Doctoral Thesis

A Study on Hash-based Signature Schemes

Supervisor: Masayuki Abe & Mehdi Tibouchi

Department of Social Informatics

Graduate School of Informatics

Kyoto University
Japan

Quan Yuan

Submitted on August 30, 2022

A Study on Hash-based Signature Schemes

Quan Yuan

Abstract

This dissertation studies the security of hash-based signature (HBS) schemes, whose security is solely based on security notions for hash functions.

A signature scheme is an essential cryptographic primitive for authentication and is widely used in a large number of protocols. In the post-quantum settings, HBS schemes are considered attractive choices due to their efficiency, flexibility, and minor assumptions.

This dissertation covers three contributions to the security analysis of HBS schemes.

The first result is on *subset-resilient hash function families* (SRH). SRH is an essential building block of HBS schemes but lacking in research. We research subset resilience in three directions. First, we propose a generic quantum attack on subset resilience, implying an upper bound of the generic security. Second, we prove the relationship between subset resilience and decisional collision resistance, another security notion for hash functions. The existence of SRH implies the existence of decisional-collision-resistant hash function families. Third, we prove the impossibility of constructing an SRH from one-way permutations in a fully black-box manner.

The second result is on security notions for stateful signature schemes. A large number of HBS schemes are stateful, meaning that the signer needs to maintain a dynamic state in issuing signatures. The classical security notions for signature schemes only consider the case that the state is invisible to the adversary. In practice, there are some situations where the state is vulnerable, meaning that it can be visible or even be modified in some cases. In our research, we give some fine-grained security notions for stateful signature schemes and show some separation and general conversions among them. We remark that the conversions also work on signature schemes other than hash-based ones.

The third result is on quantum-access security of stateless HBS schemes. Usually, signature schemes are required to be secure against chosen message attacks. Any polynomial-time (quantum) adversary cannot forge a signature given access to the signing oracle that can sign any message. Note that here the signing oracle is classical, meaning that it can only sign one message for each query. If the signing oracle becomes quantum, the case will be completely different. Our research gives some positive and negative results on quantum-access security of HBS schemes. First, we give generic security bounds for few-time stateless HBS schemes (including classical security and quantum-access security). Second, we show some quantum-access attacks on existing many-time stateless HBS schemes. The time complexity is much lower than that of optimal classical chosen message attacks, implying that quantum-access attacks are more threatening than classical ones to these schemes. Second, we propose a variant of SPHINCS+ and give security proof against quantum-access attacks. As far as we know, it is the first practical stateless HBS scheme against quantum-access attacks with provable security.

Acknowledges

First and foremost, the author would like to show his best appreciation to his supervisors, Professor Masayuki Abe and Professor Mehdi Tibouchi. Every work in this dissertation was done under their patient supervision. The author also would like to thank his advisors, Professor Takayuki Kanda and Professor Tsuyoshi Takagi, for giving valuable comments and advice.

Contents

Chapter 1

Introduction

1.1 Hash-based Signature Schemes

A (digital) signature scheme [41] is one of the essential primitives in cryptography. It is inspired by signatures in the real world. In our society, signatures are widely used in various situations requiring authentication. For instance, when a person signs a message (or a document), the signature provides the authentication that it is the signer who writes (or at least acknowledges) the message. The signature cannot be forged or denied. One can verify it by checking the signature sample stored in the library. If the handwriting is similar, we say that the signature is verified to be valid, and we confirm that it is issued by the signer.

In modern society, we cannot require all signatures to be handwritten. Indeed, the signatures should be signed by machines automatically in many cases. Signature schemes are algorithms that can issue signatures. A signature scheme consists of three polynomial-time algorithms: the key generation algorithm, the signing algorithm, and the verification algorithm. The key generation algorithm is for generating a pair of keys: a secret key and a public key. A public key is published as a "card" of the signer (that behaves as the "sample" in the library), while the secret key must be privately stored as a "password" of the signer. The signing algorithm is for signing messages. Taking as input a message and the secret key, it outputs a signature for the message. The verification algorithm is for verifying signatures. Taking as input a message, the corresponding signature, and the public key, it outputs a bit. If the signature is correctly issued, the verification algorithm will always output 1. We say that it is a valid signature for the message.

A cryptographic scheme is meaningful only if it is secure. The security of a signature scheme is usually defined by the following experiment. First, an adversary is given a public key. Then, the adversary is given access to a signing oracle that can sign any messages. Finally, the adversary is required to output a signature for a fresh message that has not been queried to the signing oracle. If *any* polynomial-time adversary can only succeed in this experiment with a negligible probability, we say the scheme is secure, or formally *existentially unforgeable under chosen message attacks* (EU-CMA).

The security of a signature scheme is usually reduced to some mathematical problems. For example, the security of RSA-FDH signature scheme [12, 35] is based on the hardness of RSA problem. ElGamal signature scheme [47] and Schnorr signature scheme [96] is based on the hardness of discrete logarithms. Since these hard problems have been studied for a long history and there is no polynomial-time solution, we believe in the hardness of the problem and thus the security of the related schemes. In other words, suppose a signature is forged by an adversary, then the adversary can solve these hard problems with a good probability, which is infeasible. Thus, these hard problems are significant and are treated as foundations of cryptographic schemes.

In recent years, the concept of quantum computers has appeared and has had a great influence on hard problems. In 1994, Shor [99] proposes a quantum algorithm that can solve factorization (and thus RSA problem) and discrete logarithms in polynomial-time. It immediately implies the insecurity of the above signature scheme against quantum adversaries. Although there is no practical quantum computer so far, it is risky enough due to the rapid development of quantum technology.

Thus, post-quantum cryptography has become a hot topic in these years. A post-quantum signature scheme must be based on problems that remain hard against quantum adversaries. For example, Falcon [50] and CRYSTALS-Dilithium

[45] are lattice-based signature schemes. Rainbow [42] is a multivariate-based signature scheme. They are candidates in the third round of the NIST post-quantum cryptography standardization project. However, these problems have a shorter history than the classical ones. Thus, it is hard to say whether these problems will remain unsolved in the following decades.

We can try to solve this problem in other ways. Note that one-way function is one of the most fundamental concepts in cryptography. It is proven that the existence of a secure signature scheme implies the existence of one-way functions [95]. In addition, Lamport [81] proposes a one-time signature scheme that is only based on a one-way function. That is, now we *assume* that we have a one-way function (maybe the one-wayness is based on the hardness of some mathematical problems). Then, we can construct a signature scheme based on this assumption.

One may wonder how to obtain a quantum-secure one-way function. Indeed, the one-way function can be instantiated by hash functions.

A hash function is another fundamental concept in cryptography, which maps a string of arbitrary length to a string of constant length. Usually, designing a hash function is the work in symmetric-key cryptography. The circuit of a hash function usually contains various basic operations such as and, or, and xor. Complicated structures of hash functions make them behave similarly to random functions. A well-designed hash function is expected to be one-way. Thus, if we use a hash function to instantiate the one-way function in Lamport's signature scheme [81], we obtain a signature scheme whose security is not based on any mathematical problems. If the security of a signature scheme is only based on security notions of hash functions rather than any other mathematical problems, we call this scheme a *hash-based signature scheme* (HBS scheme).

An HBS scheme is very attractive in post-quantum cryptography. The advantages are as follows.

- **Flexibility.** Most signature schemes use hash functions as a building block. It implies that usually we cannot get rid of the security assumptions of hash functions in constructing a signature scheme. And an HBS scheme does not introduce any other assumptions except that the hash functions are ideal. Of course, one cannot deny that hash functions are also potentially threatened by quantum computers in the future, just like mathematical problems about lattices. However, if a hash function is broken, we can immediately replace it with another designation. It is much easier to design a new hash function than to look for a new hard mathematical problem. On the other hand, if we turn to a new mathematical hard problem, it will take a long time for cryptoghraphers to construct signatures based on it, and the work of replacing the signature scheme is also troubling. In comparison, if we choose an HBS scheme and turn to a new hash function, the structure of the signature scheme will not be changed. We only need to change the hash function as a component in the structure. It will save plenty of time and cost in this procedure.

- **Key Size.** The size of the key pair of HBS schemes is usually competitive, especially for public keys. For example, in SPHINCS+-256s/256f (the state-of-art HBS scheme providing 128-bit quantum security), the public key size is only 64 bytes, while those of FALCON and Rainbow are 1793 bytes and 1683 KB.

- **Efficiency.** Since hash functions are usually constructed by basic operations, they run fasters than multiplications and involutions needed in mathematical problems. In some cases, a hash function is hardwired in some integrated circuits. It will greatly accelerate the computation of hash functions and the corresponding HBS schemes.

The main drawback of HBS schemes is the signature size. At the same security level, the size of SPHINCS+ is about 23 times larger than that of FALCON and 140 times larger than Rainbow. For this reason, SPHINCS+ is one of the alternative candidates for the third round of NIST standard list. However, HBS schemes are still an attractive choice in post-quantum cryptography and deserve further research.

1.2 Black-box Constructions

Black-box constructions [74] are widely used in designing cryptographic primitives and schemes. For example, the security of HBS schemes is based on the security notions for hash functions, such as one-wayness (or preimage resistance) and collision resistance. That is, any hash functions that have (or are assumed to have) the properties can be used as the building blocks in the construction. Here, the hash functions are considered "black boxes", and the concrete structures are independent of the reductions. Formally, in a black-box construction of Q from P, P is given as an oracle to Q. The syntax of Q can be realized by a sufficient number of queries to P, and the security of Q can be reduced to the security of P, regardless of the structure and security basis of P.

From a one-way function, there exist black-box constructions of signature schemes [81], commitment schemes [88], pseudorandom generators [62] and pseudorandom functions [57]. (Indeed, the pseudorandom function is constructed by a pseudorandom generator in a black-box manner.) There are also black-box constructions between schemes with different security notions. For example, there exist black-box constructions of EU-CMA signature schemes from EU-RMA (existentially unforgeable under random message attacks) ones [36] and from EU-naCMA (existentially unforgeable under non-adaptive chosen message attacks) ones [49], where EU-RMA and EU-naCMA are weaker security notions than EU-CMA. Usually, primitives in lower levels and schemes with weaker security notions are easier to design. These constructions are greatly helpful for researchers in designing high-level primitives and schemes in practice.

On the other hand, a black-box construction between two primitives implies the existential relationship between them. For example, due to the existence of black-box construction, we believe that efficient signature schemes exist as long as we believe in the existence of efficient one-way functions. Since there are various assumptions in cryptography, it is likely that a primitive exists under some assumptions but does not exist under other assumptions. For example, if we believe that $\mathbf{P} = \mathbf{NP}$, then no one-way function or signature scheme exists.

In addition, black-box constructions have some limitations. It some cases, it is impossible to construct a primitive Q from P in a black-box manner. In 1989, Impagliazzo and Rudich [74] proved the impossibility of constructing key-agreement protocols from one-way permutations. It is called black-box separations. Later, a number of black-box separations are proven between various primitives. From black-box constructions and separations, there are five kinds of cryptographic worlds [73], each of which has a different assumption and implies the existence of different primitives.

- **Algorithmica: P** $=$ **NP**.
- **Heuristica**: **NP** problems are hard in the worst case but easy on average.
- **Pessiland**: **NP** problems are hard on average but there do not exist one-way functions.
- **Minicrypt**: One-way functions exist but no public-key cryptography exists.
- **Cryptomania**: Public-cryptography exists.

Note that although signature schemes are usually considered in public-key cryptography, they exist in **Minicrypt** rather than **Cryptomania**. Black-box constructions and separations are helpful not only for cryptographic designs, but also for understanding the cryptographic worlds.

1.3 Main Contributions

This dissertation studies the security of HBS schemes. We start with the fundamental hash functions. The security of HBS schemes is reduced to assumptions of hash functions, such as preimage resistance and collision resistance. In addition, some HBS schemes are related to variants of subset resilience, which lacks research. The study explores the quantum security of subset resilience and the relationship with other assumptions. Then, the study turns to security notions for HBS schemes. HBS schemes are classified into two types, stateful ones, and stateless ones. In a stateful signature scheme, the signer needs to maintain a dynamic state, while the signer in a stateless scheme does not. This dissertation analyzes the security of stateful signature schemes against chosen state attacks and that of stateless ones against quantum-access attacks.

1.3.1 On subset-resilient Hash Function Families

Subset-resilient hashing (SRH) [94] is a sort of hash function families, which is first proposed as a building block of HORS (Hash to Obtain Random Set), a stateless few-time HBS scheme. Talking about common security notions for hash function families, we usually focus on the behavior of a single function mapping one element from the domain to one element of the range. For example, a collision-resistant hash function family (CRH) $\mathcal{H} = \{h : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}\}$ requires that for $h \leftarrow \mathcal{H}$, it is hard to find distinct x_1, x_2 such that $h(x_1) = h(x_2)$. Instead, subset resilience focuses on a hash function H mapping to a *subset* of size at most k. We call $(x, x_1, x_2, ..., x_r)$ an (r, k)-subset cover with regard to H if it holds that $H(x) \subseteq \bigcup_{i=1}^{r} H(x_i)$ and $x \neq x_i$ for any $i = 1, ..., r$. An (r, k)-subset-resilient function family requires that for randomly sampled H, it is hard to find an (r, k)-subset cover with regard to H.

Indeed, here the hash function H can be considered as a tuple of k “partial” hash functions $(h_1, ..., h_k)$. When sampling H, it samples $h_1, ..., h_k$ from a hash function family $\mathcal{H} = \{h : \{0,1\}^{m(n)} \to [k]$, and then defines $H(x) = \{h_1(x), ..., h_k(x)\}$. (Note that the h_i’s can be sampled dependently.)

Little knowledge about SRH is known. Aumasson and Endignoux [8] propose the classical generic security for SRH. Still, the power of SRH is unclear, which leads to several interesting questions. How about the generic security of SRH in the quantum world? What is the relation between subset resilience and other assumptions (such as collision resistance)? Can we construct a provable SRH by other fundamental primitives, such as one-way permutations? (If the answer of the third question is “yes”, then SRH exists in the **Minicrypt** world.)

In this dissertation, we attempt to answer the above questions around SRH.

- **A generic quantum attack against SRH**. First, we give a quantum

$$\begin{tikzcd}
& & \text{Quantum Attack} \arrow[d, squiggly, "\text{Th.8}"] & \\
\text{OWP} \arrow[r, "/" marking] \arrow[rrd, rounded corners, to path={|- (\tikztotarget)}, "/" marking, "\text{Th.10}"'] & \text{CRH} \arrow[r] & \textbf{rSRH} \arrow[r, "\text{Th.9}"'] & \text{dCRH} \\
& & \textbf{SRH} \arrow[u] &
\end{tikzcd}$$

Figure 1.1: The relation among subset resilience and other assumptions. A→B means that the existence of A implies the existence of B. A↛B means the impossibility of constructing B from A in the fully black-box manner.

algorithm finding subset covers, giving an upper bound of the quantum security of subset resilience. The algorithm is more efficient than simply implementing quantum exhaustive search [60] on this problem.

- **SRH⇒dCRH**. Second, we prove the statement that the existence of SRH implies the existence of (infinitely often) distributional collision-resistant hashing (dCRH), which is a weaker assumption than CRH. Informally, dCRH implies the hardness to find a *uniform* collision of the hash function. Thus, the power of assuming the existence of SRH is stronger than dCRH. Note that this proof does not yield a black-box construction of SRH from dCRH.

- **OWP↛SRH**. Third, we prove the impossibility of constructing an SRH from one-way permutations (OWPs) in a fully black-box manner. Our proof uses Simon's separating oracle [100], which proves the fully black-box separation of dCRH from OWP.

To sum up, the relation among SRH and other assumptions with regard to hash functions is depicted in figure 1.1 (where rSRH will be introduced in Chapter 4).

1.3.2 Security Notions for Stateful Signature Schemes

For a standard (stateless) signature scheme, security is usually considered under chosen message attacks (CMA), where the adversary has no information about the secret key but can query the signing oracle maliciously. In hash-based cryptography, there are a number of stateful signature schemes. For a stateful scheme, the secret key and states are separated so that an adversary may attack the scheme in different ways. For example, during the process of a CMA, the adversary may additionally obtain information about the dynamic state or even change it maliciously. It is obviously stronger than a general CMA adversary. This naturally brings up an interesting question: are stateful signature schemes still secure under stronger attackers than CMA adversaries?

In this dissertation, we research security notions for stateful signature schemes. We present a formal treatment for stateful signature schemes with fine-grained security notions and analyze the relations among them. We then show instantiations (not only in hash-based cases). In standard CMA experiments, the adversary cannot directly gain information from states maintained by the signing oracle. We call this attack a *hidden state chosen message attack* (HSCMA). (In the following, we say $*$SA instead of $*$SCMA for convenience.)

In addition to HSA security, we focus on a stronger adversary, which is able to obtain the states of the scheme or even program the states. We call these a *known state attack* (KSA) and a *chosen state attack* (CSA), respectively. Moreover, we consider a weaker adversary than a CSA adversary, and we call this attack an n-*weak chosen state attack* (n-wCSA). It can only program the state to one that has been maintained by the signing oracle at most n times. In the case that $n = 1$, we simply call this opponent a w-CSA adversary.

wCSA security is essential to a stateful signature scheme. It is not only a new security model that considers strong attacks but also a bridge between stateful signature schemes and stateless ones. As far as we know, there is no black-box construction from a CMA-secure stateful signature scheme (in our definition, an HSCMA-secure one) to a CMA-secure stateless scheme, but we show that a wCSA-secure or an n-wCSA-secure stateful signature scheme could qualify. We provide some generic constructions for stateful signature schemes with different levels of security in black-box manners.

In addition, we explore how to construct a wCSA-secure or an n-wCSA-secure stateful signature scheme in black-box manners. We start from primitives of

$$\begin{tikzcd}
 & \text{HSRMA/HSCMA} \arrow[d, shift left, "\text{+PRF (Th.2)}"] & \\
 & \text{KSRMA/KSCMA} \arrow[u, Rightarrow, shift left] \arrow[d, shift left, "\text{+PRF (Th.3)}"] & \\
\text{OTS} \arrow[r, "\text{(Th.7,8)}"'] & \text{wCSRMA/wCSCMA} \arrow[u, shift left, "\text{(Th.4)}"] \arrow[d, shift left, "\text{+}n\text{-time Sig (Th.9)}"] \arrow[dr, "\text{+CRH (Th.5)}"] & \\
 & n\text{-wCSRMA/}n\text{-wCSCMA} \arrow[u, Rightarrow, shift left] \arrow[r, "\text{(Th.6)}"'] & \begin{array}{c}\text{stateless}\\ \text{RMA/CMA}\end{array} \\
 & \text{CSRMA/CSCMA} \arrow[u, Rightarrow] \arrow[ur, leftrightarrow, "\text{(Th.1)}"'] &
\end{tikzcd}$$

Figure 1.2: The transform relationships and constructions among different security notions for stateful signature schemes. $A \to B$ means that there exists a black-box construction from A-secure schemes to B-secure schemes. $A \Rightarrow B$ means A-security immediately implies B-security.

"lower levels", such as a wCSA-secure stateful signature scheme with low ability to be sampled (which can be instantiated by one-time signature schemes) and a CMA-secure stateless signature scheme with low usage times. Note that all of our constructions are provably secure in standard models.

In addition to the abovementioned scenarios under chosen message attacks, we can simultaneously define the security notions under random message attacks (RMAs). Our results are summarized in Figure 1.2.

In our constructions, some additional primitives are introduced, such as pseudorandom generators (PRGs), pseudorandom functions (PRFs), and collision-resistant hash functions (CRHs). PRGs and PRFs can be theoretically constructed by one-way functions, which are implied from any (stateless or stateful) signature scheme. On the other hand, all of the primitives can be instantiated by well-constructed symmetric hash functions. Note that many digital signature schemes use hash functions before signings, so our construction does not introduce additional assumptions as long as the hash functions in the original schemes already have these properties.

1.3.3 Quantum-access Security of Stateless Hash-based Signature Schemes

Quantum attacks on cryptographic schemes are usually classified into two types [65, 25, 64]. In the first type, the adversary can use execute quantum computations and send classical queries to the online oracles. This is called post-quantum security (or Q1 security in some literature). In the second type, the queries to the oracles are also in superpositions. This is called quantum-access security (or Q2 security). In signature schemes, the above EU-CMA security is post-quantum security. As for the quantum-access security, it considers the case that the adversary can query messages in superpositions to the signing oracle, and get the corresponding signatures with quantum states in response. The adversary may gain more information than that in classical cases. There is literature [40, 52] considering such situations in practice. For example, suppose the device running protocol is a quantum machine. An adversary may manage to observe the intermediate information with a quantum state before measurements. On the other hand, quantum-access security can be required by some upper-level primitives or protocols. For example, quantum-access-secure pseudorandom functions (qPRF) have been used in constructing message authentication codes [23] and signature schemes [24].

In 2013, Boneh and Zhandry [24] propose a security notion called existential unforgeability under *quantum* chosen message attacks (EU-qCMA) considering this situation. In the EU-qCMA experiment, the adversary can query q quantum messages in superpositions to the signing oracle. Finally, it is required to output $(q + 1)$ signatures for distinct messages to the challenger. They also show a special example that is EU-CMA (against quantum adversaries) but not EU-qCMA, which implies a separating result between the two security notions. Thus, EU-qCMA is a strictly stronger security notion for post-quantum signature schemes and deserves further research.

In addition, Alagic et al. propose another notion called blind unforgeability against quantum chosen message attacks (BU-qCMA). It is a stronger notion than EU-qCMA. The BU security of stateless HBS schemes is lacking in research as well.

In this dissertation, we focus on the generic security of stateless HBS schemes, especially in the quantum-access setting.

- First, we analyze the generic security of few-time stateless HBS schemes, such as HORS [94] and FORS [17]. Especially, we analyze the security of (single-instance) FORS against quantum-access attacks. We suppose the hash functions in the HBS schemes are ideal and modeled as (quantum-accessible) random oracles. As a result, the security of the few-time HBS schemes can be guaranteed when the number of signing operations is not large.

- Second, we show some quantum-access attacks on two many-time stateless HBS schemes, SPHINCS and SPHINCS+. The complexity of our attacks is much lower than the security in the classical setting. We thus conclude that quantum-access attacks are more threatening to security than classical chosen message attacks for these schemes. We consider the attacks in the EU-qCMA model and the BU-qCMA model. See detail in 1.3.

- Third, we propose a variant of SPHINCS+, whose quantum-access security can be proven. We call it SPHINCS-FORS. We give formal security proof against classical attacks and quantum-access attacks. As far as we know, it is the first practical stateless HBS scheme with provable security against quantum-access attacks.

  The price is that the signature size of SPHINCS-FORS is larger. It depends on the security level that we need. We show instantiations of SPHINCS-FORS with different parameters to provide different security levels and give comparisons to SPHINCS+. See details in Figure 1.4 and Section 5.5.

Scheme	CMA Security		Our Attack (EU)		Our Attack (BU)	
	$\log q_s$	$\log q_H$	$\log q_s$	$\log q_H$	$\log q_s$	$\log q_H$
SPHINCS-256 [15]	50	128	43	43	43	Small
SPHINCS+-256s [7]	64	128	48	80	43	43
SPHINCS+-256f [7]	64	128	46	80	42	42

Figure 1.3: Comparisons between our quantum-access attacks and the CMA security of SPHINCS and SPHINCS+, which implies the bounds of toleratable hash queries against different attacks. q_s and q_H denote the number of (quantum) signing queries and quantum hash queries, respectively. For example, if the adversary issues 2^{43} quantum signing queries to the signing oracle and more than 2^{43} quantum hash queries, then SPHINCS-256 will be broken in the sense of PO model. “Small” means that the attack only requires a polynomial number of queries.

Scheme	Security		Size	Provably Secure?
	$\log q_s$	$\log q_H$		
SPHINCS+-256s	48	$\leq$80	29272	✗
SPHINCS-FORS (Ours) v.2	48	$\geq$80	47072	✓
SPHINCS+-256s$*$	64	$\leq$128	41792	✗
SPHINCS-FORS (Ours) v.1	64	$\leq$128	31904	✗
SPHINCS-FORS (Ours) v.3	64	$\geq$128	101424	✓

Figure 1.4: Comparison between (deterministic) SPHINCS+ and our variants against quantum chosen message attacks. $\log q_H \leq a$ means that there exists a quantum-access attack with 2^a quantum hash queries to the best of our knowledge. It implies an upper bound of the security level (without security proof). $\log q_H \geq a$ means that any quantum-access attack requires *at least* 2^a hash queries. It implies a lower bound of the security level (with security proof). SPHINCS+-256s is an extended version of SPHINCS+-256s such that our attack of Note that all the schemes in this figure can provide 128-bit EU-CMA security when $\log q_s \leq 64$.

1.4 Organizations

This dissertation is composed as follows:

Chapter 2 introduces some basic notations, preliminaries, and useful lemmas.

Chapter 3 introduces the first result on subset-resilient hash function families.

Chapter 4 introduces the second result on security notions for stateful signature schemes.

Chapter 5 introduces the third result on quantum-access security of stateless hash-based signature schemes.

Chapter 6 gives the conclusion of this dissertation.

1.5 Related Work

In this section we introduce some work that is deeply related to this dissertation.

