTS).

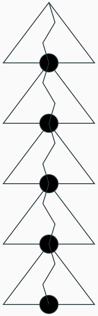


Figure 3: SPHINCS.

Secret key caching

⇒ We use a **larger root Merkle tree**, and cache more values in private key.

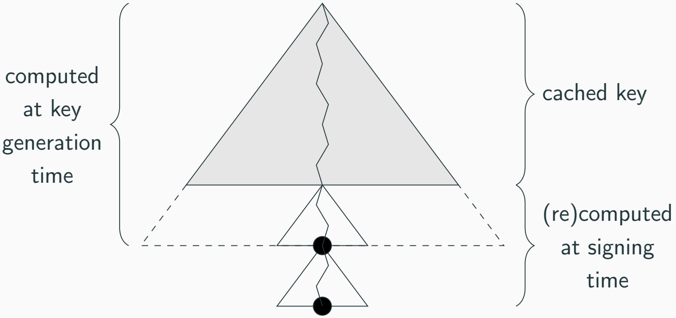
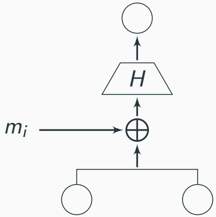


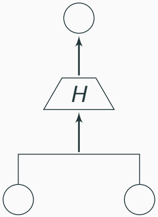
Figure 4: Secret key caching.

Non-masked hashing

- In SPHINCS, Merkle trees have a **XOR-and-hash** construction, to use a 2nd-preimage-resistant hash function H.
- Various masks, depending on location in hyper-tree; all stored in the public key.
- Post-quantum preimage search is faster with Grover's algorithm $\Rightarrow$ We remove the masks and rely on **collision-resistant** H.



(a) Masked hashing in SPHINCS.



(b) Mask off.