Hash-based Signature Schemes

HBS schemes have a long history that begins with Lamport's one-time signature scheme [81]. Then, there are some other one-time schemes such as LM-OTS [83], Biba [90], WOTS [28], and WOTS+[67]. They can be converted to a many-time stateful signature scheme by a Merkle-tree [87]. After that, there are several variants of Merkle's scheme such as CMSS [30], GMSS [31], SPR-MSS [39], XMSS [29, 26], XMSS+ [68], XMSS^{MT} [70], and XMSS-T[72]. Note that these schemes are stateful. As for stateless ones, in the few-time case, there are HORS [94], HORS++ [92], HORST [15], PORS [9], FORS [17], DFORS [1], and FORC [109]. In addition, Goldreich [56] proposes a framework of many-time schemes, but it is far from efficient. SPHINCS [15] is the first practical stateless HBS using Goldreich's framework. There are several variants of SPHINCS [71, 9, 17, 104, 109]. SPHINCS+ [17] is one of the 6 candidates of NIST's standardization in the 3rd round. As for security analysis of SPHINCS and its variants, Castelnovi et al. [32] and Kannwischer et al. [75] propose side-channel attacks on SPHINCS. Cremers et al. [38] analyze the security beyond unforgeability of SPHINCS+.

Generic Security of Hash Functions

In this dissertation, we always treat hash functions as black boxes and do not care much about their concrete structures. There are attacks regardless of the structures, called *generic attacks*. For instance, the time complexity of a generic attack on one-wayness is $O(N)$, where N denotes the size of the range. In the quantum setting, the above complexity decreases to $O(N^{1/2})$ [60].

Generic attacks are essential for the security analysis of hash functions. They provides upper bounds for the security level. On the other hand, suppose a hash function is ideal enough (modeled as a random oracle), an adversary has to make enough hash queries to break the security notion. It is called generic security. In many cases, the generic attacks are optimal, and they immediately imply the generic security (e.g., one-wayness, collision resistance [2, 107], target collision resistance [72], and multi-collision resistance [84]). However, proving the optimality of a generic attack is usually much more complicated work than searching for an attack, especially in the post-quantum setting. Yamakawa and Zhandry [108, 103] propose new techniques to prove the generic quantum security of hash functions.

Collision Resistance and its variants

Collision-resistant hashing (CRH) is first proposed in [41], and collision resistance is considered a necessary property of a secure hash function. A "birthday attack" is a generic attack finding collisions with $O(N^{1/2})$ number of hash queries where N denotes the size of the range. In a quantum setting, Brassard et al. [27] present a quantum algorithm on finding collisions of 2-to-1 functions with time complexity $O(N^{1/3})$ and quantum memory complexity $O(N^{1/3})$. This algorithm is proven to be optimal in terms of quantum time complexity [2, 107]. In addition, Chailloux et al. [33] present a quantum algorithm on finding collisions with time complexity $O(2^{2n/5})$ and quantum memory complexity $O(n)$.

Distributional collision-resistant hashing (dCRH) [44] is a hash function family with weaker secueity notion. Komargodski and Yogev [78] first analyze the power of dCRH, and prove that a statistical zero-knowledge proof implies a dCRH. Then, Bitansky et al. [18] prove that a dCRH implies a constant-round statistically-hiding commitment scheme, and a two-message statistically-hiding commitment scheme also implies a dCRH.

A multi-collision-resistant hash function (MCRH) is another variant of collision-resistant hashing. A k-MCRH requires that it is hard to find k distinct elements from the domain such that their images are all equal. Suzuki et al. [101] first analyze the time complexity of MCRH. Hosoyamada et al. [66] and Liu et al. [84] propose two quantum algorithms on finding k-collisions. Komargodski and Yogev [78] prove that the existence of MCRH implies the existence of dCRH. In recent years, there are other results on MCRH [19, 77].

Black-box Separation

The first black-box separating result is the impossibility of constructing a key-agreement protocol from one-way permutations in a (fully) black-box manner [74]. Then, Simon [100] proves the fully black-box separation of CRH from one-way permutations. Indeed, Simon's proof also rules out the possibility of constructing a dCRH from one-way permutations in fully black-box manners.

In 2005, Haitner et al. [61] prove the fully black-box separation of constant-round statistically-hiding commitment schemes from one-way permutations. Furthermore, Berman et al. [14] and Komargodski et al. [77] independently construct constant-round statistically-hiding commitment schemes from MCRH in fully black-box manners. This rules out the possibility of constructing an MCRH from one-way permutations in fully black-box manners. The latter authors also prove the fully black-box separation of CRH from MCRH.

Due to the above black-box separation results, Komargodski et al. [77] define four worlds of hashing-related primitives, which are fine-grained based on Impagliazzo's five worlds mentioned in Section 1.2. They are respectively the worlds where CRH, MCRH, and OWF exist, and the world where none of them exists.

Subset resilience and HBS schemes

Subset resilience is first proposed as one of the assumptions needed in a few-time HBS scheme called HORS [94]. The security under chosen message attack is based on one-wayness and subset resilience. Then, Pieprzyk et al. [92] propose HORS++, a variant of HORS, and remove the requirement of subset resilience. Instead, they introduce a cover-free family [48], which captures similar properties to subset resilience. In other words, a cover-free family implies an information-theoretic version of SRH, meaning that the probability of finding a subset cover is exactly 0.

Aumasson and Endignoux [8] first analyze the generic security of subset resilience in classical settings and then propose an attack on HORS called a weak message attack. To avoid weak message attacks, they propose PORS, a variant of HORS. Based on that, the authors propose a variant of SPHINCS [15] called Gravity-SPHINCS [9].

In the security analysis of SPHINCS, the security is reduced to subset resilience. However, SPHINCS essentially only requires a *target* version of subset resilience, which is a relaxation of subset resilience. On the other hand, the security of SPHINCS+ is based on interleaved target subset resilience (ITSR), a variant of target subset resilience deliberately designed for SPHINCS+.

Although they seem similar, subset resilience and ITSR are quite different notions. First, ITSR introduces an "interleaf". It means that the output of the hash function also includes an index, and all the elements of a subset cover are required to collide on a single index. Second, ITSR is a target version of subset resilience as well. The hash computations introduce a randomizer that cannot be freely computed by the adversary.

In SPHINCS+ paper, the authors propose a generic attack against ITSR. Although their attacks can be immediately extended to our non-target case, our attack is more efficient since an adversary is more powerful in non-target cases.

In addition, Yehia et al. [1] analyzes the security of FORS in SPHINCS+ and propose a variant of FORS called DFORS. The motivation of DFORS also comes from a consideration of the non-target cases.

As far as we know, all of the existing practical stateless (few-time or many-time) HBS schemes are related to subset resilience or its variants. On the other hand, all of the existing stateful ones (including one-time signature schemes) are not related to subset resilience, such as MSS [87], XMSS [29], XMSS^{MT} [70], XMSS-T [72], and LMS [83].

Stateful Signature Schemes

Many studies have been performed on stateful signature schemes. For example, Merkle [87] introduced a stateful signature scheme with a limited number of signing operations, where the state was used to count the number of issued signatures. Every time a signer issues a signature, it uses a new key pair of one-time signature schemes. After that, several multi-time stateful signature schemes are proposed based on these one-time signature schemes [30, 31, 39, 29, 70, 72]. In RSA-based constructions, the Dwork-Noar scheme [46] and the Cramer-Damgård scheme [37] are well-known stateful signature schemes, where the state is a tree with labels.

One-time signature schemes are essential building blocks for constructing stateful signature schemes. Actually, one-time signature schemes can also be considered simple stateful signature schemes with states in $\{0, 1\}$. The earliest proposal goes back to Lamport's scheme [81]. In addition, there are examples of one-time signature schemes based on other assumptions, such as lattices [85] and codes [51, 91].

To eliminate the limitation of having to maintain states, Goldreich [56] proposed a stateless signature scheme constructed by one-time signature schemes, but the size of the signature was extremely large. In 2013, Böhl et al. [21] proposed a construction for a non-adaptive CMA-secure stateless signature scheme from tag-based signature schemes, which are essentially stateful signature schemes. The construction requires a chameleon hash function to achieve adaptive CMA security, and this type of function is hard to construct by using only symmetric hash functions. In this scheme, the number of signing operations is limited. Later, Seo [97, 98] gives a tighter security proof of this scheme and eliminate the limitation of the number of signing operations.

Quantum-access Security

Quantum-access security is first considered for pseudorandom functions [106] and then generalized to message authentication codes [23], signature schemes and encryptions [24]. After that, other quantum-access security notions are proposed [53, 3] for message authentication codes (and they can also be extended to fit signature schemes). In particular, the blind unforgeability of Lamport's scheme and WOTS has been evaluated[86].

In terms of the quantum-access security of HBS schemes, Boneh and Zhandry [24] prove the EU-qCMA security of Lamport's scheme [81] and MSS [87]. The former is a one-time HBS scheme, and the latter is a stateful HBS scheme. Hopefully, stateful HBS schemes (such as XMSS [29]) are also EU-qCMA since they are essentially variants of MSS, and the structures are similar. However, when it comes to stateless schemes, the cases become different. (The authors proved the EU-qCMA security of stateless MSS, but stateless MSS is far from efficient in practice.) For practical stateless HBS schemes, such as SPHINCS and SPHINCS+, the security against quantum-access attacks still lacks research.

Note that Boneh and Zhandry [24] proposed a generic construction of EU-qCMA schemes from UU-RMA (universally unforgeable under random message attacks) schemes by introducing a hash function modeled as a quantum random oracle. However, the construction does not work on HBS. It is because the UU-RMA scheme in this construction is required to be polynomially many-time, meaning that the success probability of the adversary remains low even if it obtains a large (but polynomial) number of random message-signature pairs. As for HBS, the number of signing executions is always restricted. For example, SPHINCS+ provides 128-bit security only when the number of signing queries is under 2^{64} (if it exceeds 2^{64}, the security will be degraded rapidly). Suppose we want an EU-qCMA scheme such that the probability of breaking the security is under 2^{-10} when the adversary sends only 2^{10} quantum queries to the signing oracle and 2^{30} quantum hash queries (this requirement is fairly weak in practice). If we use the above generic construction, we need a 2^{117}-time UU-RMA scheme with 254-bit quantum security, which far exceeds the security of SPHINCS+.

In recent years, there are a large amount of research on quantum-access security of various cryptographic primitives, including pseudorandom functions [106], message authentication codes [23], block ciphers [79, 80, 76, 82, 64], encryption schemes [24], signature schemes [24, 52, 3, 34] and so on.

Chapter 2

Preliminaries

2.1 Notations

Let X be a set or a distribution, $x \leftarrow X$ means that x is uniformly sampled from X. Let $\mathcal{M}$ be a probabilistic algorithm, $x \leftarrow \mathcal{M}(\cdot; r)$ means that x is the output of $\mathcal{M}$ with randomness r. $x \leftarrow \mathcal{M}$ means x is the output of $\mathcal{M}(\cdot; r)$ where r is random chosen. For event E relative to $\mathcal{M}$, we use $\Pr_{\mathcal{M}}[E]$ to represent the probability of E over the randomness of $\mathcal{M}$.

For integer $n \in \mathbb{N}$, we denote $[n] = \{1, ..., n\}$. We say that $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is a negligible function if for every constant $c > 0$, there exists $N_c > 0$ such that $\epsilon(n) < n^{-c}$ holds for all $n > N_c$.

The statistical distance of two variables X and Y over distribution Ω, denoted by $\Delta(X, Y)$, is defined as $\Delta(X, Y) = \frac{1}{2}\sum_{x \in \Omega} |\Pr[X = x] - \Pr[Y = x]|$. If $\Delta(X, Y) \leq \delta$, we say that X and Y are δ-close.

We denote $\{0, 1\}^*$ as a string of arbitrary length, and we suppose that for any element x there is an efficient computable encoding function C_x that can encode x to $\{0, 1\}^\star$. For a function $F : \{0, 1\}^\star \to Rng$ and any element x, we say $F(x)$ instead of $F(C_x(x))$ for convenience. We denote $\perp \notin \{0, 1\}^\star$ as an error symbol, which can be an input or output of an algorithm.

A quantum system A is associated to a Hilbert space $\mathcal{H}_A$ with the inner product $\langle \cdot | \cdot \rangle$. The state of the system is described by a vector $|\phi\rangle \in \mathcal{H}_A$ such that the

Euclidean norm $|| \, |\phi\rangle \, || = \sqrt{\langle\phi|\phi\rangle} = 1$. Let A and B be two quantum systems. We define the joint quantum system through the tensor product $\mathcal{H}_A \otimes \mathcal{H}_B$. The joint state of $|\phi_A\rangle \in \mathcal{H}_A$ and $|\phi_B\rangle \in \mathcal{H}_B$ is denoted by $|\phi_A\rangle \otimes |\phi_A\rangle$, or simply $|\phi_A\rangle \, |\phi_B\rangle$. A pure n-qubit quantum state $|\phi\rangle$ can be expressed as $|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x \, |x\rangle$, where α_x is the complex amplitude of the basis $|x\rangle$ and it holds that $\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1$. We refer to [89] for more formal and detailed discussions on quantum computations.

2.2 Hash Function Families

Definition 1 *(Efficient function family ensemble.) A function family ensemble $\mathcal{F} = \{\mathcal{F}_n : \mathcal{D}_n \to \mathcal{R}_n\}_{n\in\mathbb{N}}$ is efficient if:*

- *$\mathcal{F}$ is samplable: there exists a probabilistic polynomial-time algorithm such that given 1^n, it outputs the description of a uniform element of $\mathcal{F}_n$.*
- *$\mathcal{F}$ can be efficiently computed: there exists a deterministic polynomial-time algorithm such that given $x \in \mathcal{D}_n$ and $f \in \mathcal{F}_n$, it outputs $f(x)$.*

Especially, we say that a probablistic polynomial-time algorithm Samp_k is a k-sampling algorithm of an efficient function family ensemble $\mathcal{F}$ if it outputs a tuple of (possibly non-uniform) functions $(f_1, ..., f_k) \in \mathcal{F}_n^k$.

Obviously, one can construct a k-sampling algorithm of $\mathcal{F}$ by simply sampling k elements from $\mathcal{F}_n$. However, there may exist other k-sampling algorithms such that the resulting function tuple has some special properties that we may take interest in. We give a simple example. Let $\mathcal{F} = \{\mathcal{F}_n : \{0,1\}^n \to \{0,1\}^{n+1}\}$. $\mathsf{Samp}(1^n; r)$ is a sampling algorithm that outputs $f(x) = x||0 \oplus r$ where $r \leftarrow \{0,1\}^{n+1}$ is the randomness. We want a 2-sampling algorithm Samp_2 such that for $(f_1, f_2) \leftarrow \mathsf{Samp}_2(1^n)$, it is hard to find a pair (x_1, x_2) such that $f_1(x_1) = f_2(x_2)$. If we sample (f_1, f_2) by running Samp twice, it will have this property only with probability 1/2. But if we sample the first function by randomness r and sample the section one by randomness $r \oplus 1$, the resulting (f_1, f_2) will have this property with probability 1.

If an efficient function family ensemble is compressing, which means the size of the domain is larger than that of the range, we call it a hash function family. In practice, an ideal hash function family is expected to be one-way and collision-resistant.

Definition 2 *(One-wayness.) Let $\mathcal{F} = \{\mathcal{F}_n : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}\}$ be an efficient samplable family ensemble. We say that $\mathcal{F}$ is a one-way function family (OWF) if for any probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function $\epsilon(\cdot)$ such that*

$$\Pr_{f \leftarrow \mathcal{F}_n, x \leftarrow \{0,1\}^{m(n)}} \left[f(x') = f(x) \middle| x' \leftarrow \mathcal{A}(1^n, f, f(x)) \right] \le \epsilon(n) \tag{2.1}$$

for large enough $n \in \mathbb{N}$.

In particular, if $\mathcal{F}$ is a family ensemble of permutations, we say that $\mathcal{F}$ is a one-way permutation family (OWP). If $\mathcal{F}$ is a hash function family, we say that $\mathcal{F}$ is preimage-resistant hash function family.

Let $\mathcal{H}_n$ be a hash function family and $h \leftarrow \mathcal{H}_n$. We say that (x_1, x_2) is a collision w.r.t. h if $h(x_1) = h(x_2)$ and x_1, x_2 are distinct. If there is no polynomial-time adversary that can find a collision for $h \leftarrow \mathcal{H}_n$, we say that $\mathcal{H}$ is a collision-resistant hash function family (CRH).

Definition 3 *(Collision-Resistant Hashing.[41]) Let $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}\}$ be a hash function family. We say that $\mathcal{H}$ is collision-resistant if for any probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function $\epsilon(\cdot)$ such that*

$$\Pr_{h \leftarrow \mathcal{H}_n} \left[\begin{matrix} x_1 \neq x_2 \\ h(x_1) = h(x_2) \end{matrix} \middle| (x_1, x_2) \leftarrow \mathcal{A}(1^n, h) \right] \le \epsilon(n) \tag{2.2}$$

holds for large enough $n \in \mathbb{N}$.

Distributional collision resistance is a relaxation of classical collision resistance. A distributional collision-resistant hash function family (dCRH) guarantees that there is no probabilistic polynomial-time adversary that can output a *uniform* collision. We first define the distribution of uniform collisions.

Definition 4 *For $h : \{0,1\}^m \to \{0,1\}^n$, a joint random variable $\mathit{COL}_h \subseteq \{0,1\}^m \times \{0,1\}^m$ over pairs of inputs (x_1, x_2) is sampled as follows: (1) x_1 is uniformly sampled from $\{0,1\}^m$, (2) x_2 is uniformly sampled from the set $\{x \in \{0,1\}^m : h(x) = h(x_1)\}$.*

Note that $(x_1, x_2) \leftarrow \mathsf{COL}_h$ does not imply that (x_1, x_2) is a collision of h, since it is possible that $x_1 = x_2$. But in the case that $m > n$ (e.g. $m \geq 2n$), $x_1 = x_2$ only holds with negligible probability.

Definition 5 *(Distributional Collision-Resistant hashing.[44]) Let $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}\}$ be a hash function family. We say that $\mathcal{H}$ is distributional collision-resistant if for any probabilistic polynomial-time algorithm $\mathcal{A}$ and any two negligible functions $\delta(\cdot)$ and $\epsilon(\cdot)$, it holds that*

$$\Pr_{h \leftarrow \mathcal{H}_n}[\Delta(\mathcal{A}(1^n, h), \mathit{COL}_h) \leq \delta(n)] \leq 1 - \epsilon(n) \tag{2.3}$$

for large enough $n \in \mathbb{N}$.

We say that a dCRH is infinitely often secure if the above security holds for infinitely many n's rather than large enough n's.

2.3 The Random Oracle Model

In many instances of constructions based on hash functions, the above security notions are not enough for security proof. Indeed, if a hash function is ideal enough, it is usually modeled as a *random oracle* [11], which treats the hash function as a uniformly random function. Let H be a hash function modeled as a random oracle H and a building block of a scheme Π. In the security experiment of Π, the success probability is taken over the randomness of Π, the randomness of the adversary $\mathcal{A}$, and the random choice of H. The random oracle model has the following useful properties:

- **Uniformity**. If x is not queried to the random oracle H, then the distribution of $\mathsf{H}(x)$ is uniform.

- **Extractability**. Suppose $\mathcal{A}$ is a subroutine of a reduction $\mathcal{R}$. When $\mathcal{A}$ queries x to H, x can be seen by $\mathcal{R}$.
- **Programmability**. $\mathcal{R}$ can set $\mathsf{H}(x)$ to a value of its choice, as long as it is uniformly distributed.

In post-quantum cryptography, the random oracle is usually required to be quantum-accessible, since an adversary in practice can compute the hash function by a quantum computer. We call it *quantum random oracle* [22]. That is, let H be a quantum random oracle. It samples a random function H at the beginning. One can query $\sum_{x,y} \psi_{x,y} |x, y\rangle$ (with superpositions) to the quantum random oracle, and obtains $\sum_{x,y} \psi_{x,y} |x, y \oplus H(x)\rangle$.

The reductions $\mathcal{R}$ can gain less information from quantum random oracles than the classical ones. On the one hand, the reduction cannot directly gain information from the observation of a query $\sum_{x,y} \psi_{x,y} |x, y\rangle$ without measurements. A measurement may be detected by the adversary. On the other hand, it is possible to detect the programming since the adversary may query a uniform quantum state to the quantum oracle.

Fortunately, the adaptive reprogramming lemma [59] shows that a quantum random oracle can still be programmed to some extent. (It is improved from one-way to hiding lemma [102, 5].) It shows that if we reprogram the random oracle in some partially random records, then an adversary is hard to tell the difference. For an oracle $\mathsf{H} : X \to Y$, $x \in X$ and $y \in Y$, denote $\mathsf{H}^{x \to y}$ as an oracle that is the same as H except that it maps x to y. The adaptive reprogramming lemma is as follows.

Lemma 1 *(Adaptive reprogramming lemma. [59]) Let X_1, X_2 and Y be finite sets, $\mathsf{H}_0 : X_1 \times X_2 \to Y$ be a random oracle and Repro_b be the adaptive reprogramming game depicted in Figure 2.1. Let $\mathcal{A}$ be an algorithm issuing q quantum queries to H_b and R classical queries to $\mathsf{Reprogram}$. Then, the probability of distinguishing b is at most*

$$|\Pr[\mathsf{Repro}_1^{\mathcal{A}} = 1]| - |\Pr[\mathsf{Repro}_0^{\mathcal{A}} = 1]| \leq \frac{3R}{2}\sqrt{\frac{q}{|X_1|}}. \tag{2.4}$$

```
**Game** $\mathsf{Repro}_b$
$\mathsf{H}_1 := \mathsf{H}_0$
$b' \leftarrow \mathcal{A}^{|\mathsf{H}_b\rangle, \mathsf{Reprogram}}$
**return** $b'$
```

```
$\mathsf{Reprogram}(x_2)$
$(x_1, y) \leftarrow X_1 \times Y$
$\mathsf{H}_1 := \mathsf{H}_1^{x_1 || x_2 \to y}$
**return** $x_1$
```

Figure 2.1: Adaptive reprogramming games for $b \in \{0, 1\}$.

In addition, it is proven that a quantum random oracle is both one-way and collision-resistant. Indeed, the generic security of any relation with regard to a quantum random oracle can be evaluated by the quantum query complexity lemma [103] as follows.

Lemma 2 *(Quantum query complexity lemma. [103]) Let H be a random function mapping $\mathcal{X}$ to $\mathcal{Y}$, r be a positive integer, and R be a relation over $\mathcal{Y}^r$. For any quantum adversary $\mathcal{A}$ which can query H at most q times, it holds that*

$$\Pr_H\left[x_1, ..., x_r \text{ are distinct} \wedge (H(x_1), ..., H(x_r)) \in R | (x_1, ..., x_r) \leftarrow \mathcal{A}^{|H\rangle}\right]$$
$$\leq (2q+1)^{2r} \Pr\left[\exists \pi \in \mathit{Perm}([r]) \text{ s.t. } (y_{\pi(1)}, ..., y_{\pi(r)}) \in R | (y_1, ..., y_r) \leftarrow \mathcal{Y}^r\right],$$

where $\mathit{Perm}([r])$ denotes the set of permutations of $[r]$.

Corollary 1 *Let $\mathcal{F}$ be an efficient function esemble modeled by a quantum random oracle. For any quantum adversary $\mathcal{A}$, it holds that*

$$Adv^{OW}_{\mathcal{F},q}(\mathcal{A}) \leq (2q+1)^2 \cdot 2^{-n}. \tag{2.5}$$

It implies that to break the preimage resistance of a hash function modeled as a quantum random oracle with constant probablity, any quantum adversary $\mathcal{A}$ needs at least $O(2^{n/2})$ quantum hash queries (even if the adversary has infinite computing power).

2.4 Primitives

Definition 6 *(Pseudorandom generator [57].) Let $l(n) : \mathbb{N} \to \mathbb{N}$ be a polynomial such that for any $n > 0$, $l(n) > n$. Let $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ be a polynomial-time function. We say G is a pseudorandom generator if for any probabilistic polynomial-time distinguisher D, there exists a negligible function ϵ such that*

$$Adv_{G,D}^{\textit{Ind-PRG}}(n) = |\,\mathbf{Pr}[D(G(s))] - \mathbf{Pr}[D(r)]| \leq \epsilon(n), \tag{2.6}$$

where s is uniformly chosen in $\{0,1\}^n$ and r is uniformly chosen in $\{0,1\}^{l(n)}$. The probability is taken over the choice of s, r and the randomness of D.

Definition 7 *(Pseudorandom function [57].) Let $F : Dom \times K \to \{0,1\}^n$ be a polynomial-time keyed function with a key set K that can be efficiently sampled. We say F is a pseudorandom function if for any probabilistic polynomial-time distinguisher D, there exists a negligible function ϵ such that*

$$Adv_{F,D}^{\textit{Ind-PRF}}(n) = |\,\mathbf{Pr}[D^{F_k(\cdot)}(1^n)] - \mathbf{Pr}[D^{f(\cdot)}(1^n)]| \leq \epsilon(n), \tag{2.7}$$

where k is uniformly chosen in K and $f : Dom \to \{0,1\}^n$ is a random function. The probability is taken over the choices of k and f and the randomness of D.

Definition 8 *(Static accumulator [10].) Let $N(\lambda)$ be a polynomial. A static accumulator A consists of four polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Wit}, \mathsf{Ver})$:*

- *The probabilistic key generation algorithm $\mathsf{Gen}(1^\lambda, N)$ takes as input a security parameter 1^λ and an accumulation threshold N. It outputs an accumulator key ak. It also determines an accumulation domain L_λ.*

- *The probabilistic evaluation algorithm $\mathsf{Eval}_{ak}(x_1, ..., x_N)$ takes as input an accumulator key ak and N elements in L_λ. It outputs an accumulated value z.*

- *The probabilistic witness extraction algorithm $\mathsf{Wit}_{ak}(x, z)$ takes as input an accumulator key ak, an element $x \in L_\lambda$ and an accumulated value z. It outputs a witness w.*

- *The deterministic verification algorithm* $\mathsf{Ver}_{ak}(x, w, z)$ *takes as input an accumulator key* ak*, an element* $x \in L_\lambda$*, a witness* w *and an accumulated value* z*. It outputs a bit* $b \in \{0, 1\}$*.*

The static accumulator satisfies two properties:

- *(Completeness.) For all* $\lambda, N > 0$*, if* $ak \leftarrow \mathsf{Gen}(1^\lambda, N)$*,* $x_i \in L_\lambda$*,* $z \leftarrow \mathsf{Eval}_{ak}(x_1, ..., x_N)$*, for every* $i \in \{1, 2, ..., N\}$*, and* $w_i \leftarrow \mathsf{Wit}_{ak}(x_i, z)$*, then* $\mathsf{Ver}_{ak}(x_i, w_i, z) = 1$ *holds.*

- *(Soundness.) For any* $N > 0$*,* $x_i \in L_\lambda$*, and any probabilistic polynomial-time Turing machine* M*, there exists a negligible function* ϵ *such that*

$$Adv_{\mathsf{A},M}^{Accu}(\lambda) = \Pr\left[\begin{matrix} ak \leftarrow \mathsf{Gen}(1^\lambda, N) \\ (x_1, ..., x_N, x^\star, w^\star) \leftarrow M(ak) \\ z \leftarrow \mathsf{Eval}_{ak}(x_1, ..., x_N) \end{matrix} : \begin{matrix} \mathsf{Ver}_{ak}(x^\star, w^\star, z) = 1 \\ x^\star \neq x_i, i = 1, ..., N \end{matrix}\right] \leq \epsilon(\lambda), \tag{2.8}$$

where the probability is taken over the randomness of Gen*,* Eval *and* M*.*

The Merkle-tree structure [87] can be used to construct a static accumulator. The soundness of Merkle's accumulator is based on the collision resistance of hash functions.

2.5 Signature Schemes

Definition 9 *[58] A signature scheme* $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *consists of three polynomial-time algorithms along with an associated message space* $\mathcal{M} = \{M_n\}$ *such that:*

- *The key generation algorithm* KeyGen *takes as input the security parameter* 1^n*. It outputs a pair of keys* (pk, sk)*, where* pk *and* sk *are called the public key and the secret key respectively.*

- *For security parameter* n*, the signing algorithm* Sign *takes as input a secret key* sk *and a message* $m \in \mathcal{M}$*. It outputs a signature* σ*.*

- *For security parameter n, the verification algorithm Ver takes as input a public key pk, a message $m \in \mathcal{M}$ and a signature σ. It outputs a bit b. If $b = 1$, we say σ is a valid signature of m.*

(Correctness.) For any $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$, $m \in \mathcal{M}_n$ and $\sigma \leftarrow \mathsf{Sign}(sk, m)$, it holds that $\mathsf{Ver}(pk, m, \sigma) = 1$.

Let $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme. SigO denotes the signing oracle that computes $\mathsf{Sign}(sk, m)$ where sk is the secret key. If SigO is quantum-accessible, it means that

$$\mathsf{SigO} : \sum_{m,t} \psi_{m,t} |m, t\rangle \rightarrow \sum_{m,t} \psi_{m,t} |m, t \oplus \mathsf{Sign}(sk, m)\rangle . \tag{2.9}$$

Especially, if Sign is probabilistic, SigO replies $\sum_{m,t} \psi_{m,t} |m, t \oplus \mathsf{Sign}(sk, m; r)\rangle$ for a random r for each query $\sum_{m,t} \psi_{m,t} |m, t\rangle$.

When focus on HBS schemes, the security of an HBS scheme is related to the number of hash queries executed by the adversary. In detail, the security is defined as follows.

Experiment $\mathsf{Exp}^{\mathsf{EU\text{-}CMA}}_{\Gamma,q_s,q_H}(1^n, \mathcal{A})$

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$

$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SigO}}(pk)$

If m^* has not been queried to SigO and $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$, **return** 1, otherwise **return** 0.

Definition 10 *Let Γ be a signature scheme. We say it is existentially unforgeable under chosen message attack (EU-CMA) if for all probabilistic polynomial-time adversary $\mathcal{A}$ given classical access to SigO, it holds that $\Pr[\mathit{Exp}^{\mathit{EU\text{-}CMA}}_{\Gamma,q_s,q_H}(1^n, \mathcal{A})] \leq$ **negl**(n), where **negl** is a negligible function, q_s denotes the number of queries to SigO, and q_H denotes the number of queries by $\mathcal{A}$. Especially, for HBS schemes, q_H denotes the number of hash queries calculated by $\mathcal{A}$.*

Experiment $\mathsf{Exp}^{\mathsf{EU\text{-}qCMA}}_{\Gamma,q_s,q_H}(1^n, \mathcal{A})$

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$

$\{(m_j, \sigma_j)\}_{j\in[r+1]} \leftarrow \mathcal{A}^{\mathsf{SigO}}(pk)$

If m_j's are distinct and for $\forall j \in [q_s + 1]$, $\mathsf{Ver}(pk, m_j, \sigma_j) = 1$, **return** 1, otherwise **return** 0.

Definition 11 *[24] Let Γ be a signature scheme. We say it is existentially unforgeable under quantum chosen message attack (EU-qCMA) if for all probabilistic polynomial-time adversary $\mathcal{A}$ given quantum access to SigO, it holds that $\Pr[\mathit{Exp}^{\mathit{EU\text{-}CMA}}_{\Gamma,q_s,q_H}(1^n, \mathcal{A})] \leq \boldsymbol{negl}(n)$, where **negl**, q_s and q_H are denoted as above.*

In addition, we introduce a security notion called blind unforgeability [3]. Let $\varepsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be an efficiently computable function. $B_{\varepsilon,n}$ be a subset of the message space $\mathcal{M}_n$ that is selected by placing each $m \in M_n$ independently with probability $\varepsilon(n)$. Define the blind signing oracle $B_{\varepsilon,n}\mathsf{SigO}$ as follows:

$$B_{\varepsilon,n}\mathsf{SigO}(m) = \begin{cases} \mathsf{Sign}(sk, m) & (m \notin B_{\varepsilon,n}), \\ \perp, & (otherwise). \end{cases} \tag{2.10}$$

Then, the experiment of blind unforgeability under quantum chosen message attacks is as follows:

```
**Experiment** $\mathsf{Exp}^{\mathsf{BU\text{-}qCMA}}_{\Gamma,q_s,q_H}(1^n, \mathcal{A})$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{B_{\varepsilon,n}\mathsf{SigO}}(pk)$
  If $m \in B_{\varepsilon,n} \wedge \mathsf{Ver}(pk, m^*, \sigma^*) = 1$, **return** 1, otherwise **return** 0.
```

Definition 12 *[3] Let Γ be a signature scheme. We say it is blind unforgeable under quantum chosen message attacks (BU-qCMA) if for all probabilistic polynomial-time adversary $\mathcal{A}$ given quantum access to $B_{\varepsilon,n}$SigO, it holds that $\Pr[\mathit{Exp}^{\mathit{BU}}_{\Gamma,q_s,q_H}(1^n, \mathcal{A})] \leq \boldsymbol{negl}(n)$, where **negl**, q_s and q_H are denoted as above.*

When we refer to blind unforgeability, we always consider the case that the signing oracle is quantum-accessible. Thus, we simply call it blind unforgeability (BU) in the following.

Remark 1 *In this dissertation, we mainly discuss EU-qCMA security instead of BU except in Section 5.4.5 when considering quantum-access security. Apart from this subsection, we use the model in Definition 11 to evaluate the security against quantum-access attacks by default.*

We omit the security parameter 1^n in the inputs of the algorithms and experiments for national simplicity.

2.6 Toolbox

2.6.1 Grover's Algorithm and Its Applications

Let $F : X \to \{0, 1\}$ be a function or a database mapping an element of the set X to a bit. It is called a *database search problem* with regard to F that finding an $x \in X$ such that $F(x) = 1$. We suppose an adversary can only evaluate the image of x by querying F as an oracle and suppose $|F^{-1}(1)|$ is non-empty ($|F^{-1}(1)| \ll |X|$). The adversary can only solve this problem after $O(|X|)$ (classical) queries to F in the worst case.

Now we consider the case that the adversary is given quantum access to $F : X \to Y$. It means the adversary submits $\sum_{x\in X,y\in Y} \alpha_{x,y} |x, y\rangle$ to F and receive in return $\sum_{x\in X,y\in Y} \alpha_{x,y} |x, y \oplus F(x)\rangle$. This is called the quantum query model. In this model, the time complexity of a quantum algorithm is measured by the number of quantum queries to F.

In the quantum query model, the adversary can solve a database search problem with a smaller number of queries [60].

Lemma 3 *([60]) Let $F : X \to \{0, 1\}$ be a function mapping an element of set X to a bit and $F^{-1}(1)$ is non-empty. Let $t = |F^{-1}(1)| > 0$ be the number of x such that $F(x) = 1$. There is a quantum algorithm that finds x^* such that $F(x^*) = 1$ with at most $O(\sqrt{\frac{|X|}{t}})$ quantum queries to F.*

In the following, we regard the above algorithm as a black box that can solve the database search problem with regard to any function/database F with at most $O(\sqrt{\frac{|X|}{|F^{-1}(1)|}})$ quantum queries to F. We call it Grover's algorithm.

An important application of Grover's algorithm is finding collisions for hash functions [27, 107]. If a function $H : X \to Y$ satisfies $|X| = k|Y|$ and $|H^{-1}(H(x))| = k$ for each $x \in X$, we say that H is a k-to-1 function. Given a 2-to-1 function $H : X \to Y$ where $|X| = 2|Y| = 2N$, a quantum algorithm can output a collision of H with $O(N^{1/3})$ queries by following steps:

1. Let $t = N^{1/3}$. Pick a list $L_1 = \{x_1, ..., x_t\}$, where x_i is chosen uniformly from the set X.

2. Evaluate $y_i = H(x_i)$ for $i = 1, ..., t$. List them in $L_2 = \{y_1, ..., y_t\}$. If there exist i_1 and i_2 such that $x_{i_1} \neq x_{i_2}$ and $y_{i_1} = y_{i_2}$, output (x_{i_1}, x_{i_2}). This step requires $N^{1/3}$ queries to H.

3. Let $F : X \to \{0, 1\}$ be a function defined as follows: $F(x) = 1$ if and only if $H(x) \in L_2$ and $x \notin L_1$. Run Grover's algorithm on F. Note that $|F^{-1}(1)| = t = N^{1/3}$, and evaluating F requires a query to H. The Grover's algorithm outputs x' after at most $O(\sqrt{N/t}) = O(N^{1/3})$ queries to H.

4. Find $y_i \in L_2$ such that $F(x') = y$. Output (x_i, x') as a collision of H.

The above algorithm submits $O(N^{1/3})$ quantum queries to H in total and outputs a collision of H. There are many studies on quantum collision-finding algorithms [4, 13, 107, 33].

In addition, Hosoyamada et al. [66] and Liu et al. [84] generalized this idea and constructed quantum algorithms finding k-multi-collisions, that is, k distinct elements that collide w.r.t. a hash function. On finding k-multi-collision, the time complexity is respectively $O(N^{\frac{1}{2}(1-\frac{1}{3^k})})$ and $O(N^{\frac{1}{2}(1-\frac{1}{2^k-1})})$ in these studies. For example, on finding 3-collisions, the time complexity is respectively $O(N^{4n/9})$ and $O(N^{3n/7})$.

2.6.2 Useful Lemmas

This subsection shows some useful lemmas that will be helpful to security proof.

The next lemma show that if we perform a partial measurement in the process of a quantum algorithm and the measurement obtains one of t outcomes, then the final output of the algorithm will remain unchanged with probability at least $1/t$.

Lemma 4 *[24] Let A be a probabilistic quantum algorithm. Let A' be another algorithm described as follows: A' runs as A but pauses it at an arbitrary stage of execution, performs a partial measurement that obtains one of t outcomes, and then resumes A. For any x, it holds that*

$$\Pr_{A'}[x \leftarrow A'] \geq \Pr_{A}[x \leftarrow A]/t. \tag{2.11}$$

Definition 13 *[72] Let $\mathcal{F} \triangleq \{f : \{0,1\}^m \to \{0,1\}\}$ be the collection of all boolean functions with input space $\{0,1\}^m$. Let $\lambda \in [0,1]$ be a constant. Define a family of distributions D_λ on $\mathcal{F}$ such that for $f \leftarrow D_\lambda$, $\forall x \in \{0,1\}^m$, $f(x) = 1$ with probability λ and $f(x) = 0$ with probability $1 - \lambda$.*

Lemma 5 *[72] Let $\mathcal{A}$ be an algorithm $\mathcal{A}$ issuing q quantum queries to $f(\cdot)$. Define*

$$Adv_{\mathcal{F},q}^{\textsf{Avg-Search}_\lambda}(\mathcal{A}) \triangleq \Pr_{f \leftarrow D_\lambda}[f(x) = 1 | x \leftarrow \mathcal{A}^f]. \tag{2.12}$$

Then, for any adversary $\mathcal{A}$, it holds that

$$Adv_{\mathcal{F},q}^{\textsf{Avg-Search}_\lambda}(\mathcal{A}) \leq 8\lambda(q+1)^2. \tag{2.13}$$

Chapter 3

On Subset-resilient Hash Function Families

3.1 Subset-resilient Hash Funtion Families

This section introduces definition of subset-resilient hash function families and some variants. The core of research on subset resilience is finding subset covers. Before givng formal definitions, this section begins with some obvervations on this problem in prior. This introduces motivations for the new variants. Then, this section introduces the formal definitions and applications of subset-resilient hash function families.

3.1.1 Observations on Subset Cover Finding Problems

First, we reduce the subset cover finding problem to another one that is easier to analyze. Consider the following two problems:

Problem 1 *Given* $(h_1, ..., h_k)$ *and* $h_i : X \to Y$ *for each* i*, find* $(x, x_1, ...x_r)$ *such that* $H(x) \subseteq \bigcup_{i=1}^{r} H(x_i)$ *where* $H(x)$ *is denoted by* $H(x) = \{h_1(x), ..., h_k(x)\}$.

Problem 2 *Given* $(h_1, ..., h_k)$ *and* $h_i : X \to Y$ *for each* i*, find* $(x, x_1, ...x_r)$ *such that* $h_i(x) \in \{h_i(x_1), ..., h_i(x_r)\}$ *for each* i*.*

Problem 1 is the original subset cover finding problem, and Problem 2 is a harder variation. A solution to Problem 2 is immediately a solution to Problem 1. However, a solution to Problem 1 is possibly not a solution to Problem 2. We call a solution to Problem 2 an (r,k)-*restricted subset cover* with regard to $(h_1,...,h_k)$. If it is hard to find an (r,k)-restricted subset cover with regard to a randomly sampled function tuple from hash function family $\mathcal{H}$, we say that $\mathcal{H}$ is an (r,k)-*restricted subset-resilient hash function family* and (r,k)-rSRH in short. (r,k)-restricted subset resilience is a weaker notion than (r,k)-subset resilience.

Furthermore, we have some observations on these problems:

Observation 1 *For any $k < k'$, finding an (r,k)-subset cover of $(h_1,...,h_k)$ is not harder than finding an (r,k')-subset cover of $(h_1,...,h_{k'})$.*

Observation 2 *For any $r < r'$, finding an (r',k)-subset cover is not harder than finding an (r,k)-subset cover.*

We can simply add $(r'-r)$ elements into an (r,k)-subset cover and obtain an (r',k)-subset cover. Observation 1 and 2 also work on Problem 2.

Observation 3 *For any $r > k$, finding an (r,k)-restricted subset cover is as hard as finding a (k,k)-restricted subset cover.*

Let $(x,x_1,...,x_r)$ be an (r,k)-restricted subset cover of $(h_1,...,h_k)$. It implies that for each $i \in [k]$, there exists $x_{a_i} \in \{x_1,...,x_r\}$ such that $h_i(x) = h_i(x_{a_i})$. Thus, $(x,x_{a_1},...,x_{a_k})$ is a (k,k)-restricted subset cover. Also, we can obtain an (r,k)-restricted cover from any (k,k)-subset cover due to Observation 2. (Note that Observation 3 is valid for Problem 2 but not for Problem 1.)

Observation 4 *Suppose $r|k$ and $k = \omega r$. For $i \in [r]$, let $h'_i(x) \triangleq h_{(i-1)\omega+1}(x)||...||h_{i\omega}(x)$ where $||$ denotes the concatenation operation. If $(x,x_1,...x_r)$ is an (r,r)-restricted subset cover with regard to $(h'_1,...,h'_r)$, it is also an (r,k)-restricted subset cover with regard to $(h_1,...,h_k)$.*

Therefore, finding a (k,k)-restricted subset cover is the core of these problems. It can be extended to general (r,k) case by these observations. Recall that finding a (k,k)-restricted subset cover implies finding $(x, x_{a_1}, ..., x_{a_i})$ such that $h_i(x) = h_i(x_i)$ for each i. It is equivalent to the following problem:

Problem 3 *Given $(h_1, ..., h_k)$ and $h_i : X \to Y$ for each i, find $(x, x_1, ...x_r)$ such that $h_i(x) = h_i(x_i)$ for each i.*

A solution to Problem 3 is immediately a (k,k)-restricted subset cover of $(h_1, ..., h_k)$. We simply call it a k-restricted subset cover. Also, we use k-rSRH to represent (k,k)-rSRH for simplicity.

Problem 3 can be considered a variant of collision-finding problems. The difference is that this problem focuses on k functions and requires $k+1$ elements as a solution. For each i, (x, x_i) is a collision of h_i. Thus, k-rSRH is a weaker assumption than CRH.

Our study will begin with Problem 3 and k-rSRH, and then the result can be simply extended to general (r,k)-rSRH and (r,k)-SRH.

3.1.2 Definitions

In this subsection, we give formal definitions of subset-resilient hash function families and variants that has been discussed above.

Given a parameter n and a set $T = \{0,1\}^{l(n)}$, suppose there is a function H mapping to a subset of T of size at most k. An (r,k)-subset cover of H is defined as $(x, x_1, ..., x_r)$ such that $H(x)$ is covered by the union of $H(x_i)$. From now on, we let r and k be constant integers. Since we have introduced the definition of k-sampling algorithms, an (r,k)-subset cover can also be defined as follows.

Definition 14 *Let $H = (h_1, ..., h_k)$ be a tuple of functions where $h_i : X \to Y$ for each $i \in [k]$. Let $\mathsf{ORS}_k(x) = \{h_1(x), ..., h_k(x)\}$. We say $(x, x_1, ..., x_r) \in X^{r+1}$ is an (r,k)-subset cover of H if $\mathsf{ORS}_k(x) \subseteq \bigcup_{i\in[r]} \mathsf{ORS}_k(x_i)$.*

Definition 15 *(Subset-Resilient Hash Function Families.) Let $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}\}$ be a hash function family and Samp_k be a k-sampling* *algorithm of $\mathcal{H}$. We say that $\mathcal{H}$ is an (r,k)-subset-resilient hash function family ((r,k)-SRH) with regard to Samp_k such that for any probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function ϵ such that*

$$\Pr\left[\begin{array}{c} x \notin \{x_1,...,x_r\} \\ \mathsf{ORS}_k(x) \subseteq \bigcup\limits_{j\in[r]} \mathsf{ORS}_k(x_j) \end{array}\middle| \begin{array}{c} (h_1,...,h_k) \leftarrow \mathsf{Samp}_k(1^n) \\ (x,x_1,...,x_r) \leftarrow \mathcal{A}(1^n,h_1,...,h_k) \end{array}\right] \leq \epsilon(n) \tag{3.1}$$

holds for large enough n.

Next, we define a weaker assumption than subset resilience, which we call *restricted* subset resilience. Before introducing this assumption, we propose a definition called restricted subset cover, similar to subset cover. The difference is that for a restricted subset cover $(x,x_1,...,x_r)$, $h_i(x)$ is required to be covered by the union of $h_i(x_j)$ for each i. It is a sufficient but unnecessary condition for a subset cover.

Definition 16 *Let $H=(h_1,...,h_k)$ be a tuple of functions where $h_i : X \to Y$ for each $i \in [k]$. We say that $(x,x_1,...,x_r) \in X^{r+1}$ is an (r,k)-restricted subset cover of H if $h_i(x) \in \bigcup_{j\in[r]}\{h_i(x_j)\}$ for any $i \in [k]$.*

Definition 17 *(Restricted Subset-Resilient Hash Function Families.) Let $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}\}$ be a hash function family and Samp_k be a k-sampling algorithm of $\mathcal{H}$. We say that $\mathcal{H}$ is an (r,k)-restricted subset-resilient hash function family ((r,k)-rSRH) with regard to Samp_k such that for any probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function ϵ such that*

$$\Pr\left[\begin{array}{c} \forall i\in[k], x\neq x_i \\ h_i(x) \in \bigcup\limits_{j\in[r]}\{h_i(x_j)\} \end{array}\middle| \begin{array}{c} (h_1,...,h_k) \leftarrow \mathsf{Samp}_k(1^n) \\ (x,x_1,...,x_r) \leftarrow \mathcal{A}(1^n,h_1,...,h_k) \end{array}\right] \leq \epsilon(n), \tag{3.2}$$

holds for large enough n.

In the following, we focus on (k,k)-restricted subset resilience in particular, and simply call it *k-restricted subset resilience*. In this situation, finding an

(k, k)-restricted-subset cover for $H = (h_1, ..., h_k)$ is equivalent to finding a tuple $(x, x_1, ...x_k)$ such that $h_i(x) = h_i(x_i)$. Thus, (k, k)-restricted subset resilience can be redefined as follows:

Definition 18 *Let $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}\}$ be an efficient family ensemble and Samp_k be a k-sampling algorithm of $\mathcal{H}$. We say that $\mathcal{H}$ is a secure k-subset-resilient hash function family (k-rSRH) with regard to Samp_k such that for any probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function ϵ such that*

$$\Pr\left[\begin{array}{c|c} \forall i \in [k], x \neq x_i & (h_1, ..., h_k) \leftarrow \mathsf{Samp}_k(1^n) \\ h_i(x) = h_i(x_i) & (x, x_1, ..., x_k) \leftarrow \mathcal{A}(1^n, h_1, ..., h_k) \end{array}\right] \leq \epsilon(n), \tag{3.3}$$

holds for large enough n.

If $(x, x_1, .., x_k)$ is a (k, k)-restricted subset cover of $H = (h_1, ..., h_k)$, we simply say that it is a restricted subset cover of H.

3.1.3 SRH and Signature Schemes

SRH and rSRH are helpful to construct hash-based few-time signature schemes. For instance, the EU-CMA security of HORS [94] is based on SRH and one-way function families.

Lemma 6 *(HORS: Hash to Obtain Random Subset. [94]) Assuming there exists an (r, k)-subset-resilient hash function family $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^{m(n)} \to \{0,1\}^{m'(n)}\}$ and a one-way function family $\mathcal{F} = \{\mathcal{F}_n : \{0,1\}^{l(n)} \to \{0,1\}^{l'(n)}\}$, then there exists an existentially unforgeable signature scheme under r-time chosen message attacks such that:*

- *The message space is $\{0,1\}^m$.*
- *The size of the public key and the secret key are respectively $O(l'2^{m'})$ bits and $O(l2^{m'})$ bits.*
- *The size of a signature is $km'l$ bits.*

- *The key generation algorithm runs* $2^{m'}$ *one-way functions.*

In addition, HORS can be simply modified to be based on rSRH (instead of SRH) and one-wayness: instead of picking $2^{m'}$ number of random strings in the key generation algorithm, it picks k groups of them, each of which contains $2^{m'}$ random strings. Just like HORS, the public key includes $k2^{m'}$ number of values. To sign the message m, it leaks the element indexed $h_i(m)$ in group i for each $i \in [k]$. Compared to HORS, the size of the secret key and the public key becomes k times larger, and the running time of the key generation algorithm also becomes k times longer. However, it requires a weaker assumption, resisting weak message attacks [8].

Lemma 7 *Assuming there exists an* (r, k)*-restricted subset-resilient hash function family* $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^{m(n)} \to \{0,1\}^{m'(n)}\}$ *and a one-way function* $\mathcal{F} = \{\mathcal{F}_n : \{0,1\}^{l(n)} \to \{0,1\}^{l'(n)}\}$*, then there exists an existentially unforgeable signature scheme under* r*-time chosen message attacks such that:*

- *The message space is* $\{0,1\}^m$.
- *The size of the public key and the secret key are respectively* $O(kl'2^{m'})$ *bits and* $O(kl2^{m'})$ *bits.*
- *The size of a signature is* $km'l$ *bits.*
- *The key generation algorithm runs* $k2^{m'}$ *one-way functions.*

Proof.

Let $\mathcal{H}$ be an (r, k)-rSRH mapping $m(n)$-bit strings to $m'(n)$-bit strings and Samp_k be a k-sampling algorithm. Let $f : \{0,1\}^{l(n)} \to \{0,1\}^{l'(n)}$ be a one-way function. A signature scheme $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ is depicted as follows:

- $\mathsf{KeyGen}(1^n)$: $(h_1, ..., h_k) \leftarrow \mathsf{Samp}_k(\mathcal{H})$. Randomly pick $s_{i,j}$ from $\{0,1\}^l$ for all $i \in [k]$ and $j \in \{0,1\}^{m'}$. For each i and j, compute $y_{i,j} = h_i(s_{i,j})$. The public key pk contains $(h_1, ..., h_k)$ and all the $y_{i,j}$'s. The secret key sk contains $(h_1, ..., h_k)$ and all the $s_{i,j}$'s. Then, output (pk, sk).

- $\mathsf{Sign}(sk, m)$: For each $i \in [k]$, compute $t_i = h_i(m)$. Output $\sigma = (s_{1,t_1}, ..., s_{k,t_k})$.
- $\mathsf{Ver}(pk, m, \sigma)$: Parse $\sigma = (x_1, ..., x_k)$. For each $i \in [k]$, compute $t_i = h_i(m)$. If for each $i \in [k]$ it holds that $y_{i,t_i} = f(x_i)$, output 1. Otherwise, output 0.

The correctness of Γ can be easily verified. Next, we prove the security.

Suppose there exists an adversary $\mathcal{A}$ that can break EU-CMA security of Γ, we construct a reduction algorithm $\mathcal{R}^{\mathcal{A}}$ that can break one-wayness of f or restricted subset resilience of $\mathcal{H}$. Given y as the challenge of one-wayness and $(h_1, ..., h_k)$ as the challenge of restricted subset resilience, $\mathcal{R}$ randomly picks $s_{i,j}$ from $\{0,1\}^l$ for all $i \in [k]$ and $j \in \{0,1\}^{m'}$, and computes $y_{i,j} = h_i(s_{i,j})$. Then, it randomly picks (i', j') from $[k] \times \{0,1\}^{m'}$ and replaces $y_{i',j'}$ with the challenge y. after that, $\mathcal{R}$ runs $\mathcal{A}(1^n, pk)$.

When $\mathcal{A}$ queries to signing oracle, $\mathcal{R}$ replies as $\mathsf{Sign}(sk, \cdot)$ should do. $\mathcal{R}$ can perfectly similate the signing oracle unless it meets $s_{i',j'}$. In this case, $\mathcal{R}$ halts and restarts with fresh randomness. Since (i', j') is randomly chosen and $\mathcal{A}$ is required to query at most r times, the probability that $\mathcal{R}$ halts is at most $\frac{rk}{k2^{m'}} = r2^{-m'} \le c$. After approximate c^{-1} repetitions, $\mathcal{R}$ finally similates the signing oracle for $\mathcal{A}$.

After queries, $\mathcal{A}$ output (m^*, σ^*). Let $M = (m_1, ..., m_r)$ be the queries of $\mathcal{A}$. If $\mathcal{A}$ succeeds, it holds that $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$ and $m^* \notin M$. Let $\sigma^* = (x_1^*, ..., x_k^*)$. We consider the two cases if $\mathcal{A}$ succeeds:

- **Case 1:** There exists an $i \in [k]$ such that $h_i(m^*) \notin \bigcup_{j \in [r]}\{h_i(m_j)\}$. In this case, $\mathcal{R}$ never leaked any information of $s_{i,h_i(m^*)}$ to $\mathcal{A}$ except its image w.r.t. f. Since σ is valid, it holds that $y_{i,h_i(m^*)} = f(x_i^*)$. Thus, $\mathcal{A}$ actually inverts $y_{i,h_i(m^*)}$. Since (i', j') is randomly chosen, the event that $(i, h_i'(m))$ is equal to (i', j') occurs with probability at least $1/k2^{m'}$. If it happens, $\mathcal{R}$ successfully obtains an inverse of y. Since $2^{m'}$ is a polynomial of n, $\mathcal{R}$ inverts y with polynomial probability in this case.
- **Case 2:** It holds that $h_i(m^*) \in \bigcup_{j \in [r]}\{h_i(m_j)\}$ for each $i \in [k]$, which implies the fact that $(m, m_1, ..., m_r)$ is an (r, k)-resilient subset cover of $(h_1, ..., h_k)$.

In this case, $\mathcal{R}$ returns $(m, m_1, ..., m_r)$ as a result of breaking subset resilience of $\mathcal{H}$.

Denote by $\mathsf{Adv}_f^{\mathsf{OWF}}(n)$ the upper bound of the probability of breaking one-wayness of f and denote $\mathsf{Adv}_{\mathcal{H}}^{(r,k)\text{-}\mathsf{rSRH}}(n)$ the upper bound of the probability of breaking subset resilience of $\mathcal{H}$. We have

$$\begin{aligned}\Pr[\mathsf{Exp}_\Gamma^{\mathsf{EU\text{-}CMA}}(\mathcal{A})] &= \Pr[\mathsf{Exp}_\Gamma^{\mathsf{EU\text{-}CMA}}(\mathcal{A})|\textbf{Case 1}] + \Pr[\mathsf{Exp}_\Gamma^{\mathsf{EU\text{-}CMA}}(\mathcal{A})|\textbf{Case 2}] \\ &\leq k2^{m'}\mathsf{Adv}_f^{\mathsf{OWF}}(n) + \mathsf{Adv}_{\mathcal{H}}^{(r,k)\text{-}\mathsf{rSRH}}(n),\end{aligned}$$

which is negligible due to the assumptions. □

One may consider that the sizes of keys are extremely large and impractical. It is not a big issue since the public keys and the secret keys can be compressed by introducing a Merkle tree and a pseudorandom function, respectively. However, the running time of the key generation algorithm cannot be compressed.

Remark 2 *Essentially, the scheme in Lemma 7 is a very simplified version of FORS [17]. The differences are as follows. First, FORS introduces a Merkle tree structure to compress the public key. Second, it replaces the one-way functions with tweakable hash functions to decrease the security loss. Third, it introduces a randomizer on the message. That is, instead of computing* $h_i(m)$*, it picks a randomizer* r *and computes* $h_i(r||m)$*. Here* r *is computed by* $PRF(k, m)$ *where* PRF *is a pseudorandom function, and* k *is a secret key (in the randomized version of FORS,* r *is computed by* $PRF(k, \mathsf{rand}, m)$ *where* rand *is a random nonce). This step makes the security based on a "target version" of restricted subset resilience, a weaker assumption. See details in [17].*

Consider the case that the pseudorandom function key k *is leaked in the deterministic version of FORS. We have* $h_i(r||m) = h_i(PRF(k, m)||m)$*. Denote* $h'_i(x) = h_i(PRF(k, m)||m)$*. Then, the EU-CMA security of FORS will be degraded to restricted subset resilience of* $H' = (h'_1, ..., h'_k)$*.*

Apart from HBS schemes, since SRH has similar properties to cover-free families (CFFs), it can also be used in other CFF-based primitives. For example, in [105] and [63], the authors respectively propose an aggregate signature scheme and a programmable hash function based on CFF. By replacing the CFF with an (r, k)-SRH in these primitives, we obtain an r-time aggregate signature scheme and an $(r, 1)$-programmable hash function. However, this will lead to larger public keys and do not behave better than the original versions. It is an open question about other practical applications of SRH besides HBS schemes.

3.2 Quantum Algorithms Breaking k-rSRH

This section gives a quantum algorithm for finding k-restricted subset covers, showing an upper bound of k-restricted subset resilience security for hash functions in the quantum world. This work is inspired by [84], which shows a quantum algorithm finding multi-collisions. Interestingly, the time and memory complexity for finding a k-restricted subset cover are roughly the same as those needed for finding a $(k + 1)$-collision.

In this section, we treat the target function as an oracle. We suppose that an algorithm can only obtain the target function value by querying this oracle. The time complexity is evaluated by the number of (quantum) queries to this oracle.

3.2.1 The First Attempt

To warm up, we first show a quantum algorithm finding 2-restricted subset cover for two 2-to-1 functions. Given $H = (h_1, h_2)$ where $h_i : Dom \to Rng$ are 2-to-1 functions for each i, let $|Dom| = 2|Rng| = 2N$. The quantum algorithm runs as follows:

1. Let $t_1 = N^{3/7}$. Pick a list $X_1 = \{x_1^{(1)}, ..., x_1^{(t_1)}\}$, where $x_1^{(\cdot)}$ is uniformly chosen from Dom.
2. Evaluate $y_1^{(i)} = h_1(x_1^{(i)})$ for $i \in [t_1]$. List them in $Y_1 = \{y_1^{(1)}, ..., y_1^{(t_1)}\}$. This step requires $N^{3/7}$ queries to H.

3. Define $F_1 : \mathit{Dom} \to \{0,1\}$:

$$F_1(x) = \begin{cases} 1, & x \notin X_1 \text{ and } h_1(x) \in Y_1, \\ 0, & \text{otherwise}. \end{cases} \tag{3.4}$$

   Run Grover's algorithm on F_1. Note that $|F_1^{-1}(1)| = t_1$, and evaluating F_1 needs a query to H. Grover's algorithm returns a solution after at most $O(\sqrt{2N/t_1}) = O(N^{2/7})$ queries to H.

4. Let $t_2 = N^{1/7}$. Repeat step 3 t_2 times. It requires $O(N^{2/7} \cdot N^{1/7}) = O(N^{3/7})$ queries to H. List the result as $X_2 = \{x_2^{(1)}, ..., x_2^{(t_2)}\}$.

5. Evaluate $y_2^{(i)} = h_2(x_2^{(i)})$ for $i \in [t_2]$. List them in $Y_2 = \{y_2^{(1)}, ..., y_2^{(t_2)}\}$. This step requires $N^{1/7}$ queries to H.

6. Define $F_2 : \mathit{Dom} \to \{0,1\}$:

$$F_2(x) = \begin{cases} 1, & x \notin X_2 \text{ and } h_2(x) \in Y_2, \\ 0, & \text{otherwise}. \end{cases} \tag{3.5}$$

   Run Grover's algorithm on F_2. Note that $|F_2^{-1}(1)| = t_2$. Grover's algorithm returns a solution x^* after at most $O(\sqrt{2N/t_2}) = O(N^{3/7})$ queries to H.

7. Find $i \in [t_2]$ such that $h_2(x^*) = h_2(x_2^{(i)})$. $(x^*, x_2^{(i)})$ is a collision of h_2.

8. Find $j \in [t_1]$ such that $h_1(x_2) = h_1(x_1^{(j)})$ (there exists such a j since $F_1(x_2) = 1$ for every $x_2^{(j)} \in X_2$). $(x_1^{(j)}, x_2^{(i)})$ is a collision of h_1.

9. Output $(x_2^{(i)}, x_1^{(j)}, x^*)$.

Since $(x_1^{(j)}, x_2^{(i)})$ is a collision of h_1 and $(x^*, x_2^{(i)})$ is a collision of h_2, $(x_2^{(i)}, x_1^{(j)}, x^*)$ is a 2-restricted subset cover w.r.t. (h_1, h_2). In total, the algorithm requires $O(N^{3/7} + N^{3/7} + N^{1/7} + N^{3/7}) = O(N^{3/7})$ queries to H.

To "inject" the third function into the algorithm, we repeat the steps a little bit more times than before and slightly change some of the steps. We first pick $N^{7/15}$ elements from the domain and list them in X. After evaluating their images of h_1 and running Grover's algorithm repeatedly, we find collisions of a subset of X w.r.t. h_1. We remove from the X the elements whose collisions are not found. Next, we evaluate the images of X w.r.t. h_2, and run Grover's algorithm repeatedly again. As a result, we obtain a subset of X, of which the collision are found w.r.t. h_2 and h_1. Then, we do the same things on h_3, and get the final result. It implies a 3-restricted subset cover of (h_1, h_2, h_3).

Our quantum algorithm on finding 3-restricted subset cover for $H = (h_1, h_2, h_3)$ is depicted as follows:

1. Let $t_1 = N^{7/15}$ (instead of $N^{3/7}$ in the case of two functions). Pick a list $X_1 = \{x_1^{(1)}, ..., x_1^{(t_1)}\}$, where $x_1^{(\cdot)}$ is uniformly chosen from Dom.

2. Evaluate $y_1^{(i)} = h_1(x_1^{(i)})$ for $i \in [t_1]$ and list them in $Y_1 = \{y_1^{(1)}, ..., y_1^{(t_1)}\}$. This step requires $N^{7/15}$ queries to H.

3. Define $F_1 : Dom \to \{0, 1\}$:

$$F_1(x) = \begin{cases} 1, & x \notin X_1 \text{ and } h_1(x) \in Y_1, \\ 0, & \text{otherwise}. \end{cases} \tag{3.6}$$

   Run Grover's algorithm on F_1. It returns after at most $O(\sqrt{2N/t_1}) = O(N^{4/15})$ queries to H.

4. Let $t_2 = N^{3/15}$ (instead of $N^{1/7}$). Repeat the last step t_2 times. Here, we list the solutions in the list X_2 orderedly. Every time Grover's algorithm returns a solution x, we evaluate $h_1(x)$ and find the $y_1^{(i)} \in Y_1$. Then, denote the solution x by $x_2^{(i)}$ and list it in X_2.

   Note that $(x_1^{(i)}, x_2^{(i)})$ is a collision of h_1. Next, we remove from X_1 all the $x_1^{(i)}$'s such that $x_2^{(i)}$ does not exist in X_2. Now all the elements of X_1 can find their second-preimages with regard to h_1 in X_2 with the same labels.

   This step requires $O(N^{4/15} \cdot N^{3/15} + N^{3/15}) = O(N^{7/15})$ queries to H.

5. Here, Y_2 is no longer the images of X_2 with regard to h_2. Instead, we evaluate $h_2(x_1^{(i)})$ for each $x_1^{(i)} \in X_1$ and list them in Y_2. Since now X_1 only contains t_2 elements, this step requires $t_2 = N^{3/15}$ queries to H.

6. Define $F_2 : Dom \to \{0,1\}$:

$$F_2(x) = \begin{cases} 1, & x \notin X_1 \text{ and } h_2(x) \in Y_2, \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

   Run Grover's algorithm on F_2. It returns a solution after at most $O(\sqrt{N/t_2}) = O(N^{6/15})$ queries to H.

7. Let $t_3 = N^{1/15}$. Repeat the last step t_3 times. Similar to step 4, list the solution of Grover's algorithm $x_3^{(i)}$ in X_3 orderedly. Again, remove from X_1 the elements whose second-preimages are not found in X_3. Now for every $x_1^{(i)} \in X_1$, $(x_1^{(i)}, x_3^{(i)})$ is a collision of h_2. This step requires $O(N^{1/15} \cdot N^{6/15} + N^{1/15}) = O(N^{7/15})$ queries to H.

8. Evaluate $h_3(x_1^{(i)})$ for each $x_1^{(i)} \in X_1$, and list them in Y_3. This step requires $O(N^{1/15})$ queries to H.

9. Define $F_3 : Dom \to \{0,1\}$:

$$F_2(x) = \begin{cases} 1, & x \notin X_1 \text{ and } h_3(x) \in Y_3, \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

   Run Grover's algorithm on F_3. It returns a solution x_4 after at most $O(\sqrt{2N/t_3}) = O(N^{7/15})$ queries to H.

10. Find $y_3^{(i)} = h_3(x_4)$ in Y_3. Output $(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4)$.

In total, the algorithm requires $O(N^{7/15})$ queries to H.

Generally, for any constant k, let $t_s = N^{(2^{k-s+1}-1)/(2^{k+1}-1)}$ for each $s \in \{1, ..., k+1\}$. Let $H = (h_1, ..., h_k)$ be a tuple of 2-to-1 functions, where $h_i : 2X \to Y$ and $|Dom| = 2|Rng| = 2N$. Our quantum algorithm finding k-restricted subset cover for H is as follows:

1. Pick a list $X_0 = \{x^{(1)}, ..., x^{(t_1)}\}$, where $x^{(\cdot)}$ is uniformly chosen from Dom.

2. For $s = 1$ to k:

(a) For any $x^{(i)} \in X_{s-1}$, evaluate $y_s^{(i)} = h_s(x^{(i)})$ and list them in Y_s.

(b) Define by $F_s : Dom \to \{0,1\}$ a function such that $F_s(x) = 1$ if and only if $x \notin X_{s-1} \wedge h_s(x) \in Y_s$. Run Grover's algorithm on F_s t_{s+1} times.

(c) For each solution x' of Grover's algorithm, find $y_s^{(i)} \in Y_s$ such that $y_s^{(i)} = h_s(x')$. Denote $x_s^{(i)} \triangleq x^{(i)}$ and $x_s'^{(i)} \triangleq x'$. Note that $(x^{(i)}, x_s'^{(i)})$ is a collision w.r.t. h_s. (If x' repeatedly apprears, discard it and run Grover's algorithm again.)
List all the $x_s^{(i)}$ in X_s and all the $x_s'^{(i)}$ in X_s'.

3. After k repetitions, X_k contains only one element $x_k^{(i)} = x^{(i)}$. Output $(x^{(i)}, x_1'^{(i)}, ..., x_k'^{(i)})$.

Remark 3 *In practice, $(h_1, ..., h_k)$ is usually instantiated as a division of a long hash $H : Dom \leftarrow Rng^k$. In this case, the hash values of x w.r.t. $h_1, ..., h_k$ can be computed by a single hash query. Thus, if the memory is large enough, we can compute all the $Y_1, ..., Y_k$ right after step 1 by t_1 hash computations and skip step 2(a) in the loops. This modification can decrease the time cost (but the complexity is not changed).*

Now we analyze the time complexity in the above algorithm. Step 2(a) requires t_s classical queries. Step 2(b) requires $O(t_{s+1}\sqrt{N/t_s})$ quantum queries. Step 2(c) requires t_{s+1} classical queries. The number of quantum queries required in the algorithm is in total

$$\sum_{s=1}^{k} t_{s+1}\sqrt{\frac{N}{t_s}} = \sum_{s=1}^{k} N^{\frac{2^{k-s}-1}{2^{k+1}-1}+\frac{1}{2}(1-\frac{2^{k-s+1}-1}{2^{k+1}-1})} = \sum_{s=1}^{k} N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})} = kN^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})}. \tag{3.9}$$

The number of classical queries required in the algorithm is in total

$$\sum_{s=1}^{k} t_s + t_{s+1} < \sum_{s=1}^{k} 2t_1 = 2kN^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})}. \tag{3.10}$$

Since k is a constant, we conclude the following statement.

Theorem 1 *For constant $k \geq 2$, let $H = (h_1, ..., h_k)$ be a tuple of 2-to-1 functions where $h_i : X \to Y$ and $|X| = 2|Y| = 2N$ for each i. There exists an algorithm finding a k-restricted subset cover for H using $O(N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})})$ quantum queries to H.*

Note that here the functions are required to be 2-to-1. Namely, it guarantees that any $x \in Dom$ has a second-preimage of h_s and thus $|F_s^{-1}(1)|$ is exactly equal to $|Y_s| = t_s$. For a general function, there may exist some "bad" elements in Dom which have no second-preimage w.r.t. some h_i. If we unfortunately pick bad elements into X_{s-1}, $|F_s^{-1}(1)|$ will be smaller than what we expected, making Grover's algorithm fail in the expected steps. In other words, for a general function, we need to ensure that in each loop, X_{i-1} always contains enough "good" elements which have second-preimages w.r.t. each h_i.

Next, we try to eliminate the need of 2-to-1 property. We require that the size of the domain is $(k+1)$ times larger than that of the range. In this case, a constant fraction of x's have their second-preimages w.r.t. each h_i.

Lemma 8 *Let $H = (h_1, ..., h_k)$ where $h_i : Dom \to Rng$ for each h_i and $|Dom| = (k+1)|Rng| = (k+1)N$. The probability that x has a second-preimage w.r.t. h_i for each i is at least $\frac{1}{k+1}$, where the probability is taken over the uniform choice of $x \in Dom$.*

Proof. Let $Dom_{h_i} \subset Dom$ be the set of x that does not have a second-preimage w.r.t. h_i. We observe that $|Dom_{h_i}| \leq N$, otherwise the elements not contained in Dom_{h_i} cannot find images w.r.t. h_i in Rng. Thus we have $|\bigcup_{i\in[k]} Dom_{h_i}| \leq kN$ and

$$\Pr_x[x \notin \bigcup_{i\in[k]} Dom_{h_i}] = \frac{|Dom| - |\bigcup_{i\in[k]} Dom_{h_i}|}{|Dom|} \geq \frac{(k+1)N - kN}{kN} = \frac{1}{k}, \quad (3.11)$$

This completes the proof. □

Then, we require $H = (h_1, ..., h_k)$ is a tuple of general functions where $|Dom| \geq (k+1)|Rng|$ and improve our algorithm. Indeed, we only need to focus on the case that $|Dom| = (k+1)|Rng|$. If $|Dom| > (k+1)|Rng|$, we can randomly choose a subset $Dom' \subseteq Dom$ such that $|Dom'| = (k+1)|Rng|$, and then run the algorithm in the former case.

We slightly change some of the steps in our algorithm. Let $c > 0$ be a constant. In step 1, We pick $(1+c)kt_1$ number of $x^{(i)}$'s from Dom (instead of t_1

number of them in the previous version). In step 2(b) of loop s, we run Grover's algorithm on F_s $(1+c)kt_{s+1}$ times rather than t_{s+1} times.

Let S_H be the set of x that has a secone-preimage w.r.t. each h_i (which implies $S_H = Dom \backslash \bigcup_{i\in[k]} Dom_{h_i}$). Note that the elements of X_0 are uniformly chosen. Due to Chernoff bound, there are at least $\frac{1}{(1+c)k}$ fraction of x's in X with overwhelming probability.

$$\Pr[|F_1^{-1}(1)| < t_1] < \Pr[|X_1 \cup S_H| < t_1] < e^{-\frac{c^2 t_1}{2}}, \tag{3.12}$$

which is negligible since $t_1 = N^{\frac{2^k-1}{2^{k+1}-1}}$ and c is a constant.

Suppose $|F_1^{-1}(1)| \geq t_1$. Grover's algorithm successfully runs on F_1 in step 2(b) of loop 1.

When we run Grover's algorithm on F_1, it randomly picks an element from $F_1^{-1}(1)$, which corresponds to a uniformly random element from X_0. Since X_0 is uniformly chosen in step 1, the distribution of each element in X_1 is also uniform. Note that the size of X_1 is $(1+c)kt_2$. Again due to Chernoff bound, there are at least t_2 number of x's in X_1 such that x drops in S_H with overwhelming probability. It implies that $|F_2^{-1}(1)| \geq t_2$ holds with overwhelming probability. Suppose it holds. Then, Grover's algorithm in step 2(b) of loop 2 can be completed with the expected number of queries.

Similarly, $|F_s^{-1}(1)| \geq t_s$ holds with overwhelming probability for each s. Suppose it always holds for each s, the algorithm will go through and output a k-restricted subset cover for H. Note that k and c are constant, and the number of quantum queries is $(1+c)k$ times larger than the previous one. The total number of queries is still $O(N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})})$.

Thus, we have the following statement:

Theorem 2 *For constant $k \geq 2$, let $H = (h_1, ..., h_k)$ be a tuple of functions where $h_i : Dom \to Rng$ and $|Dom| \geq (k+1)|Rng| = (k+1)N$. There exists an algorithm finding a k-restricted subset cover for H with overwhelming probability using $O(N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})})$ quantum queries to H.*

Remark 4 *Indeed, this algorithm also works in the case that the ranges of functions are not identical, in other words, the case that $h_i : Dom \to Rng_i$,* *where* $|Dom| \geq (k+1)|Rng_i| = (k+1)N$ *but* Rng_i *may differ from* Rng_j *for* $i \neq j$.

3.2.2 A time-memory tradeoff

In the last subsection, we show an algorithm finding k-restricted subset covers which requires $O(N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})})$ quantum queries to the functions. However, we observe that the memory required in this algotithm is also $O(N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})})$ (the memory cost is measured by the number of preimages and images that are necessary to be stored in the database). It is because that the algorithm stores $|X_0| = t_1 = N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})}$ elements of the domain in step 1.

Note that the memory cost mainly depends on t_1. We can flexibly adapt $|X_0| = t_1$ and other t_s for $s \in \{2, ..., t+1\}$ to decrease the memory cost, but increase the running time.

We redifine $t_s = t_1^{(2^{k-s+1}-1)/(2^k-1)}$ for each $s \in \{2, ..., k+1\}$ for fixed t_1. (The original version can be considered a specific case that $t_1 = N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})}$.) As the result, the expected number of quantum queries required in step 2(b) becomes

$$\sum_{s=1}^{k} t_{s+1}\sqrt{\frac{N}{t_s}} = \sum_{s=1}^{k} N^{\frac{1}{2}(1-\frac{\log_N t_1}{2^k-1})} = kN^{\frac{1}{2}(1-\frac{\log_N t_1}{2^k-1})}. \tag{3.13}$$

In total, the time complexity becomes $O(t_1) + O(N^{\frac{1}{2}(1-\frac{\log_N t_1}{2^k-1})})$.

If $t_1 < N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})}$, the algorithm will require less memories and more running time. When t_1 becomes a polynomial of $\log N$, then the running time becomes close to $O(N^{1/2})$, which is the time complexity of simply running Grover's algorithms.

3.3 Constructing dCRH from k-rSRH

In this section, we show that the existence of k-rSRH implies the existence of dCRH. This work is inspired by [78], which shows a similar relation between dCRH and MCRH. Note that our construction is non-black-box. That is, we do not present an explicit construction of dCRH from k-rSRH, but only prove the existence of dCRH instead.

3.3.1 From 2-rSRH to dCRH

In this subsection, we prove a weaker statement: the existence of 2-rSRH implies the existence of dCRH.

Theorem 3 *Assuming the existence of a secure 2-rSRH such that each of functions compresses $2n$ bits to n bits, then there exists an (infinity often) secure dCRH.*

Proof. To prove this statement, we will show the contradiction that if infinity-often secure dCRH does not exist, then there does not exist a secure 2-rSRH with regard to any 2-sampling algorithm. We assume that there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ that breaks distributional collision resistance of *any* hash function family. Then, we construct a polynomial-time algorithm BreakSR to break 2-restricted subset resilience of any $\mathcal{H}$ with regard to any 2-sampling algorithm Samp.

Given $\mathcal{H}$ and Samp, let $(\mathcal{D}_1, \mathcal{D}_2)$ be the distribution of the output of $\mathsf{Samp}(1^n)$. Note that $\mathcal{D}_1$ and $\mathcal{D}_2$ are two distributions of $\mathcal{H}_n$. Due to our hypothesis, there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ and two negligible functions δ and ϵ such that for large enough n, it holds that

$$\Pr_{h_1 \leftarrow \mathcal{D}_1}[\Delta(\mathcal{A}(1^n, h_1), \mathsf{COL}_{h_1}) \leq \delta(n)] > 1 - \epsilon(n), \tag{3.14}$$

and thus

$$\Pr_{(h_1,h_2)\leftarrow \mathsf{Samp}}[\Delta(\mathcal{A}(1^n, h_1), \mathsf{COL}_{h_1}) \leq \delta(n)] > 1 - \epsilon(n). \tag{3.15}$$

Let r be the randomness of $\mathcal{A}$. Here, $\mathcal{A}$ is given the security parameter, a function h_1 sampled from $\mathcal{D}_1$ and the randomness r, then it outputs a collision that is statistically close to COL_h with all but negligible probability over the choice of h. Let $(x_1, x_2) \leftarrow \mathcal{A}(1^n, h_1; r)$ and denote by $\mathcal{A}^1$ the deterministic algorithm with input $(1^n, h_1, r)$ and output x_1.

We omit the security parameter 1^n in the following. Note that fixing h_1 in the input of $\mathcal{A}^1$, $\mathcal{A}^1$ becomes a deterministic algorithm whose input is r and output is $x \in \{0,1\}^{2n}$. Without loss of generality, suppose the length of the randomness of $\mathcal{A}$ is at most $l_r(n) > 2n$. For $(h_1, h_2) \leftarrow \mathsf{Samp}$, we define a special function $h'_2 : \{0,1\}^{l_r(n)} \to \{0,1\}^n$ as follows:

$$h'_2(r) \triangleq h_2(\mathcal{A}^1(h_1; r)). \tag{3.16}$$

It is not hard to observe that h'_2 is samplable by Samp and it is efficiently computable.

Due to our hypothesis that no dCRH exists, there exists another probabilistic polynomial-time algorithm $\mathcal{A}'$ that can find uniform collisions for h'_2. That is, there exist two negligible functions δ' and ϵ' such that

$$\Pr_{(h_1,h_2)\leftarrow\mathsf{Samp}}[\Delta(\mathcal{A}'(1^n, h'_2), \mathsf{COL}_{h'_2}) \leq \delta'(n)] > 1 - \epsilon'(n) \tag{3.17}$$

holds for large enough n.

Due to the existence of $\mathcal{A}$ and $\mathcal{A}'$, we can construct an algorithm $\mathsf{BreakSR}(1^n, h_1, h_2)$ to output a 2-restricted subset cover w.r.t. (h_1, h_2):

1. Define $h'_2(r) \triangleq h_2(\mathcal{A}^1(h_1; r))$.
2. $(r_1, r_2) \leftarrow \mathcal{A}'(h'_2)$.
3. $(x_1, x_2) \leftarrow \mathcal{A}(h_1; r_1)$.
4. $(x_3, x_4) \leftarrow \mathcal{A}(h_1; r_2)$.
5. Output (x_1, x_2, x_3).

Note that if $\mathcal{A}$ succeeds in finding collisions in the process of $\mathsf{BreakSR}(1^n, h_1, h_2)$, we have $h_1(x_1) = h_1(x_2)$ and $h_1(x_3) = h_1(x_4)$. In addition, if $\mathcal{A}'$ succeeds as well, we have $h'_2(r_1) = h'_2(r_2)$ and thus $h_2(\mathcal{A}^1(h_1, r_1)) = h_2(\mathcal{A}^1(h_1, r_2))$. Due to the definition of $\mathcal{A}^1$, it holds that $A^1(h_1, r_1) = x_1$ and $A^1(h_1, r_2) = x_3$. Thus, we have $h_2(x_1) = h_2(x_3)$. As a result, we have $h_1(x_1) = h_1(x_2)$ and $h_2(x_1) = h_2(x_3)$, which implies the fact that (x_1, x_2, x_3) is a 2-restricted subset cover of (h_1, h_2).

Formally, we aim to prove that there exists a negligible function μ such that

$$\Pr\left[\begin{array}{c|c} x_1 \neq x_2, x_1 \neq x_3 & (h_1,h_2) \leftarrow \mathsf{Samp}(1^n) \\ h_1(x_1) = h_1(x_2), h_2(x_1) = h_2(x_3) & (x_1,x_2,x_3) \leftarrow \mathsf{BreakSR}(1^n, h_1, h_2) \end{array}\right] > 1 - \mu(n) \quad (3.18)$$

holds for large enough n.

Define the above experiment by **Game 0**. We show the probability of **Game 0** is overwhelming in the following steps:

- **Game 1** differs **Game 0** in the following parts. BreakSR does not run $(r_1, r_2) \leftarrow \mathcal{A}'(h_2')$ in step 2. Instead, it directly picks $(r_1, r_2) \leftarrow \mathsf{COL}_{h_2'}$. Due to equation (3.17), the statistical distance of **Game 0** and **Game 1** is less than $\delta'(n)$ except with probability $\epsilon'(n)$ (over the choice of h_2'). We have

  $$\Pr[\Delta(\textbf{Game 1}, \textbf{Game 0}) \leq \delta'(n)] > 1 - \epsilon'(n), \quad (3.19)$$

  and thus

  $$|\Pr[\textbf{Game 1}] - \Pr[\textbf{Game 0}]| < \epsilon'(n) + \delta'(n), \quad (3.20)$$

  where the probability is taken over the choice of (h_1, h_2) and the randomness of BreakSR.

- In **Game 1**, since $(r_1, r_2) \leftarrow \mathsf{COL}_{h_2'}$, it holds that $h_2(\mathcal{A}^1(h_1; r_1)) = h_2(\mathcal{A}^1(h_1; r_2))$ and thus $h_2(x_1) = h_2(x_3)$ with probability 1. Next, we prove that the other three events in equation (3.18) also occur with overwhelming probability.

  Lemma 9

  $$\Pr_{(h_1,h_2),COL_{h_2'}}[h_1(x_1) \neq h_1(x_2) \vee x_1 = x_2] < \epsilon(n) + \delta(n) + 2^{-n/2+1}. \quad (3.21)$$

  Proof.

  In **Game 1**, r_1 is the first element of a sample from $\mathsf{COL}_{h_2'}$, which means that r_1 is uniform from $\{0,1\}^{l_r(n)}$. Recall that $(x_1, x_2) \leftarrow \mathcal{A}(h_1; r_1)$. Thus, the probability in equation (3.21) is essentially taken over the choice of h_1

and r_1. Suppose $\mathcal{A}(h_1)$ and COL_{h_1} are $\delta(n)$-close (except with probability $\epsilon(n)$ over the choice of h_1 due to equation (3.15)). We have

$$\Pr_{(x_1,x_2)\leftarrow\mathcal{A}(h_1;r_1)}[h_1(x_1) \neq h_1(x_2) \vee x_1 = x_2] \leq \Pr_{(x_1',x_2')\leftarrow\mathsf{COL}_{h_1}}[h_1(x_1') \neq h_1(x_2') \vee x_1' = x_2'] + \delta(n) \tag{3.22}$$

For $(x_1', x_2') \leftarrow \mathsf{COL}_{h_1}$, $h_1(x_1) \neq h_1(x_2')$ holds with probability 0. Next, we show that

$$\Pr_{(x_1',x_2')\leftarrow\mathsf{COL}_{h_1}}[x_1 = x_2] < 2^{-n/2+1}. \tag{3.23}$$

For any $y \in \{0,1\}^n$, denote by $X_y^{h_1} \subseteq \{0,1\}^{2n}$ the set of x such that $h_1(x) = y$. For convenience, we say X_y instead of $X_y^{h_1}$ in the following.

We say X_y is "large" if $|X_y| \geq 2^{n/2}$ or X_y is "small" otherwise. Since X_y's are disjoint for different y, there are at most 2^n different X_y's. Thus, there exist at most $2^{3n/2}$ number of x such that $X_{h_1(x)}$ is small (otherwise the number of bad X_y's will be more than 2^n). As a result, the number of x's such that $X_{h_1(x)}$ is large is more than $2^{2n} - 2^{3n/2}$. We have

$$\Pr_{x\leftarrow\{0,1\}^{2n}}[X_{h_1(x)} \text{ is large}] \geq \frac{2^{2n} - 2^{3n/2}}{2^{2n}} = 1 - 2^{-n/2}. \tag{3.24}$$

Thus,

$$\begin{aligned}\Pr_{(x_1',x_2')\leftarrow\mathsf{COL}_{h_1}}[x_1' \neq x_2'] &\geq \Pr_{x_1'\leftarrow\{0,1\}^{2n}}[X_{h_1(x_1')} \text{ is large}] \Pr_{x_2'\leftarrow X_{h(x_1')}}[x_1' \neq x_2' | X_{h_1(x_1')} \text{ is large}] \\ &\geq (1 - 2^{-n/2})(1 - 2^{-n/2}) > 1 - 2^{-n/2+1}.\end{aligned}$$

From equation (3.22) and (3.23), we complete the proof of Lemma 9.

□

Lemma 10

$$\Pr_{(h_1,h_2),COL_{h_2'}}[x_1 = x_3] < \epsilon(n) + 2\delta(n) + 2^{-n/2+1} \tag{3.25}$$

Proof. Again, we assume that $\Delta(\mathcal{A}(h_1), \mathsf{COL}_{h_1}) \leq \delta(n)$ (except with probability $\epsilon(n)$). Then, we have

$$\sum_{x_1 \in \{0,1\}^{2n}} \left| \Pr_{r_1 \in \{0,1\}^{l_r(n)}} [x_1 \leftarrow \mathcal{A}^1(h_1; r_1)] - \frac{1}{2^{2n}} \right| \leq \delta(n). \tag{3.26}$$

For any $x \in \{0,1\}^{2n}$, denote by $R_x \subseteq \{0,1\}^{l_r(n)}$ the set of random coins making $\mathcal{A}$ output x as the first element:

$$R_x \triangleq \{r | \mathcal{A}^1(h_1; r) = x\}. \tag{3.27}$$

Then, we have

$$\sum_{x_1 \in \{0,1\}^{2n}} \left| \frac{|R_{x_1}|}{2^{l_r(n)}} - \frac{1}{2^{2n}} \right| \leq \delta(n). \tag{3.28}$$

Therefore, the mapping from r_1 to x_1 is regular except with probability $\delta(n)$.

Recall how $(r_1, r_2) \leftarrow \mathsf{COL}_{h_2'}$ and x_1 are chosen. First, we uniformly choose r_1 from $\{0,1\}^{l_r(n)}$ and let $x_1 = \mathcal{A}^1(h_1; r_1)$. Then, r_2 is uniformly chosen from the following set:

$$S_{x_1} = \{r | h_2(\mathcal{A}^1(h_1; r)) = h_2(x_1)\}. \tag{3.29}$$

That is, S_{x_1} is the set of r which maps to x' where $h_2(x') = h_2(x_1)$. Let $X_y^{h_2}$ be the set of x such that $h_2(x) = y$. We have

$$S_{x_1} = \bigcup_{x' \in X_{h_2(x_1)}^{h_2}} R_{x'}. \tag{3.30}$$

For convenience, we say $X_{h_2(x_1)}$ instead of $X_{h_2(x_1)}^{h_2}$ in the following.

Fix r_1. Obviously, we have $r_1 \in R_{x_1} \subset S_{x_1}$. In addition, recall that $x_3 = \mathcal{A}^1(h_1; r_2)$. Thus, $x_3 = x_1$ holds if and only if r_2 also drops in R_{x_1}. It occurs with probability $|R_{x_1}|/|S_{x_1}|$ (over the choice of r_2). Note that the mapping between r_1 and x_1 is regular and S_{x_1} contains $|X_{h_2(x_1)}|$ number of $R_{x'}$. We have

$$\left| \frac{|R_{x_1}|}{|S_{x_1}|} - \frac{1}{|X_{h_2(x)}|} \right| \leq \delta(n). \tag{3.31}$$

The above inequality is for a fixed r_1. Since r_1 is uniformly chosen, the distribution of x_1 is $\delta(n)$-close to the uniform distribution. Thus,

$$\Pr_{r_1}[X_{h_2(x_1)} \text{ is large}] \geq \Pr_{x_1}[X_{h_2(x_1)} \text{ is large}] - \delta(n) \geq 1 - 2^{-n/2} - \delta(n), \quad (3.32)$$

where the second inequality is due to equation (3.24).

From equation (3.31) and (3.32) we have

$$\begin{aligned}\Pr_{\mathsf{COL}_{h'_2}}[x_3 = x_1] &\leq \Pr_{r_2 \leftarrow S_{x_1}}[x_3 = x_1 | X_{h_2(x_1)} \text{ is large}] + \Pr_{r_1}[\overline{X_{h_2(x_1)} \text{ is large}}] \\ &\leq \delta(n) + \frac{1}{2^{n/2}} + \delta(n) + 2^{-n/2} = 2\delta(n) + 2^{-n/2+1}.\end{aligned}$$

Considering the choice of (h_1, h_2), we get

$$\Pr_{(h_1,h_2),\mathsf{COL}_{h'_2}}[x_3 = x_1] < \epsilon(n) + 2\delta(n) + 2^{-n/2+1}, \quad (3.33)$$

which completes the proof of Lemma 10. □

From Lemma 9 and 10 we have

$$\Pr[\textbf{Game 1}] \geq 1 - \epsilon(n) - 3\delta(n) - 2^{-n/2+2}, \quad (3.34)$$

where the factor of $\epsilon(n)$ is not accumulated since the proofs of two lemmas begin with the same assumption that $\mathcal{A}(h_1)$ and COL_{h_1} are $\delta(n)$-close.

To sum up, letting $\mu(n) = \epsilon'(n) + \delta'(n) + \epsilon(n) + 3\delta(n) + 2^{-n/2+2}$, inequality (3.18) holds. This completes the proof. □

3.3.2 From general k-rSRH to dCRH

In the last subsection, we construct an algorithm breaking the security of 2-rSRH with an algorithm breaking dCRH. Indeed, this construction can also be extended to break the security of k-rSRH for any constant $k > 2$. This implies the relation between dCRH and general k-rSRH.

Our extension is overviewed as follows. First, we construct a machine $\mathsf{BreakSR}$ breaking 2-rSRH with $\mathcal{A}$ breaking dCRH as we present in the last subsection. Next, we construct a machine $\mathsf{Break\text{-}3\text{-}SR}$ breaking 3-rSRH with $\mathsf{BreakSR}$ and $\mathcal{A}$. Iteratively, we construct a machine $\mathsf{Break\text{-}}(s+1)\mathsf{\text{-}SR}$ breaking $(s+1)$-rSRH with $\mathsf{Break\text{-}}s\mathsf{\text{-}SR}$ and $\mathcal{A}$ for $s = 2, ..., k-1$. Finally, we obtain a $\mathsf{Break\text{-}}k\mathsf{\text{-}SR}$ breaking k-rSRH, which proves our statement.

The induction from s to $s+1$ is similar to the construction of $\mathsf{BreakSR}$ from $\mathcal{A}$. Consider a simple case that $s = 2$. Given $(h_1, h_2, h_3) \leftarrow \mathsf{Samp}_3(1^n)$, define $h'_3(r) = h_3(\mathsf{BreakSR}^1(1^n, h_1, h_2))$, where $\mathsf{BreakSR}^1(\cdot)$ is the first element of the output of $\mathsf{BreakSR}(\cdot)$. Next, run $(r_1, r_2) \leftarrow \mathcal{A}_{\mathsf{dCRH}}(1^n, h'_3)$. After that, run $(x_1, x_2, x_3) \leftarrow \mathsf{BreakSR}(1^n, h_1, h_2; r_1)$ and $(x_4, x_5, x_6) \leftarrow \mathsf{BreakSR}(1^n, h_1, h_2; r_2)$. Finally, output (x_1, x_2, x_3, x_4) as a 3-restricted subset cover of (h_1, h_2, h_3).

Formally, the recursive algorithm $\mathsf{Break\text{-}}(s+1)\mathsf{\text{-}SR}(1^n, h_1, h_2, ..., h_{s+1})$ breaking $(s+1)$-rSRH runs as follows:

1. Define $h'_{s+1}(r) \triangleq h_{k+1}(\mathsf{Break\text{-}}s\mathsf{\text{-}SR}^1(1^n, h_1, h_2, ..., h_s))$.
2. $(r_1, r_2) \leftarrow \mathcal{A}_{\mathsf{dCRH}}(1^n, h'_{s+1})$,
3. $(x_1, x_2, ..., x_s) \leftarrow \mathsf{Break\text{-}}s\mathsf{\text{-}SR}(1^n, h_1, h_2, ..., h_s; r_1)$,
4. $(x_{s+1}, x_{s+2}, ..., x_{2s}) \leftarrow \mathsf{Break\text{-}}s\mathsf{\text{-}SR}(1^n, h_1, h_2, ..., h_s; r_2)$,
5. Output $(x_1, ..., x_s, x_{s+1})$.

Due to the induction from $s = 2$ to $s = k-1$, we obtain the following statement.

Theorem 4 *For constant $k \geq 2$, assuming the existence of a secure k-rSRH such that each of functions compresses 2n bits to n bits, then there exists an (infinitely often) secure dCRH.*

Proof. We analyze the algorithm in terms of efficiency and correctness:

- Efficiency: Let t_1 be the upper bound of the running time of $\mathcal{A}_{\mathsf{dCRH}}$ and t_s be the upper bound of the running time of Break-s-SR. Since Break-$(s+1)$-SR runs Break-s-SR twice and $\mathcal{A}$ once, we have $t_{s+1} = 2t_s + t_1$ for each $s \geq 1$. By induction, we have $t_k = (2^k - 1)t_1$. Note that k is constant and t_1 is polynomial. Thus, t_k is also a polynomial.

- Correctness. The proof of correctness is similar to proofs in the last subsection, so we omit the details. Suppose Break-s-SR fails to output an s-restricted subset cover with probability at most $\epsilon_s(n)$. We have $\epsilon_{s+1}(n) \approx 2\epsilon_s$ (where we omit the probability of error made by $\mathcal{A}_{\mathsf{dCRH}}$ in each iteration). The failure probability of the final algorithm is upper bounded by $2^{k-1}\epsilon_2$, which is a negligible probability since k is constant.

$\square$

3.4 Separating k-rSRH from OWP

In this section, we show that there is no fully black-box construction of k-rSRH from OWP. This separation uses Simon's separating oracle [100]. Using this oracle, [6] proves the separation result of CRH from OWP and indistinguishability obfuscators. [14] proves the separation result of MCRH from OWP. We follow their work and show a similar result about k-rSRH.

Definition 19 *A fully black-box construction of k-rSRH from a one-way permutation consists of a probabilistic polynomial-time generation algorithm* Samp *and a probabilistic polynomial-time reduction algorithm $\mathcal{R}$.*

- ***Correctness:*** *Given an oracle of any permutation $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$, the algorithm $\mathsf{Samp}^f(1^n)$ outputs a tuple of k oracle-aided circuits $C^f = (C_1^f, .., C_k^f)$, where for each $i \in [k], C_i : \{0,1\}^n \to \{0,1\}^{l(n)}$ where $l(n) < n - \log k$.*

- ***Black-box Security:*** *Let $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$ be any permutation. For any (possibly non-uniform) adversary $\mathcal{A}$ and polynomial*

$p_{\mathcal{A}}(n)$ such that

$$\Pr_{\mathsf{Samp},\mathcal{A}}\left[\begin{array}{c|c} \forall i, x \neq x_i & (C_1^f, ..., C_k^f) \leftarrow \mathsf{Samp}^f(1^n) \\ C_i(x) = C_i(x_i) & (x, x_1, ..., x_k) \leftarrow \mathcal{A}^f(C_1, ..., C_k) \end{array}\right] \geq \frac{1}{p_{\mathcal{A}}(n)} \tag{3.35}$$

holds for infinitely many $n \in \mathbb{N}$, there is a polynomial-time algorithm $\mathcal{R}$ and a polynomial $p_{\mathcal{R}}(n)$ such that

$$\Pr_{y \leftarrow \{0,1\}^n, \mathcal{R}}[y = f_n(x) | x \leftarrow \mathcal{R}^{f,\mathcal{A}}(y)] \geq \frac{1}{p_{\mathcal{R}}(n)} \tag{3.36}$$

holds for infinitely many $n \in \mathbb{N}$.

We rule out fully black-box constructions of k-rSRH from OWP.

Theorem 5 *(Informal.) There is no fully black-box construction of a k-rSRH from a one-way permutation.*

We prove this statement in the following steps. First, in Section 3.4.1, we construct a separating oracle which consists of a random permutation oracle f and an oracle CoverFinder, which outputs a k-restricted subset cover for any $(C_1^f, ..., C_k^f)$ with high probability. CoverFinder oracle plays the role of the adversary $\mathcal{A}$ that breaks k-rSRH. Then, we prove that any polynomial-time algorithm given the access to $\Gamma = (f, \mathsf{CoverFinder})$ cannot output the preimage of y w.r.t. f with non-negligible probability. Note that if the algorithm is only given the access to f (without CoverFinder), this statement is obviously true because a random permutation is one-way with overwhelming probability [54]. To prove the stronger statement, we define a special event called y-**hit**. In Section 3.4.2, we prove that any polynomial-time algorithm cannot invert y without triggering this event. This uses the Reconstruction Paradigm in [55]. In Section 3.4.3, we prove that if there exists a reduction algorithm that can invert y with triggering y-**hit**, then there exists another algorithm that can do the same without triggering y-**hit**. Finally, in Section 3.4.4, we conclude that any polynomial-time algorithm cannot invert y with non-negligible probability by querying the separating oracle. It implies the impossibility of fully black-box constructions.

3.4.1 The Separating Oracle

In this subsection, we define a separating oracle Γ and show that it can break k-rSRH with high probability. The oracle is depicted as follows:

Separating Oracle Γ: The oracle Γ consists of two oracles $(f, \mathsf{CoverFinder}^f)$:

- The function $f = \{f_n\}$: For every n, f_n is a random permutations on n bits.
- The oracle $\mathsf{CoverFinder}^f(C_1, ..., C_k)$: Let $C_1^f, ..., C_k^f : \{0,1\}^n \to \{0,1\}^{l(n)}$ be k circuits given access to oracle f. Given the description of $C_1^f, ..., C_k^f$, $\mathsf{CoverFinder}^f(C_1, ..., C_k)$ tries to output a k-restricted subset cover for $(C_1^f, ..., C_k^f)$ as follows. First, it picks a random n-bit string w. Then, for each $i \in [k]$, it picks a random permutations π_i on n bits. After that, it independently computes the lexicographically smallest n-bit string w_i such that $C_i^f(w) = C_i^f(\pi_i(w_i))$. Finally $\mathsf{CoverFinder}^f$ outputs $(w, \pi_1(w_1), ..., \pi_k(w_k))$. Note that $\mathsf{CoverFinder}^f$ is expected to be exponential-time.

In the following, we say $\mathsf{CoverFinder}$ instead of $\mathsf{CoverFinder}^f$ for simplicity. We show that this oracle outputs a k-restricted subset cover for any k-tuple of circuits with at least polynomial probability for infinitely many n.

Lemma 11 *For any permutation f_n and any oracle-aided circuits $C_1^f, ..., C_k^f : \{0,1\}^n \to \{0,1\}^{l(n)}$ where $n > l(n) - \log k$, there exists a polynomial $p(n)$ such that*

$$\Pr_{\mathit{CoverFinder}}\left[\begin{array}{l|l} \forall i \in [k], x \neq x_i & \\ C_i^f(x) = C_i^f(x_i) & \end{array} (x, x_1, ..., x_k) \leftarrow \mathit{CoverFinder}(1^n, C_1^f, ..., C_k^f)\right] \geq \frac{1}{p(n)} \tag{3.37}$$

for infinitely many n.

Proof.

We first show that for $(x, x_1, ..., x_k) \leftarrow \mathsf{CoverFinder}^f(1^n, C_1^f, ..., C_k^f)$, x has a collision w.r.t. each C_i^f with constant probability.

For $i \in [k]$, let X_{C_i} be the set of x that does not have a collision w.r.t. C_i. We observe that $|X_{C_i}| < 2^{l(n)}$, otherwise the range size of C_i must be larger than $2^{l(n)}$.

Since x is randomly chosen from $\{0,1\}^n$, it holds that

$$\Pr_{\mathsf{CoverFinder}}[x \in \bigcup_{i\in[k]} X_{C_i}] \le \frac{k\cdot 2^{l(n)}}{2^n} = \frac{1}{2^{n-l(n)-\log k}} < \frac{1}{2}, \tag{3.38}$$

where the second inequality holds since $n > l(n) - \log k$.

Next, we show that $(x, x_1, ..., x_k)$ is a k-restricted subset cover w.r.t. $(C_1, ..., C_k)$ with constant probability. Suppose $x \notin \bigcup_{i\in[k]} X_{C_i}$ (with probability is more than $1/2$). Due to the stategy of CoverFinder, it holds that $C_i(x) = C_i(x_i)$ for each $i \in [k]$, but it is possible that $x = x_i$ for some i. Note again that x is randomly chosen from $\{0,1\}^n$ and π_1 is a random permutation on $\{0,1\}^n$. Since $x \notin X_{C_i}$, there are at least two $x' \in \{0,1\}^n$ such that $C_i(x) = C_i(x_i)$ and x_i is one of them with uniform distribution. Thus, the probability that $x = x_i$ is at most $\frac{1}{2}$. We have

$$\Pr_{\mathsf{CoverFinder}}\left[\begin{matrix}\forall i\in[k], x\neq x_i\\ C_i(x) = C_i(x_i)\end{matrix}\middle| x\notin \bigcup_{i\in[k]} X_{C_i}\right] \ge \frac{1}{2^k}. \tag{3.39}$$

Note that k is constant. Due to inequality (3.38) and (3.39), the probability that $(x, x_1, ..., x_k)$ is a k-restricted subset cover is more than $\frac{1}{2^{k+1}}$. This completes the proof. □

3.4.2 From Inversing to Compressing

From this subsection, we show that for every polynomial-time algorithm $\mathcal{A}$ given the access to the oracle Γ, there exists a negligible function **negl** such that

$$\Pr_{\substack{y\leftarrow\{0,1\}^n\\ f_n,\mathsf{CoverFinder},\mathcal{A}}}[y = f_n(x) | x \leftarrow \mathcal{A}^{\Gamma}(1^n, y)] \le \mathbf{negl}(n) \tag{3.40}$$

for large enough n.

Let $\mathcal{A}$-**win** be the above event and we need to show that $\Pr[\mathcal{A}\text{-}\mathbf{win}] \le \mathbf{negl}(n)$. In this subsection, we show a weaker statement. We define a special event called y-**hit** and prove that without triggering y-**hit**, $\mathcal{A}$-**win** only occurs with negligible probability.

Definition 20 *In the process of running $\mathcal{A}^{\Gamma}(y)$, when $\mathcal{A}$ makes a query $(C_1, ..., C_k)$ to* CoverFinder *and obtains $(x, x_1, ..., x_k)$, we say that this query triggers the event y-**hit** if in evaluating $C_i^f(x)$ and $C_i^f(x_i)$ for some $i \in [k]$, it queries an x to f_n such that $y = f_n(x)$. If there exists such a query to* CoverFinder *triggering y-**hit**, we simply say that y-**hit** occurs.*

Lemma 12 *For any polynomial-time adversary $\mathcal{A}$, it holds that*

$$\Pr_{\substack{y \leftarrow \{0,1\}^n, f_n, \mathcal{A} \\ \textsf{CoverFinder}}} [\mathcal{A}\text{-}\boldsymbol{win} \wedge \overline{y\text{-}\boldsymbol{hit}}] \leq 2^{-n/7} \tag{3.41}$$

for large enough n.

Proof. We prove a stronger statement, fixing the randomness of CoverFinder and $\mathcal{A}$. Since the randomness of $\mathcal{A}$ is fixed, we consider a deterministic adversary $\mathcal{A}$. Let Z be the truth table of f_n. We show that given an adversary $\mathcal{A}$ that inverts y without triggering y-**hit**, it is possible to compress Z with a more efficient encoding (X_f, Y_f, Z_f). Here, Z_f is the part of Z, and X_f, Y_f are respectively the set of preimages and images which are not covered in Z_f. Thus, the truth table between X_f and Y_f is not recorded in (X_f, Y_f, Z_f). We introduce a reconstruction algorithm to reconstruct the whole truth table Z from (X_f, Y_f, Z_f). However, since f_n is a random permutation, the truth table cannot be compressed. This yields a contradiction.

Next, we show how to pick (X_f, Y_f, Z_f). Let $I_f \subseteq \{0,1\}^n$ be the set of $y \in Y$ that $\mathcal{A}$ can invert without triggering y-**hit**.

$$I_f = \{y | \mathcal{A}\text{-}\mathbf{win} \wedge \overline{y\text{-}\mathbf{hit}}\}. \tag{3.42}$$

Now we pick Y_f as follows: (1) pick the lexicographically smallest $y^* \in I_f$, (2) run $\mathcal{A}(y^*)$, (3) every time $\mathcal{A}$ makes f_n-query x and obtains an image $y = f_n(x)$, remove this y from I_f, (4) every time $\mathcal{A}$ makes CoverFinder-query $(C_1, ..., C_k)$ and obtains $(x, x_1, ..., x_k)$, evaluate $C_i(x)$ and $C_i(x_i)$ for each $i \in [k]$, and removes from I_f all the outputs of f_n-queries during the evaluation, (5) store this y^* in a set Y_f, and (6) go to step (1).

Without loss of generality, we suppose that if $x \leftarrow \mathcal{A}(y)$, $\mathcal{A}$ has queried $f_n(x)$ in the execution. Thus, for each y^* picked in step (1), it is removed from I_f in step (3).

Lemma 13 *Let $q_{\mathcal{A}}$ be the upper bound of the number of queries made by $\mathcal{A}$ and q_C be the upper bound of the number of f-queries required in evaluating C_i^f. Let $q = \max\{q_{\mathcal{A}}, q_C\}$. It holds that*

$$|I_f| \leq 3kq^2|Y_f|. \tag{3.43}$$

Proof. Suppose a query to CoverFinder$(C_1, ..., C_k)$ is replied by $(x, x_1, ..., x_k)$. Note that evaluating all the $C_i(x)$ and $C_i(x_i)$ makes at most $2kq_C$ queries to f_n. For every y, $\mathcal{A}(y)$ makes at most $q_{\mathcal{A}}$ queries to f_n and also at most $q_{\mathcal{A}}$ queries to CoverFinder. When we pick Y_f, for each $y \in Y_f$, we remove at most $q_{\mathcal{A}}$ elements in step (3) and at most $q_{\mathcal{A}} \cdot 2kq_C$ elements in step (4). Thus, in each loop, we remove at most

$$q_{\mathcal{A}} \cdot 2kq_C + q_{\mathcal{A}} \leq 3kq^2 \tag{3.44}$$

number of elements from I_f and then add one element to Y_f. This implies the lemma. □

Let $X_f = f_n^{-1}(Y_f)$. Let Z_f be the partial truth table that stores all the maps of f_n except those from X_f to Y_f. Next, we show that (X_f, Y_f, Z_f) can encode the whole truth table of f_n. We introduce a reconstruction algorithm that outputs the truth table Z of f_n taking as input (X_f, Y_f, Z_f).

1. While $Y_f \neq \emptyset$
   (a) Pick the lexicographically smallest $y \in Y_f$
   (b) Run $\mathcal{A}(y)$ as follows:
      - When $\mathcal{A}$ queries x to f_n, if $x \in Z_f$, answers $Z_f(x)$. Else, let $Z_f = Z_f \cup \{(x, y)\}$. Remove y from Y_f and go to step 1.
      - When $\mathcal{A}$ queries $(C_1, ..., C_k)$ to CoverFinder, do as follows:
         i. Obtain $(w, \pi_1, ..., \pi_k)$ from the random tape of CoverFinder.

ii. Compute $C_i^f(w)$ for each $i \in [k]$. When it queries x to f_n, answer $Z_f(x)$.

iii. For every j from 0^n to 1^n, evaluate $C_1^f(\pi_1(j))$. During the evaluation, when it queries x to f_n, answer $Z_n(x_n)$ if it is stored. Otherwise, pick the next j. If all f_n queries are answered and $C_1^f(\pi_1(j)) = C_1^f(w)$, then let $w_1 = \pi_1(j)$.

iv. Do the same on π_i for each $i \in [k]$ and obtains w_i.

v. return $(w, w_1, w_2, ..., w_k)$.

(c) When $\mathcal{A}$ outputs x, let $Z_f = Z_f \cup \{(x, y)\}$ and remove y from Y_f. Go to step 1.

2. Output Z_f.

Lemma 14 *The whole truth table for f_n can be encoded by (X_f, Y_f, Z_f).*

Proof. We claim that the above reconstruction can build the whole truth table Z. Since Y_f is the set of images that is not stored in Z_f but needs to be stored in Z. We fill in the "blanks" of Z by starting from the smallest element of Y_f. If the blanks are filled in correctly, the reconstruction outputs the real Z. In the reconstruction algorithm, we use $\mathcal{A}$ to fill in these blanks. Since $Y_f \subseteq I_f$, which means for every $y \in Y_f$, $\mathcal{A}(y)$ can correctly output the preimage of y w.r.t. f_n, we only need to guarantee that $\mathcal{A}$ gets the same responses from f_n and CoverFinder as the real one.

Next, we show that in step 1(b) of the reconstruction algorithm, it replies to the queries of $\mathcal{A}$ perfectly as the true oracles.

- The reconstruction algorithm replies CoverFinder-queries correctly.

  Note that the random tape of CoverFinder is fixed, π_i and w is identical to that of the real CoverFinder. The only difference is the stategy computing $C_i^f(w)$ in step (ii) and $C_i^f(\pi_i(j))$ in step (iii). In detail, in the execution of the real CoverFinder, $C_i^f(w)$ and $C_i^f(\pi_i(j))$ are evaluated with access to the real truth table of f, while for the reconstruction algorithm, they are evaluated with Z_f. However, we claim that this will not cause a different response for CoverFinder-queries.

First, we claim that $C_i^f(w)$ is computed correctly in step (ii). Suppose $y \in Y_f$ and $\mathcal{A}(y)$ queries CollFinder(C) obtaining $(w, w_1, ..., w_k)$. For each f_n-query x in computing $C_i^f(w)$, $f_n(x)$ is removed from I_f. That is, $f_n(x)$ is not in Y_f, and thus $(x, f_n(y))$ is covered in Z_f. Hence, Z_f can reply all the queries in computing $C_i^f(w)$.

Next, we assume that computing $C_i^f(\pi_i(j))$ is different with that of $C_i^f(\pi_i(j))$ for some j in step (iii), and resulting in a different response for a CoverFinder-query. It implies that in computing $C_i^f(\pi_i(j))$, it queries an x' to f_n, but $x' \in X_f$ and thus the map of x' is not stored in Z_f. This causes a difference between the real CoverFinder and our reconstruction algorithm.

However, we claim that this event never occurs. When we construct Y_f from I_f, we remove all y from I_f queried in computing $C_i^f(w_i)$. Thus, for all $y \in Y_f$, if $\mathcal{A}(y)$ queries to CoverFinder and obtains a $(w, w_1, ..., w_k)$, the f_n-queries in evaluating $C_i^f(w_i)$ are not contained in X_f. This yields a contradiction that computing $C_i^f(w_i)$ triggers a query x' to f_n such that $x' \in X_f$.

- The reconstruction algorithm replies f_n-queries correctly.

  Assume that the reconstruction algorithm cannot reply f_n correctly for some x. It implies that given a $y \in Y_f$ and the real CoverFinder, $\mathcal{A}^{f_n,\mathsf{CoverFinder}}(y)$ queries an x to f_n, which is not stored in Z_f. There are two cases:

  – $f_n(x) = y$. In this case the reconstruction algorithm will store (x, y) in the truth table and terminate the loop.

  – $f_n(x) \neq y$. This event never happens. Assume it does, it implies that $\mathcal{A}(y)$ queries x to f_n but $x \in X_f$, and thus $f_n(x) \in Y_f$. Note that $y \in Y_f \subseteq I_f$. Due to the description of Y_f, all the f_n-queries during the execution of $\mathcal{A}(y)$ have been removed from Y_f. Thus, $\mathcal{A}(y)$ never queries an x to f_n that $f_n(x) \in Y_f$.

As is discussed above, the reconstruction algorithm correctly replies to the queries from $\mathcal{A}(y)$ for any $y \in Y_f$. The reconstruction algorithm identically builds the real truth table Z. $\square$

Let $\epsilon = 2^{-n/3}$. We say f_n is "ϵ-good" if $|I_f| \geq \epsilon 2^n$. It implies that for any ϵ-good f_n, the probability of $\mathcal{A}$**-wins** $\wedge$ $\overline{y\textbf{-hit}}$ is more than ϵ over the choice of $y \in \{0,1\}$. Let S_ϵ be the set of f_n such that f_n is ϵ-good. We reconsider the probability $\Pr_{f_n,y}[\mathcal{A}\textbf{-wins} \wedge \overline{y\textbf{-hit}}]$.

Since f_n is a random permutation, we consider the following cases:

- **Case 1**: f_n is ϵ-good.

  In this case, we have $|I_f| \geq \epsilon 2^n = 2^{2n/3}$. Since $|I_f| \leq 3kq^2|Y_f|$, we have

$$|Y_f| \geq \frac{|I_f|}{3kq^2} \geq \frac{2^{2n/3}}{3kq^2}. \tag{3.45}$$

  Note that f_n can be encoded by (X_f, Y_f, Z_f). Here, $|Z_f|$ is encoded by $\log((2^n - |Y_f|)!)$ bits. We have

$$\Pr_{f_n}[f_n \in S_\epsilon] \leq \frac{\binom{2^n}{|Y_f|}^2 \cdot (2^n - |Y_f|)!}{(2^n)!} = \frac{\binom{2^n}{|Y_f|}}{|Y_f|!} \leq (\frac{e2^n}{|Y_f|})^{|Y_f|}(\frac{e}{|Y_f|})^{|Y_f|} \leq (\frac{e^2 2^n}{|Y_f|^2})^{|Y_f|}. \tag{3.46}$$

  Note that q is a polynomial of n. Since $|Y_f| \geq 2^{2n/3}/5q^2$, for large enough n, we have

$$(\frac{2^n e^2}{|Y_f|^2})^{|Y_f|} \leq (\frac{9e^2k^2q^4 2^n}{2^{4n/3}})^{|Y_f|} = (\frac{9e^2k^2q^4}{2^{n/3}})^{|Y_f|} \leq 2^{-n}. \tag{3.47}$$

  Thus, for every random tape of CoverFinder and $\mathcal{A}$, we have

$$\Pr_{f_n,y}[\mathcal{A}\textbf{-wins} \wedge \overline{y\textbf{-hit}} \wedge f_n \in S_\epsilon] \leq \Pr_{f_n}[f_n \in S_\epsilon] \leq 2^{-n} \tag{3.48}$$

  for large enough n.

- **Case 2**: f_n is not ϵ-good.

  In this case we have $|I_f| < \epsilon 2^n$, and thus for $y \leftarrow \{0,1\}^n$, the probability that $\mathcal{A}(y)$ inverts y is at most $\epsilon = 2^{n/3}$. We have

$$\Pr_{f_n,y}[\mathcal{A}\textbf{-wins} \wedge \overline{y\textbf{-hit}} \wedge f_n \notin S_\epsilon] \leq \Pr_{y}[\mathcal{A}\textbf{-wins} \wedge \overline{y\textbf{-hit}} | f_n \notin S_\epsilon] \leq 2^{-n/3}. \tag{3.49}$$

From inequality (3.48) and (3.49), we have

$$\Pr_{f_n,y}[\mathcal{A}\textbf{-win} \wedge \overline{y\textbf{-hit}}] \leq 2^{-n} + 2^{-n/3} \leq 2^{-n/2}, \tag{3.50}$$

for large enough n, which completes the proof. □

3.4.3 From y-hit to $\overline{y\text{-hit}}$

In this subsection, we show that if there exists an adversary $\mathcal{A}^{f,\mathsf{CoverFinder}}$ that can invert y with triggering y-**hit**, then we can construct an another algorithm to invert y without triggering y-**hit**.

Lemma 15 *For any $y \in \{0,1\}^n$ and any permutation f_n, suppose there exist a polynomial $p(n)$ and a polynomial-time algorithm $\mathcal{A}$ such that*

$$\Pr_{\mathsf{CoverFinder},\mathcal{A}}[\mathcal{A}\textbf{-win} \wedge y\textbf{-hit}] \geq \frac{1}{p(n)} \tag{3.51}$$

for infinitely many n. Then, there exists a polynomial-time algorithm $\mathcal{B}$ such that

$$\Pr_{\mathsf{CoverFinder},\mathcal{B}}[\mathcal{B}\textbf{-win} \wedge \overline{y\textbf{-hit}}] \geq \frac{1}{2p(n)} \tag{3.52}$$

for infinitely many n.

Proof. The intuition of constucting $\mathcal{B}$ is as follows. Suppose $\mathcal{A}$ is a polynomial-time algorithm depicted above. $\mathcal{B}$ mainly follow the steps of $\mathcal{A}$. The difference is that every time $\mathcal{A}(y)$ queries to CoverFinder (that is, every time $\mathcal{A}$ has chances to trigger the event y-**hit**), $\mathcal{B}$ tries to inverse y before the query to CoverFinder by additional operations (without querying CoverFinder). Suppose that $\mathcal{A}(y)$ triggers y-**hit** for the first time in the ith query to CoverFinder, and that $\mathcal{B}$ succeeds in inversing y by additional operations before this query. This implies that $\mathcal{B}$ inverses y without triggering y-**hit**.

Now we show how to inverse y by additional operations without querying CoverFinder. Let us review the strategy of CoverFinder. Taking as input $(C_1^f, ..., C_k^f)$, it picks a random $w \in \{0,1\}^n$ and k random permutations π_i on $\{0,1\}^n$. For each

$i \in [k]$, it picks the lexicographically smallest $j_i \in \{0,1\}^n$ such that $C_i(\pi_i(j_i))$ is equal to $C_i(\pi_i(w))$. Note that here the behavior of $\mathsf{CoverFinder}$ is independent to y. If the computation of $C^f(\cdot)$ requires a f_n-query x such that $f_n(x) = y$, we say $C^f(\cdot)$ "hits" y. Since w and π_i are uniformly random, the event that $C_i^f(w)$ hits y and the event that $C_i^f(w_i)$ hits y are of the same probability for each i, where the probability is taken over the choice of y and randomness of $\mathsf{CoverFinder}$.

We add following operations to $\mathcal{A}$. Before $\mathcal{A}$ queries $(w, w_1, ..., w_k) \leftarrow \mathsf{CoverFinder}(C_1, ..., C_k)$, it picks random $w^* \in \{0,1\}^n$ and then computes $C_i^f(w^*)$ for each i. This is the same as the behaviors of $\mathsf{CoverFinder}(C_1, ..., C_k)$. This implies that for any f, the probability that $C_i^f(w^*)$ hits y is equal to the probability that $C_i^f(w)$ hits y, and thus is equal to the half of the probability that $C_i^f(w)$ and $C_i^f(w_i)$ hit y. To add up the case of $i \in [k]$, the probability that our additional operations hit y is a half of the probability that $\mathsf{CoverFinder}(C_1, ..., C_k)$ triggers y-**hit**, where the probability is taken over the choice of y and the randomness of $\mathsf{CoverFinder}$ and $\mathcal{B}$.

Formally, taking as input $y \in \{0,1\}^n$, the algorithm $\mathcal{B}$ follows the steps of $\mathcal{A}(y)$. Additionally, when $\mathcal{A}$ makes a query to oracles f and $\mathsf{CoverFinder}$, $\mathcal{B}$ behaves as follows:

- When it queries x to f, $\mathcal{B}$ also queries x to f.
- When $\mathcal{A}$ queries $(C_1, ..., C_k)$ to $\mathsf{CoverFinder}$, $\mathcal{B}$ randomly chooses $w^* \in \{0,1\}^n$ and evaluates $C_i^f(w^*)$ for each $i \in [k]$. In this process, if it ever queries x' to f_n where $y = f_n(x)$, $\mathcal{B}$ terminates and outputs x. If it does not terminate, it queries $(C_1, ..., C_k)$ to $\mathsf{CoverFinder}$.

Next, we prove the inequality (3.52). Let q be the upper bound of the number of queries to $\mathsf{CoverFinder}$ made by $\mathcal{A}$, and let $C^{(1)}, ..., C^{(q)}$ be the queries to $\mathsf{CoverFinder}$ made by $\mathcal{A}(y)$ (each $C^{(i)}$ is a tuple of k circuits $(C_1^{(i)}, ..., C_k^{(i)})$). There are at most q "chances" for $\mathcal{B}$ to terminate. We define two events as follows:

- $\mathbf{Jump}_i$: Before querying $\mathsf{CoverFinder}(C^{(i)})$, $\mathcal{B}$ chooses w^* such that $C_j^{(i)}(w^*)$ hits y for some $j \in [k]$, which leads to termination.
- $\mathbf{Fail}_i$: The query $C^{(i)}$ to $\mathsf{CoverFinder}$ triggers y-**hit**.

We observe that the event $\mathcal{B}$-**win**$\wedge y$-**hit** occurs if and only if $\mathbf{Jump}_i$ happens for some $i \in [q]$ and $\mathbf{Fail}_j$ never occurs for any $j < i$. That is,

$$\Pr[\mathcal{B}\text{-}\mathbf{wins} \wedge \overline{y\text{-}\mathbf{hit}}] = \sum_{i\in[q]} \Pr[\mathbf{Jump}_i \wedge \bigwedge_{j<i} \overline{\mathbf{Fail}_j}]. \tag{3.53}$$

In addition, we observe that for any f_n, $\mathbf{Jump}_i$ happens with the half of probability that $\mathbf{Fail}_i$ happens. That is,

$$\Pr_{\mathsf{CoverFinder},\mathcal{B}}[\mathbf{Jump}_i | \bigwedge_{j<i} \overline{\mathbf{Fail}_j}] = \frac{1}{2}\Pr[\mathbf{Fail}_i | \bigwedge_{j<i} \overline{\mathbf{Fail}_j}]. \tag{3.54}$$

From equality (3.53) and (3.54), we have

$$\Pr_{\mathsf{CoverFinder},\mathcal{B}}[\mathcal{B}\text{-}\mathbf{wins} \wedge \overline{y\text{-}\mathbf{hit}}] = \frac{1}{2}\sum_{i\in[q]} \Pr[\mathbf{Fail}_i \wedge \bigwedge_{j<i} \overline{\mathbf{Fail}_j}] = \frac{1}{2}\Pr[\bigvee_{i\in[q]} \mathbf{Fail}_i]. \tag{3.55}$$

Note that $\bigvee_{i\in[q]} \mathbf{Fail}_i$ implies y-**hit** and that $\Pr[y\text{-}\mathbf{hit}] \geq \Pr[\mathcal{A}\text{-}\mathbf{wins} \wedge y\text{-}\mathbf{hit}] \geq \frac{1}{p(n)}$. We have $\Pr[\mathcal{B}\text{-}\mathbf{wins} \wedge \overline{y\text{-}\mathbf{hit}}] \geq \frac{1}{2p(n)}$. This completes the proof.

□

3.4.4 Main Result

In this section, we give the separation result using the lemma in the last three subsections.

Theorem 6 *For any constant $k \geq 2$ and r, there is no fully black-box construction from OWP of k-SRH compressing n bits to $l(n) < n - \log k$ bits.*

Proof.

Suppose there exists such a fully black-box construction $(\mathsf{Samp}, \mathcal{R})$. Let $\Gamma = (f, \mathsf{CoverFinder})$ be the oracle depicted in section 3.4.1. Due to Lemma 11, for any permutation f_n, there exists a polynomial $p(n)$ such that

$$\Pr_{\mathsf{Samp},\mathsf{CoverFinder}}\left[\begin{array}{c|l} \forall i, x \neq x_i & (C_1^f, ..., C_k^f) \leftarrow \mathsf{Samp}^f(1^n) \\ C_i(x) = C_i(x_i) & (x, x_1, ..., x_k) \leftarrow \mathsf{CoverFinder}^f(1^n, C_1, ..., C_k) \end{array}\right] \geq \frac{1}{p(n)}, \tag{3.56}$$

for infinitely many $n \in \mathbb{N}$.

Since $(\mathsf{Samp}, \mathcal{R})$ is a fully black-box construction, given access to any oracle $\Gamma = (f, \mathsf{CoverFinder})$, there exists a polynomial $p_{\mathcal{R}}$ such that

$$\Pr_{y,\mathcal{R}}[y = f_n(x) | x \leftarrow \mathcal{R}^{\Gamma}(y)] \geq \frac{1}{p_{\mathcal{R}}(n)} \tag{3.57}$$

for infinitely many $n \in \mathbb{N}$ and thus

$$\Pr_{\substack{f_n,y,\mathcal{R}\\ \mathsf{CoverFinder}}}[y = f_n(x) | x \leftarrow \mathcal{R}^{\Gamma}(y)] \geq \frac{1}{p_{\mathcal{R}}(n)}. \tag{3.58}$$

That is,

$$\Pr[\mathcal{R}\textbf{-win} \wedge y\textbf{-hit}] + \Pr[\mathcal{R}\textbf{-win} \wedge \overline{y\textbf{-hit}}] \geq \frac{1}{p_{\mathcal{R}}(n)}. \tag{3.59}$$

Due to Lemma 12, $\Pr[\mathcal{R}\textbf{-win} \wedge \overline{y\textbf{-hit}}] \leq 2^{-n/7}$ for large enough n. Thus, there exists a polynomial $p'_{\mathcal{R}}(n)$ such that $\Pr[\mathcal{R}\textbf{-win} \wedge y\textbf{-hit}] \geq \frac{1}{p'_{\mathcal{R}}(n)}$. From an average argument, we have

$$\Pr_{f_n,y}[\Pr_{\mathsf{CoverFinder},\mathcal{R}}[\mathcal{R}\textbf{-win} \wedge y\textbf{-hit}] \geq \frac{1}{2p'_{\mathcal{R}}(n)}] \geq \frac{1}{2p'_{\mathcal{R}}(n)}. \tag{3.60}$$

Let $T = \{(f_n, y) | \Pr_{\mathsf{CoverFinder},\mathcal{R}}[\mathcal{R}\textbf{-win} \wedge y\textbf{-hit}] \geq \frac{1}{2p'_{\mathcal{R}}(n)}\}$. Due to Lemma 15, for any $(f_n, y) \in T$, there exists a polynomial-time machine $\widetilde{\mathcal{R}}$ such that

$$\Pr_{\mathsf{CoverFinder},\widetilde{\mathcal{R}}}[\widetilde{\mathcal{R}}\textbf{-win} \wedge \overline{y\textbf{-hit}}] \geq \frac{1}{4p'_{\mathcal{R}}(n)} \tag{3.61}$$

for infinitely many n. Therefore,

$$\begin{aligned}
\Pr_{\substack{f_n,y,\widetilde{\mathcal{R}}\\ \mathsf{CoverFinder}}}[\widetilde{\mathcal{R}}\textbf{-win} \wedge \overline{y\textbf{-hit}}] &\geq \Pr_{\substack{f_n,y,\widetilde{\mathcal{R}}\\ \mathsf{CoverFinder}}}[\widetilde{\mathcal{R}}\textbf{-win} \wedge \overline{y\textbf{-hit}} \wedge (f_n, y) \in T] \\
&= \Pr_{\mathsf{CoverFinder},\widetilde{\mathcal{R}}}[\widetilde{\mathcal{R}}\textbf{-win} \wedge \overline{y\textbf{-hit}} | (f_n, y) \in T] \cdot \Pr_{f_n,y}[(f_n, y) \in T] \\
&\geq \frac{1}{4p'_{\mathcal{R}}(n)} \cdot \frac{1}{2p'_{\mathcal{R}}(n)} = \frac{1}{8p'_{\mathcal{R}}(n)^2}
\end{aligned}$$

for infinitely many n. This contradicts to Lemma 12 showing that this probability is negligible. □

3.5 From k-rSRH to General (r, k)-SRH

In the last sections, we discuss the attacks and properties of k-rSRH. In this section, we extend the results to general (r, k)-SRH.

Since k-restricted subset resilience is a weaker assumption than (k, k)-subset resilience, our results can be smoothly extended to (k, k)-subset resilience.

- In Section 3.2, we propose a quantum algorithm finding k-restricted subset covers that are also (k, k)-subset covers.
- In Section 3.3, we prove that the existence of k-rSRH implies the existence of dCRH. Note that the existence of (k, k)-SRH immediately implies the existence of k-rSRH, so it further implies that of dCRH.
- In Section 3.4, we prove the fully black-box separation of k-rSRH from OWP, which also implies the same result of (k, k)-SRH from OWP.

On the other hand, it is non-trivial to extend the result from (k, k)-SRH to (r, k)-SRH, since (k, k)-SRH is a stronger assumption than (r, k)-SRH for $r < k$. It is natural that when we turn to (r, k)-SRH, there will be some additional constraint conditions upon our results.

Now we explain how our results of (k, k)-SRH is generalized to (r, k)-SRH. We give a simple example on generalizing our first result to finding an (r, k)-restricted subset cover. Given quantum access to $H = (h_1, ..., h_4)$ where $h_i : X \to Y$, we aim to find a (2,4)-restricted subset cover (x, x_1, x_2) for H. It implies that for each $i \in [4]$, $h_i(x) \in \{h_i(x_1), h_i(x_2)\}$ holds. Denote $h_{1||2}(x) \triangleq h_1(x)||h_2(x)$ and $h_{3||4}(x) \triangleq h_3(x)||h_4(x)$. Here $h_{1||2}$ and $h_{3||4}$ map from X to $Y' \triangleq Y^2$. Suppose $X \geq 3|Y'| = 3|Y|^2$. We can run the algorithm in Section 3.2 on $(h_{1||2}, h_{3||4})$ and obtain (x, x_1, x_2), where $h_{1||2}(x) = h_{1||2}(x_1)$ and $h_{3||4}(x) = h_{3||4}(x_2)$. Thus, it is a $(2, 4)$-restricted subset cover (and also a (2,4)-subset cover) w.r.t. $(h_1, ..., h_4)$. The time complexity is $O(|Y'|^{3/7}) = O(N^{6/7})$.

Formally, we generalize our results to (r, k)-SRH (and also to (r, k)-rSRH) as follows:

Theorem 7 *(Extended Theorem 2) For constant $k \geq 2$ and r, denote $\omega = \lceil k/r \rceil$. Let $H = (h_1, ..., h_k)$ be a tuple of functions where $h_i : X \to Y$ and $|X| \geq (r+1)|Y|^\omega = (r+1)N^\omega$. There exists an algorithm finding an (r,k)-subset cover for H with overwhelming probability using $O(N^{\frac{\omega}{2}(1-\frac{1}{2^{r+1}-1})})$ quantum queries to H.*

Proof.

To prove this statement, we only need to consider the case where $k = \omega r$. If $k < \omega r \triangleq k'$, we denote $h_{k+1}, ..., h_{k'}$ by functions mapping X to a constant element of Y. Then, we obtain a tuple of k' functions. Finding a subset cover for this tuple implies finding a subset cover for $(h_1, ..., h_k)$.

Next we assume that $k = \omega r$. For each $i \in [r]$, denote $h_i^*(x) \triangleq h_{(i-1)\omega+1}^*(x)||...||h_{i\omega}^*(x)$. Thus, $H^* = (h_1^*, ..., h_r^*)$ is a tuple of functions where $h_i^* : X \to Y'$ and $|Y'| = |Y|^\omega = N^\omega$. Note that $|X| \geq (r+1)|Y'|$ due to the conditions on each h_i. We can run the algorithm in Theorem 2 on H^*, and obtain $(x, x_1, ..., x_r)$ as a r-restricted subset cover of H^* with overwhelming probability. The output is an (r,k)-restricted subset cover of H (and also an (r,k)-subset cover). Due to Theorem 3, the number of required quantum queries to H is $O(|Y'|^{\frac{1}{2}(1-\frac{1}{2^{r+1}-1})}) = O(N^{\frac{\omega}{2}(1-\frac{1}{2^{r+1}-1})})$.

□

Theorem 8 *(Extended Theorem 4) For constant $k \geq 2$ and r, denote $\omega = \lceil k/r \rceil$. Assuming the existence of a secure (r,k)-SRH such that each of functions compresses $2\omega n$ bits to n bits, then there exists an (infinitely often) secure dCRH.*

Proof. Similar to the last theorem, we also only need to consider the case that $k = \omega r$. In the case that $k < \omega r \triangleq k'$, the existence of (r,k)-SRH implies the existence of (r,k')-SRH. Then we can turn to prove that the existence of (r,k')-rSRH implies the existence of dCRH, which implies the original statement.

Next we assume that $k = \omega r$ and there exists an (r,k)-SRH $\mathcal{H} = \{h : \{0,1\}^{2\omega n} \to \{0,1\}^n\}$ w.r.t. k-sampling algorithm Samp_k. Denote another function family ensemble by $\mathcal{H}^* = \{h^* : \{0,1\}^{2\omega n} \to \{0,1\}^{\omega n}\}$ and its k-sampling algorithm by

Samp_k^*. Samp_k^* runs as follows. It samples $(h_1, ..., h_k) \leftarrow \mathsf{Samp}_k$, and let $h_i^* \triangleq h_{(i-1)\omega+1}||...||h_{i\omega}$ for each $i \in [r]$. Then, it outputs $(h_1^*, ..., h_r^*)$. We observe that an r-restricted subset cover of $(h_1^*, ..., h_r^*) \leftarrow \mathsf{Samp}_k^*$ implies an (r, k)-subset cover of $(h_1, ..., h_k) \leftarrow \mathsf{Samp}_k$. Thus, $\mathcal{H}^*$ is an r-rSRH w.r.t. Samp_k^* since $\mathcal{H}$ is (r, k)-rSRH w.r.t. Samp_k^*. Furthermore, the existence of $\mathcal{H}^*$ (mapping $2\omega n$ bits to ωn bits) implies the existence of a dCRH due to Theorem 4. This completes the proof.

□

Theorem 9 *(Extended Theorem 6) For constant $k \geq 2$ and r, let $\omega = \lceil k/r \rceil$. There is no fully black-box construction from OWP to (r, k)-SRH compressing n bits to $l(n) < (n - \log r)/\omega$ bits.*

Proof. Again, we consider the case that $k = \omega r$. In the proof of Theorem 8, we show that an (r, k)-rSRH compressing n bits to $l(n)$ bits immediately implies a k-rSRH compressing n bits to $\omega l(n) \triangleq l'(n)$ bits. Note that $l'(n) < n - \log r$. Due to Theorem 6, there is no fully black-box construction of such a k-rSRH from OWP. It implies that there is no fully black-box construction of (r, k)-rSRH compressing n bits to $l(n)$ bits from OWP. □

Remark 5 *Note that cover-free families (CFFs) are information-theoretic version of SRH, implying the possibility to construct a perfect SRH without* any *assumption (such as OWP). For instance, by implementing the CFF in [48], we can constuct an (r, k)-SRH mapping $\{0, 1\}^n$ to $\{0, 1\}^{l(n)}$ where $2^{l(n)} \leq 16r^2 n$ and $k = 2^{l(n)}/4r$. However, it does not contradicts to our result, since the parameter k in this instance is far from a constant. We stress that our Theorem 8 and 9 only work on their constraint conditions.*

3.6 Conclusion and Open questions

In this work, we present three results on the studies of subset resilience. The first result is a generic quantum attack against subset resilience. This implies an upper bound of the security of subset resilience. The second result is the relation with dCRH. It implies that the power of assuming SRH is stronger than dCRH. The third result is the fully black-box separation from one-way permutations, which rules out the possibility of constructing SRH from one-way permutations in a fully black-box manner.

Note that there is a constraint condition in each statement. (For example, we only rule out the possibility of constructing an (r, k)-SRH from OWP in the case that $l(n) < (n - \log r)/\omega$ where $\omega = \lceil k/r \rceil$.) Indeed, we do not know whether the bounds of the parameters and the complexity of the attacks are optimal, and we cannot give a counterexample when the parameter is out of the bound. It leaves an open question of whether we can improve the results presented in this chapter.

Target subset resilience is a weaker variant of subset resilience. It is first proposed as a security notion needed in RMA security of HORS[94]. Although the CMA security of SPHINCS[15] and Gravity-SPHINCS[9] is reduced to subset resilience, the reductions are non-tight since finding a subset cover does not immediately cause a forgery. SPHINCS+[17] fills this gap by introducing interleaved target subset resilience (ITSR), a variant of target subset resilience. Thus, it is also an interesting open question whether our results can be extended to target versions of subset resilience.

There are still a number of open questions around SRH. First, we do not know how to construct a provable SRH based on other assumptions, such as hard mathematical problems (we only ruled out the possibility of being constructed by one-way permutation). Second, we do not know other practical applications of SRH, such as constructing commitment schemes or analyzing hash functions with particular structures. Third, we do not know whether it is possible to construct a CRH from SRH. These questions have been answered with regard to other relaxations of CRH, such as multi-collision-resistant hash functions (MCRH).

Talking about MCRH, it has very similar properties to SRH on our results. However, we cannot observe a precise relation between SRH and MCRH. This is another interesting question about SRH.

Chapter 4

Security Notions for Stateful Signature Schemes

4.1 Stateful Signature Schemes

A stateful signature scheme is a special signature scheme where the signer needs to maintain a state during signing operations. When signing a message, the signer inputs the state along with the secret key and the message, and then, the current state may be updated to a new state.

4.1.1 Definitions

In this subsection, we give a formal definition for stateful signature schemes.

Definition 21 *(Stateful signature scheme.) A stateful signature scheme Γ consists of three polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ along with an associated message space $\mathcal{M} = \{M_\lambda\}$ and an associated state space $\mathcal{ST} = \{ST_\lambda\}$ such that:*

- *The probabilistic key generation algorithm $\mathsf{KeyGen}(1^\lambda; c)$ takes as input the security parameter 1^λ. It outputs a pair of keys (pk, sk) and a state st_0. The behavior of KeyGen is determined by a random coin c uniformly chosen from $\{0,1\}^\lambda$.*

- *The signing algorithm Sign$_{sk}(m, st)$ takes as input a secret key sk, a message $m \in M_\lambda$ and a state $st \in ST_\lambda$. It outputs a signature and a new state (σ, st').*

- *The deterministic verification algorithm Ver$_{pk}(m, \sigma)$ takes as input a public key pk, a message $m \in M_\lambda$ and a signature σ. It outputs a bit $b \in \{0, 1\}$.*

Obviously, a stateless signature scheme is a special case of a stateful signature scheme, where $st \equiv \emptyset$.

Similar to a stateless signature scheme, a stateful signature scheme needs to be correct. It is a natural idea to define the correctness as $\mathsf{Ver}_{pk}(m, \sigma) = 1$ for any $(pk, sk, st_0) \leftarrow \mathsf{Gen}(1^\lambda)$, $m \in M_\lambda$, $st \in ST_\lambda$ and $\sigma \leftarrow \mathsf{Sign}_{sk}(m, st)$, but this is too strong for a general stateful signature scheme. In some cases, there are some $st \in ST_\lambda$ that never appear as the signer's state. We call them "invalid" states. It is meaningless to confirm the validity of signatures that are signed with such invalid states. We thus define the correctness of stateful signature schemes as follows.

Definition 22 *Let $\Gamma = (\mathit{Gen}, \mathit{Sign}, \mathit{Ver})$ be a stateful signature with state space $\{ST_\lambda\}$. For $(pk, sk, st_0) \leftarrow \mathit{Gen}(1^\lambda)$, a valid state space $ST_\lambda^\top(sk, st_0) \subset ST_\lambda$ is described as follows:*

$$ST_\lambda^\top(sk, st_0) := \{st_0\} \cup \{st' | \exists m \in M_\lambda, st \in ST_\lambda^\top(sk, st_0) : (\sigma, st') \leftarrow \mathit{Sign}_{sk}(m, st)\}. \tag{4.1}$$

We say a stateful signature scheme $\Gamma = (\mathit{KeyGen}, \mathit{Sign}, \mathit{Ver})$ is correct if for any $(pk, sk, st_0) \leftarrow \mathit{KeyGen}(1^\lambda)$, $m \in M_\lambda$, and $st \in ST_\lambda^\top(sk, st_0)$, if $(\sigma, *) \leftarrow \mathit{Sign}_{sk}(m, st)$, then $\mathit{Ver}_{pk}(m, \sigma) = 1$ holds.

The correctness of stateless signature schemes is slightly different than that of stateful schemes. A stateless signature σ is a valid signature as long as it is generated by $\mathsf{Sign}_{sk}(m)$. However, for a stateful signature scheme, σ is a valid signature if it is generated by $\mathsf{Sign}_{sk}(m, st)$, where the state st is a valid state. It is possible that $\mathsf{Sign}_{sk}(m, st)$ outputs an invalid signature when st is not valid. In some cases, the signing algorithm of a stateful signature scheme contains a verification operation for checking whether the state is valid. This verification operation is helpful for protecting the security of the scheme in cases where an attacker can exchange the current state with a malicious state.

4.1.2 Security Notions

In this section, we introduce the security notions for stateful signature schemes. We consider various adversaries that have different levels of control over the messages and states.

First, we consider the adversary, which is identical to the that in standard definition. In this case, the adversary can obtain the signatures of chosen messages. The states are maintained by the signing oracle and not given to the adversary. We call this scenario a *hidden state attack* (HSA). The formal definition is depicted as follows.

Definition 23 *Let $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be a stateful signature scheme. If for all probabilistic, polynomial-time adversaries $\mathcal{A}$ in Figure 4.1 there exists a negligible function $\epsilon : \mathbb{N} \to \mathbb{R}$ such that for every $\lambda \in \mathbb{N}$,*

$$\Pr[\boldsymbol{Exp}^{\mathsf{EU\text{-}HSCMA}}_{\Gamma,\mathcal{A}}(\lambda) = 1] \leq \epsilon(\lambda), \tag{4.2}$$

where the probability is taken over the randomness of $\mathcal{A}$ and Γ, we say that Γ is existentially unforgeable under a hidden state chosen message attack (EU-HSCMA).

Next, we take two more powerful adversaries into consideration. In one case, the adversary has knowledge of the current state maintained by the signing oracle. We call this *known state attack* rather than a hidden state attack. In another case, the adversary can not only obtain the current states but also change them. This means that the adversary is able to request signatures for a message m along with any state st as desired, and it obtains a corresponding signature and a new state in response. We call this a *chosen state attack*. The formal definition is depicted as follows.

```
$\mathbf{Exp}^{\textsf{EU-HSCMA}}_{\Gamma,\mathcal{A}}(\lambda)$
$\mathcal{Q}_{msg} \leftarrow \emptyset$
$(pk, sk, st) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{HSCMA}}}(1^\lambda, pk)$
$b \leftarrow \mathsf{Ver}_{pk}(m^\star, \sigma^\star)$
**if** $b = 1 \wedge m^\star \notin \mathcal{Q}$ **then**
    **return** 1
**return** 0

$\mathcal{O}^{\mathsf{HSCMA}}(m)$
$\mathcal{Q}_{msg} \leftarrow \mathcal{Q}_{msg} \cup \{m\}$
$(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$
$st \leftarrow st'$
**return** $\sigma$
```

Figure 4.1: Definition of the experiment $\mathbf{Exp}^{\textsf{EU-HSCMA}}_{\Gamma,\mathcal{A}}(\lambda)$ from Definition 23.

Definition 24 *Let $\Gamma = (\mathit{KeyGen}, \mathit{Sign}, \mathit{Ver})$ be a stateful signature scheme. Let $* \in \{\mathit{KSCMA}, \mathit{CSCMA}\}$. If for all probabilistic, polynomial-time adversaries $\mathcal{A}$ in Figure 4.2 there exists a negligible function $\epsilon : \mathbb{N} \to \mathbb{R}$ such that for every $\lambda \in \mathbb{N}$,*

$$\Pr[\boldsymbol{Exp}^{\mathit{EU}\text{-}*}_{\Gamma,\mathcal{A}}(\lambda) = 1] \leq \epsilon(\lambda), \tag{4.3}$$

where the probability is taken over the coins flipped by $\mathcal{A}$ and the choice of c, we say:

- *Γ is existentially unforgeable under a chosen state chosen message attack (EU-KSCMA) if $* = \mathit{KSCMA}$.*
- *Γ is existentially unforgeable under a chosen state chosen message attack (EU-CSCMA) if $* = \mathit{CSCMA}$.*

It is very reasonable to consider security against known state and chosen state adversaries in practice. When using a stateful signature scheme, the secret key and the states can be saved separately. The secret key can be saved in read-only storage while the state is maintained in writable memory. For some side-channel attackers, it is much easier to extract information from writable memory than from read-only memory.

```
$\mathbf{Exp}^{\mathsf{EU}\text{-}*}_{\Gamma,\mathcal{A}}(\lambda)$
$\mathcal{Q}_{msg}, \mathcal{Q}_s \leftarrow \emptyset$
$(pk, sk, st) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}^*}(1^\lambda, pk, st)$
$b \leftarrow \mathsf{Ver}_{pk}(m^\star, \sigma^\star)$
**if** $b = 1 \wedge m^\star \notin \mathcal{Q}_{msg}$ **then**
  **return** 1
**return** 0
```

```
$\mathcal{O}^{n\text{-}\mathsf{wCSCMA}}(m, st)$
**if** $\exists i$ s.t. $(st, i) \in \mathcal{Q}_s$ **then**
    **if** $i \geq n$ **return** $(\perp, st)$
    $\mathcal{Q}_s \leftarrow \mathcal{Q}_s \cup \{(st, i+1)\} \setminus \{(st, i)\}$
**else** $\mathcal{Q}_s \leftarrow \mathcal{Q}_s \cup \{(st, 1)\}$
$\mathcal{Q}_{msg} \leftarrow \mathcal{Q}_{msg} \cup \{m\}$
$(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$
**return** $(\sigma, st')$
```

```
$\mathcal{O}^{\mathsf{KSCMA}}(m)$
$\mathcal{Q}_{msg} \leftarrow \mathcal{Q}_{msg} \cup \{m\}$
$(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$
$st \leftarrow st'$
**return** $(\sigma, st')$
```

```
$\mathcal{O}^{\mathsf{CSCMA}}(m, st)$
$\mathcal{Q}_{msg} \leftarrow \mathcal{Q}_{msg} \cup \{m\}$
$(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$
**return** $(\sigma, st')$
```

Figure 4.2: Definitions of the experiments from Definitions 24 and 25

.

It is obvious that being EU-CSCMA implies being EU-KSCMA and further implies being EU-HSCMA. In addition, we prove that being EU-CSCMA is equivalent to being *stateless* EU-CMA. If a stateless signature scheme is EU-CMA, it is EU-CMA immediately because states play no role in this case. Next, we show that a stateful EU-CSCMA signature scheme can be simply converted to a stateless EU-CMA scheme.

Theorem 10 *(EU-CSCMA $\leftrightarrow$ stateless EU-CMA.) Suppose a stateful signature scheme*
$\Gamma = (\mathit{KeyGen}, \mathit{Sign}, \mathit{Ver})$ *is EU-CSCMA; then, there exists an EU-CMA stateless signature scheme* Γ'.

Proof. $\Gamma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ runs as follows:

When KeyGen' takes 1^λ as input, it runs $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk, st_0)$ and outputs $(pk', sk') = ((pk, st_0), (sk, st_0))$. When Sign' takes as input a message m and secret key $sk' = (sk, st_0)$, it runs $\mathsf{Sign}_{sk}(m, st) \rightarrow (\sigma, st_0)$ and outputs σ. When Ver' takes as input $pk' = (pk, st_0)$, m and σ, it outputs $b \leftarrow \mathsf{Ver}_{pk}(m, \sigma)$.

Note that the behaviors of $\mathcal{O}_{\Gamma'}^{\mathsf{CMA}}(\cdot)$ and $\mathcal{O}_{\Gamma}^{\mathsf{CSCMA}}(\cdot, st_0)$ are the same, and the verification algorithms of Γ' and Γ are the same as well. This means that if there exists an adversary $\mathcal{A}$ that can break the CMA security of Γ', there exists a reduction $\mathcal{R}^{\mathcal{A}}$ that can break the CSCMA security of Γ. Here, $\mathcal{R}^{\mathcal{A}}$ simply simulates $\mathcal{O}_{\Gamma'}^{\mathsf{CMA}}(\cdot)$ by $\mathcal{O}_{\Gamma}^{\mathsf{CSCMA}}(\cdot, st_0)$ for $\mathcal{A}$.

$\square$

From what has been discussed above, if a stateful signature scheme is EU-CSCMA, it is unnecessary to maintain any state at all. Thus, this notion is too strong and goes beyond what we expect for stateful signature schemes.

In the following, we present a more relaxed notion that fits our expectations. We call it an *n-weak chosen state attack* (n-wCSA). In this case, the adversary can also query the signing oracle with a message and a state, but there is an additional restriction where the adversary is not allowed to query on the same state more than n times. A formal definition follows.

Definition 25 *Suppose $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ compose a stateful signature scheme. If for all probabilistic, polynomial-time adversaries $\mathcal{A}$ in Figure 4.2 there exists a negligible function $\epsilon : \mathbb{N} \to \mathbb{R}$ such that for every $\lambda \in \mathbb{N}$,*

$$\Pr[\boldsymbol{Exp}_{\Gamma,\mathcal{A}}^{\mathsf{EU\text{-}n\text{-}wCSCMA}}(\lambda) = 1] \leq \epsilon(\lambda), \tag{4.4}$$

where the probability is taken over the randomness of $\mathcal{A}$ and Γ, we say that Γ is existentially unforgeable under an n-weak chosen state chosen message attack (EU-n-wCSCMA).

Note that if n is extremely large (e.g., exponential with respect to the parameter λ), EU-n-wCSCMA is equivalent to EU-CSCMA since a polynomial-time adversary cannot query the signing oracle that many times. In particular, we simply call 1-wCSCMA wCSCMA for convenience. In the EU-wCSCMA experiment, all states are distinct. The adversary is allowed to read and change the state but not to reuse any state.

We can similarly define the security notions for stateful signature schemes under RMA, where the adversary can only obtain signatures for random messages and hidden/known/chosen/weak-chosen states. This results in HSRMA/KSRMA/CSRMA/wCSRMA security. Starting in the next subsection, we focus on security under CMAs since the results under RMAs can also be proven in similar ways. For convenience, we say CSA, wCSA, KSA, and HSA instead of CSCMA, wCSCMA, KSCMA, and HSCMA.

In the following, we explore the relationships among these security notions.

4.1.3 From HSA-security to KSA-security

It is obvious that EU-KSA implies EU-HSA. Furthermore, we can convert an EU-HSA signature into an EU-KSA signature with the help of a pseudorandom function. Here, the pseudorandom function is used to "encrypt" the states, thereby preventing a KSA adversary from gaining additional information from the knowledge of states.

Theorem 11 *(EU-HSA + PRF $\to$ EU-KSA.) Let Γ be an EU-HSA stateful signature scheme with a state space $\mathcal{ST} = \{0,1\}^\lambda$ and $F_k : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$*

```
$\mathsf{KeyGen}'(1^\lambda)$
$(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$r, k \xleftarrow{\$} \{0,1\}^\lambda$
$\widetilde{sk} := (sk, k)$, $\widetilde{st_0} := (r, F_k(r) \oplus st_0)$
**return** $(pk, \widetilde{sk}, \widetilde{st_0})$

$\mathsf{Sig}'_{\widetilde{sk}}(m, \widetilde{st})$
Parse $\widetilde{sk} = (sk, k)$, $\widetilde{st} = (r, t)$
$st = t \oplus F_k(r)$
$(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$
$r' \xleftarrow{\$} \{0,1\}^\lambda$
$\widetilde{st'} := (r', F_k(r') \oplus st')$
**return** $(\sigma, \widetilde{st'})$

$\mathsf{Ver}'_{pk}(m, \sigma)$
**return** $\mathsf{Ver}_{pk}(m, \sigma)$
```

Figure 4.3: Construction of an EU-HSA scheme from an EU-KSA scheme.

be a pseudorandom function. Then, the $\Gamma' = (\mathit{KeyGen}', \mathit{Sign}', \mathit{Ver}')$ depicted in Figure 4.3 is an EU-KSA stateful signature scheme.

Proof.

We show that the success probability of a probabilistic polynomial-time adversary in the EU-KSA experiment against Γ' is negligible as long as Γ is EU-HSA and F is a pseudorandom function. We consider the following sequence of games.

- **Game 0** is the original EU-KSA experiment against Γ'.
- **Game 1** differs from **Game 0** in the following ways. In KeyGen', it picks a random function $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$. After that, all the F_k used in KeyGen' and Sign' are replaced by f.

  Suppose the success probability of adversary $\mathcal{A}$ differs in **Game 0** and **Game 1**, and we construct a distinguisher $D^{\mathcal{A}}$ for breaking the pseudorandomness of F. Given access to an oracle $\mathcal{O} : \{0,1\}^\lambda \to \{0,1\}^\lambda$, which is either $F_k(\cdot)$ or a truly random function f, D runs $(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and randomly picks $r \in \{0,1\}^\lambda$. Let $st = st_0$. Then, it computes $\widetilde{st} = (r, st_0 \oplus \mathcal{O}(r))$ by making a query to $\mathcal{O}$. After that, D runs $\mathcal{A}(1^\lambda, pk, \widetilde{st})$. When $\mathcal{A}$ queries for a signature for a message m, D runs $(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$ and updates st with st'. Then, D randomly picks $r' \leftarrow \{0,1\}^\lambda$, queries $\mathcal{O}(r')$, and returns $(\sigma, st \oplus \mathcal{O}(r'))$ to $\mathcal{A}$. When $\mathcal{A}$ outputs a forgery $(m^\star, \sigma^\star)$, D outputs 1 iff $\mathsf{Ver}_{pk}(m^\star, \sigma^\star) = 1$.

  If $\mathcal{O}$ is a pseudorandom function F_k with a random $k \in \{0,1\}^\lambda$, D perfectly simulates the challenge and the signing oracle for $\mathcal{A}$ in **Game 0**. D outputs 1 iff $\mathcal{A}$ succeeds in **Game 0**. On the other hand, if $\mathcal{O}$ is a truly random function f, D perfectly simulates those in **Game 1** and outputs 1 iff $\mathcal{A}$ succeeds in **Game 1**. Thus, we have

  $$|\Pr[\mathbf{Exp}^{\textbf{Game 0}}_{\Gamma',\mathcal{A}}(\lambda)] - \Pr[\mathbf{Exp}^{\textbf{Game 1}}_{\Gamma',\mathcal{A}}(\lambda)]| = |\Pr[D^{F_k}(1^\lambda)] - \Pr[D^{f}(1^\lambda)]| = Adv^{\text{Ind-PRF}}_{F,D}(\lambda), \tag{4.5}$$

  which is negligible if F is a pseudorandom function.

- **Game 2** differs from **Game 1** in the following ways. Let $\mathcal{Q}_{rand} := \{r\}$, where r is the randomness chosen in KeyGen'. In Sign', every time a fresh r' is chosen randomly from $\{0,1\}^\lambda$, Sign' additionally checks whether $r' \in \mathcal{Q}_{rand}$. If so, it halts and **Game 2** outputs 0. Otherwise, let $\mathcal{Q}_{rand} = \mathcal{Q}_{rand} \cup r'$.

  Suppose the adversary can query for at most q signatures during the experiment. Let E be the event that Sign' halts during **Game 2**. **Game 2** differs from **Game 1** only if E happens, that is, two identical randomness values are repeatedly chosen from among $q+1$ choices. Note that r and r' are randomly chosen in $\{0,1\}^\lambda$, and we have $\Pr[E] = 1-(1-\frac{1}{2^\lambda})^{q+1} < \frac{q+1}{2^\lambda}$. Thus, we have

$$|\Pr[\mathbf{Exp}^{\textbf{Game 1}}_{\Gamma',\mathcal{A}}(\lambda)] - \Pr[\mathbf{Exp}^{\textbf{Game 2}}_{\Gamma',\mathcal{A}}(\lambda)]| < \frac{q+1}{2^\lambda}, \tag{4.6}$$

  which is a negligible function.

- In **Game 2**, state $\widetilde{st}$ in the experiment consists of $(r, f(r) \oplus st)$, where r is a fresh random string and f is a truly random function. Thus, $f(r) \oplus st$ is uniformly distributed in $\{0,1\}^\lambda$. We construct a reduction $\mathcal{R}$ that can break the HSA security of Γ. Given pk, $\mathcal{R}$ randomly chooses $r \in \{0,1\}^\lambda$ and $y \in \{0,1\}^\lambda$ and then runs $\mathcal{A}(1^\lambda, pk, (r,y))$. Every time $\mathcal{A}$ queries with a message m, $\mathcal{R}$ queries $\mathcal{O}^{\mathsf{HSA}}(m)$ and obtains σ. After that, $\mathcal{R}$ randomly chooses $r' \in \{0,1\}^\lambda$ and $y' \in \{0,1\}^\lambda$. If r' appeared before, $\mathcal{R}$ halts. Otherwise, $\mathcal{R}$ returns $(\sigma, (r', y'))$ to $\mathcal{A}$. When $\mathcal{A}$ outputs $(m^\star, \sigma^\star)$, $\mathcal{R}$ outputs $(m^\star, \sigma^\star)$ as a forgery of Γ'.

  Since $f(r) \oplus st$ is uniformly distributed in $\{0,1\}^\lambda$, $\mathcal{R}$ perfectly simulates the $\mathcal{O}^{\mathsf{HSA}}_{\Gamma'}$ for $\mathcal{A}$. Note that the verification algorithms of Γ and Γ' are the same, and $\mathcal{R}$ succeeds if $\mathcal{A}$ succeeds in forging a signature for Γ'. Thus, we have

$$\Pr[\mathbf{Exp}^{\textbf{Game 2}}_{\Gamma',\mathcal{A}}(\lambda)] \leq \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}HSA}}_{\Gamma,\mathcal{R}^{\mathcal{A}}}(\lambda)], \tag{4.7}$$

  which is negligible since Γ is EU-HSA.

To summarize, we have

$$\Pr[\mathbf{Exp}^{\mathsf{EU\text{-}KSA}}_{\Gamma',\mathcal{A}}(\lambda)] \leq \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}HSA}}_{\Gamma,\mathcal{R}^{\mathcal{A}}}(\lambda)] + Adv^{\mathsf{Ind\text{-}PRF}}_{F,D^{\mathcal{A}}}(\lambda) + \frac{q+1}{2^\lambda}, \tag{4.8}$$

which is negligible due to the abovementioned assumptions. □

4.1.4 KSA-security and wCSA-security

Note that EU-CSA implies EU-n-wCSA and further implies EU-KSA, but the relationship between EU-KSA and EU-n-wCSA is not clear. Many well-known examples of stateful signature schemes (such as the MSS [87]) are both EU-KSA and EU-wCSA, but the next fact shows that EU-KSA does not imply EU-wCSA in general.

Lemma 16 *(EU-KSA $\not\Rightarrow$ EU-wCSA.) There exists an EU-KSA stateful signature scheme Γ' that is not EU-wCSA.*

Proof. Suppose $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ compose an EU-KSA stateful signature scheme with a state space $\{ST_\lambda\}$. Without loss of generality, let $\perp$ be an element such that $\perp \notin ST_\lambda$ for every λ. Consider a stateful signature scheme $\Gamma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ with a state space $\mathcal{ST}' = \{ST'_\lambda\}$, where $ST'_\lambda = ST_\lambda \cup \{\perp\}$. Taking as input 1^λ with a random coin c, KeyGen' runs $(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda; c)$. Then, it outputs (pk', sk'), where $pk' := pk$ and $sk' := (sk, c)$. When signing a message m together with a state $st \in ST'_\lambda$, Sign' checks whether $st = \perp$. If so, it outputs $(\sigma, st) := (c, \perp)$. Otherwise, it outputs $(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$. When verifying (m, σ), Ver' checks whether σ is a coin or a signature of Γ. If it is a coin, it runs $(\widetilde{pk}, \widetilde{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda; \sigma)$ and outputs 1 iff $\widetilde{pk} = pk$. If σ is a signature, it outputs $b \leftarrow \mathsf{Ver}_{pk}(m, \sigma)$.

It is not hard to see that if Γ is correct and EU-KSA, Γ' is also correct and EU-KSA (since the behaviors of two signing oracles in KSA experiments are the same). However, Γ' is never EU-wCSA. An adversary can query $\mathcal{O}^{\mathsf{wCSA}}_{\Gamma'}(m, \perp)$, obtain the coin c from the answer, and generate a forgery $(m^\star, c)$ for any $m^\star \neq m$. □

Despite the negative result, it is possible to convert an EU-KSA stateful signature scheme to an EU-wCSA scheme. To avoid a wCSA adversary obtaining

```
$\mathsf{KeyGen}'(1^\lambda)$
    $(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$
    $k \xleftarrow{\$} K$
    $\widetilde{sk} := (sk, k)$, $\widetilde{st_0} := (st_0, F_k(st_0))$
    **return** $(pk, \widetilde{sk}, \widetilde{st_0})$

$\mathsf{Sig}'_{\widetilde{sk}}(m, \widetilde{st})$
    Parse $\widetilde{sk} = (sk, k)$, $\widetilde{st} = (st, t)$
    **if** $t \neq F_k(st)$ **return** $(\perp, \widetilde{st})$
    $(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$
    $\widetilde{st'} := (st', F_k(st'))$
    **return** $(\sigma, \widetilde{st'})$

$\mathsf{Ver}'_{pk}(m, \sigma)$
    **return** $\mathsf{Ver}_{pk}(m, \sigma)$
```

Figure 4.4: Construction of an EU-wCSA scheme from an EU-KSA scheme.

additional information from maliciously chosen states, we can use a pseudorandom function to "authenticate" the states so that it is infeasible for an wCSA adversary to obtain a fresh state that is not output by the signing oracle. If so, a wCSA adversary can only query with the current state of the signing oracle, and thus, it is reduced to a KSA adversary. The generic construction is depicted in Figure 4.4.

Theorem 12 *(KSA + PRF → wCSA.) Let Γ be an EU-KSA stateful signature scheme with a state space $\mathcal{ST} = \{ST_\lambda\}$ and $F : ST_\lambda \times K \to \{0,1\}^\lambda$ as a pseudorandom function. Then, the $\Gamma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ depicted in Figure 4.4 is an EU-wCSA stateful signature scheme.*

Proof. We show that the success probability of a probabilistic polynomial-time adversary in an EU-wCSA experiment against Γ' is negligible as long as Γ is EU-KSA. We consider the following sequence of games.

- **Game 0** is the original EU-wCSA experiment against Γ'.

- **Game 1** differs from **Game 0** in that in KeyGen' and Sign', a truly random function $f : ST_\lambda \to \{0,1\}^\lambda$ is used instead of the pseudorandom function F_k.

  Suppose that the success probability of an adversary $\mathcal{A}$ differs in **Game 0** and **Game 1**; we construct a distinguisher D to break the pseudorandomness of F_k. Given an oracle $\mathcal{O}$ that is either $F_k(\cdot)$ or $f(\cdot)$, the distinguisher D runs $(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and computes $\widetilde{st} = (st_0, \mathcal{O}(st))$. Then, D runs $\mathcal{A}(1^\lambda, pk, \widetilde{st})$. When $\mathcal{A}$ queries with $(m, \widetilde{st})$, where $\widetilde{st} = (st, t)$, D checks whether $t = \mathcal{O}(st)$. If not, it returns $(\perp, \widetilde{st})$. Otherwise, it runs $(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$ and computes $\widetilde{st'} = (st', \mathcal{O}(st'))$. Then, it returns $(\sigma, \widetilde{st'})$ to $\mathcal{A}$. After the queries are completed, $\mathcal{A}$ outputs a forgery $(m^\star, \sigma^\star)$. D outputs 1 if $\mathsf{Ver}_{pk}(m^\star, \sigma^\star)) = 1$ and outputs 0 otherwise.

  In cases where $\mathcal{O}$ is $F_k(\cdot)$, the distinguisher D perfectly simulates the challenge and the signing oracle in **Game 0** for $\mathcal{A}$. Thus, $\Pr[D^{F_k}(1^\lambda)] = \Pr[\mathbf{Exp}_{\Gamma',\mathcal{A}}^{\mathbf{Game\ 0}}(\lambda)]$. In cases where $\mathcal{O}$ is f, D simulates those in **Game 1**. Thus, $\Pr[D^f(1^\lambda)] = \Pr[\mathbf{Exp}_{\Gamma',\mathcal{A}}^{\mathbf{Game\ 1}}(\lambda)]$. Therefore, we have

$$|\Pr[\mathbf{Exp}_{\Gamma',\mathcal{A}}^{\mathbf{Game\ 0}}(\lambda)] - \Pr[\mathbf{Exp}_{\Gamma',\mathcal{A}}^{\mathbf{Game\ 1}}(\lambda)]| = |\Pr[D^{F_k}(1^\lambda)] - \Pr[D^f(1^\lambda)]| = Adv_{F,D}^{\text{Ind-PRF}}(\lambda), \tag{4.9}$$

  which is negligible if F is pseudorandom.

- **Game 2** differs from **Game 1** in that all signing queries with a fresh state $\widetilde{st}$ (which means $\widetilde{st}$ is not contained in the answers of previous queries) will be answered by $(\perp, \widetilde{st})$.

  The outputs of **Game 2** and **Game 1** differ only if the adversary ever queried with a fresh state and was returned a valid signature. For every fresh state $\widetilde{st} = (st, t)$, $f(st)$ is never calculated by $\mathcal{O}^{\mathsf{wCSA}}$ and is thus uniformly distributed in $\{0,1\}^\lambda$. It can be returned with a valid signature only if $t = f(st)$, for which the probability is at most $2^{-\lambda}$. Suppose the adversary can query for q signatures; we have

$$|\Pr[\mathbf{Exp}^{\textbf{Game 1}}_{\Gamma',\mathcal{A}}(\lambda)] - \Pr[\mathbf{Exp}^{\textbf{Game 2}}_{\Gamma',\mathcal{A}}(\lambda)]| \leq \frac{q}{2^{\lambda}}, \tag{4.10}$$

which is a negligible function.

- In **Game 2**, when the adversary queries with $(m, \widetilde{st})$, where $\widetilde{st} = (st, t)$, the query can be answered (by a non-error signature) only if $\widetilde{st}$ is ever returned by the signing oracle. In addition, due to the rule of the wCSA experiment, if the adversary queries with a state more than once, the query will be answered with $\perp$. Thus, the adversary can only obtain a valid signature from the signing oracle if it queries with the state contained in the output of the last query (or the challenge). This is the same as the strategy in a KSA experiment.

  Consider a reduction $\mathcal{R}^{\mathcal{A}}$ against the EU-KSA experiment against Γ. Given a challenge (pk, st_0), $\mathcal{R}$ picks $t_0 \leftarrow \{0,1\}^{\lambda}$ uniformly (at random) and runs $\mathcal{A}(1^{\lambda}, pk, (st_0, t_0))$. After that, $\mathcal{R}$ records $state = (st_0, t_0)$. When $\mathcal{A}$ queries $(m, \widetilde{st})$, $\mathcal{R}$ checks whether $\widetilde{st} = state$. If not, it replies $(\perp, \widetilde{st})$. Otherwise, $\mathcal{R}$ queries $\mathcal{O}^{\mathsf{KSA}}$ with m and obtains (σ, st'). Then, $\mathcal{R}$ returns $(\sigma, (st', t'))$ to $\mathcal{A}$ with a uniformly random $t' \in \{0,1\}^{\lambda}$ and updates $state = (st', t')$.

  When $\mathcal{A}$ outputs a forgery $(m^{\star}, \sigma^{\star})$, this is also a forgery for Γ. This means that $\mathcal{R}$ succeeds in the KSA experiment against Γ as long as $\mathcal{A}$ succeeds in **Game 2**. Thus, we have

$$\Pr[\mathbf{Exp}^{\textbf{Game 2}}_{\Gamma',\mathcal{A}}(\lambda)] \leq \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}KSA}}_{\Gamma,\mathcal{R}^{\mathcal{A}}}(\lambda)], \tag{4.11}$$

  which is negligible if Γ is EU-KSA.

To summarize, we have

$$\Pr[\mathbf{Exp}^{\mathsf{EU\text{-}wCSA}}_{\Gamma',\mathcal{A}}(\lambda)] \leq \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}KSA}}_{\Gamma,\mathcal{R}^{\mathcal{A}}}(\lambda)] + Adv^{\texttt{Ind-PRF}}_{F,D^{\mathcal{A}}}(\lambda) + \frac{q}{2^{\lambda}}, \tag{4.12}$$

which is negligible due to the abovementioned assumptions.

□

We next demonstrate how to construct an EU-KSA scheme from an EU-wCSA scheme. This construction is natural: suppose there is an EU-wCSA stateful signature scheme; the remaining work is to ensure that the same state never appears twice during the process of signing.

To establish this construction, we need an additional property called samplability, which implies that valid states can be sampled from the state space.

Definition 26 *(Samplability.) We say a stateful signature is q-samplable if there is a polynomial-time algorithm $\mathsf{Samp}_q(pk, i)$ such that:*

- *Suppose $(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Taking as input pk and an index $i \in [q]$, $\mathsf{Samp}_q(pk, i)$ outputs a valid state $st \in ST_\lambda^\top(sk, st_0) \subset ST_\lambda$.*
- *For any $i_1, i_2 \in [q]$, let $st_1 \leftarrow \mathsf{Samp}_q(pk, i_1)$ and $st_2 \leftarrow \mathsf{Samp}_q(pk, i_2)$. If $i_1 \neq i_2$, then $st_1 \neq st_2$.*

We require $q \leq |ST_\lambda|$; otherwise, there exists an index i such that $\mathsf{Samp}_q(pk, i)$ can never sample a state from ST_λ. Note that not all stateful signature schemes are samplable. A counterexample is the scheme in Figure 4.4. It is difficult for one who does not know the secret key to generate a valid state in that case.

Suppose that a stateful signature is EU-wCSA and q-samplable. We show the construction of a q-time EU-KSA stateful signature by the construction method depicted in Figure 4.5. Here, the state is a counter recording the number of signing operations. For each signing, the signer samples a state by using the sampling algorithm and signs the message.

Theorem 13 *(wCSA+ Samplability $\rightarrow$ KSA.) Let $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be an EU-wCSA and q-samplable stateful signature scheme, whose sampling algorithm is Samp_q. Then, the $\Gamma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ depicted in Figure 4.5 is a q-samplable, q-time and EU-KSA stateful signature scheme with a state space $\mathcal{ST}' = \{[q]\}$.*

Proof.

```
$\mathsf{KeyGen}'(1^\lambda)$
$(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\widetilde{st_0} := 0$
**return** $(pk, sk, \widetilde{st_0})$

$\mathsf{Sig}'_{sk}(m, \widetilde{st})$
$st \leftarrow \mathsf{Samp}_q(pk, \widetilde{st})$
$(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$
$\widetilde{st'} = \widetilde{st} + 1$
**return** $(\sigma, \widetilde{st'})$

$\mathsf{Ver}'_{pk}(m, \sigma)$
**return** $\mathsf{Ver}_{pk}(m, \sigma)$
```

Figure 4.5: Construction of an EU-KSA scheme from an EU-wCSA scheme.

Γ' is q-samplable since $\mathsf{Samp}'_q(pk, i) = i$ is a sampling algorithm for Γ'. Next we prove the KSA security of the scheme.

Suppose there is an adversary $\mathcal{A}$ that succeeds in the EU-KSA experiment against Γ' with nonnegligible probability; then, there is a reduction $\mathcal{R}^{\mathcal{A}}$ that can break the wCSA security of Γ. In the EU-wCSA experiment against Γ, when $\mathcal{R}$ receives a challenge (pk, st_0), it runs $\mathcal{A}(1^\lambda, pk, 0)$ and sets $\widetilde{st} := 0$. Every time $\mathcal{A}$ queries a message m, $\mathcal{R}$ samples $st \leftarrow \mathsf{Samp}_q(pk, \widetilde{st})$ and queries $\mathcal{O}_{\Gamma}^{\mathsf{wCSA}}$ on (m, st), thereby obtaining (σ, st'). After that, it updates $\widetilde{st}$ with $\widetilde{st} + 1$ and replies $(\sigma, \widetilde{st})$ to $\mathcal{A}$. After at most q queries, $\mathcal{A}$ outputs a forgery $(m^\star, \sigma^\star)$ for Γ', and this is also a forgery for Γ. Due to the property of the sampling algorithm, $\mathcal{R}$ never queries the same state repeatedly, so it perfectly simulates the signing oracle $\mathcal{O}_{\Gamma'}^{\mathsf{KSA}}$ for $\mathcal{A}$. Thus, the KSA security of Γ' is reduced to the wCSA security of Γ.

□

4.1.5 From wCSA security to stateless CMA security

Compared with a stateful signature scheme, a stateless signature scheme is more convenient in practice since it is unnecessary to maintain the state. In Theorem 10, we showed that a stateless signature scheme can be constructed from an EU-CSA stateful scheme. In this subsection, we consider how to construct a stateless signature scheme from an EU-wCSA or EU-n-wCSA stateful signature scheme, which requirea weaker assumptions than CSA security.

To construct a stateless signature scheme with an EU-wCSA q-samplable stateful scheme, a key point is to avoid sampling two identical states without maintaining the states. This can be realized using a collision-resistant function that maps the message space to $[q]$.

Theorem 14 *(wCSA + Samplability + CRH → stateless CMA.) Let* $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *be a stateful signature scheme that is EU-wCSA and q-samplable. Let $\mathcal{H}$ be a collision-resistant function family mapping to $[q]$. Then, there exists an EU-CMA stateless signature scheme Γ'.*

Proof. Let Samp_q be the sampling algorithm of Γ. Γ' is described as follows. KeyGen' and Ver' are the same as those of Γ except that a function H is chosen from $\mathcal{H}$ and is contained in pk and sk. For the signing algorithm, when taking a message m as input, Sign' samples $st \leftarrow \mathsf{Samp}_q(pk, H(m))$ and runs $(\sigma, st') \leftarrow \mathsf{Sign}(m, st)$. Then, it outputs σ.

We show that Γ' is EU-CMA if Γ is EU-wCSA and that $\mathcal{H}$ is collision resistant. Suppose there is an adversary $\mathcal{A}$ that can succeed in the EU-CMA experiment against Γ' with nonnegligible probability; then, there is a reduction $\mathcal{R}^{\mathcal{A}}$ that can succeed in the EU-wCSA experiment against Γ or can output a collision of $\mathcal{H}$. First, $\mathcal{R}$ receives a challenge pk for the EU-wCSA experiment against Γ and a challenge H for the collision resistance experiment of $\mathcal{H}$. Then, $\mathcal{R}$ runs $\mathcal{A}(1^\lambda, pk)$. When $\mathcal{A}$ queries for the signature of a message m, $\mathcal{R}$ records $(m, H(m))$ in a list $\mathcal{Q}_H$ and checks whether there exists $(m', H(m')) \in \mathcal{Q}_H$ such that $m \neq m'$ but $H(m) = H(m')$. If so, $\mathcal{R}$ returns a collision (m, m') for H. Otherwise, $\mathcal{R}$ queries $(m, \mathsf{Samp}_q(pk, H(m)))$ to $\mathcal{O}_\Gamma^{\mathsf{wCSA}}$. Due to the property of samplability,

$\mathsf{Samp}(pk, H(m))$ is never queried as a state prior to this point. $\mathcal{O}_{\Gamma}^{\mathsf{wCSA}}$ outputs (σ, st'), where σ is also a valid signature for Γ'. $\mathcal{R}$ returns σ to $\mathcal{A}$. After the queries are completed, $\mathcal{A}$ returns a forgery $(m^\star, \sigma^\star)$, and $\mathcal{R}$ outputs $(m^\star, \sigma^\star)$ as a forgery for Γ.

Let **Collision** be the event that $\mathcal{A}$ queries m and m' at any point, where $H(m) = H(m')$. If **Collision** occurs, $\mathcal{R}$ succeeds in obtaining a collision for $\mathcal{H}$. Otherwise, $\mathcal{R}$ perfectly simulates the signing oracle for $\mathcal{A}$. Then, $\mathcal{R}$ succeeds in the wCSA experiment against Γ if $\mathcal{A}$ succeeds in the CMA experiment against Γ'. Thus, we have

$$\begin{aligned}\Pr[\mathbf{Exp}_{\Gamma',\mathcal{A}}^{\mathsf{EU\text{-}CMA}}(\lambda)] &\leq \Pr[\mathbf{Exp}_{\Gamma',\mathcal{A}}^{\mathsf{EU\text{-}CMA}}(\lambda)|\overline{\mathbf{Collision}}] + \Pr[\mathbf{Collision}] \\ &= \Pr[\mathbf{Exp}_{\Gamma,\mathcal{R}^{\mathcal{A}}}^{\mathsf{EU\text{-}wCSA}}(\lambda)] + Adv_{\mathcal{H},\mathcal{R}^{\mathcal{A}}}^{\mathsf{Coll}}(\lambda),\end{aligned} \tag{4.13}$$

which is negligible if Γ is EU-wCSA and $\mathcal{H}$ is collision resistant. □

This construction only works on the condition that q is super-polynomial. Otherwise, there does not exist such a collision-resistant function family $\mathcal{H}$ mapping to $[q]$, since a birthday attack can always find a collision with constant probability after $q^{1/2}$ queries to H. However, it is possible to convert an EU-n-wCSA scheme into a stateless scheme even if q is polynomial in the security parameter. The stateless scheme runs as follows. Every time the scheme signs for a message, it samples a state by using the sampling algorithm, taking as input a fresh randomness value r, which is uniformly chosen from $[q]$. Then, it signs the message with this state using the EU-n-wCSA stateful signature scheme. If the number of signing operations is not very large, it is of a small probability that the same state is repeatedly chosen more than n. To ensure that this probability is negligible, the number of signing operations needs to be limited.

Theorem 15 *(n-wCSA + Samplability → N-time stateless CMA). Let* $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *be a q-samplable n-wCSA stateful signature scheme and* Samp_q *be a sampling algorithm for* Γ*. The* $\Gamma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ *depicted in figure 4.6 is an EU-wCSA, N-time stateless signature scheme, where* $N(\lambda) = nq^{1-1/n} \cdot O(2^{-\lambda/n})$.

```
$\mathsf{KeyGen}'(1^\lambda; c)$
$(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda; c)$
$\widetilde{sk} := (pk, sk)$
**return** $(pk, (pk, \widetilde{sk}))$

$\mathsf{Sig}'_{\widetilde{sk}}(m)$
Parse $\widetilde{sk} = (pk, sk)$
$r \xleftarrow{\$} [q]$
$st \leftarrow \mathsf{Samp}_q(pk, r)$
$(\sigma, st') \leftarrow \mathsf{Sign}_{sk}(m, st)$
**return** $\sigma$

$\mathsf{Ver}'_{pk}(m, \sigma)$
**return** $\mathsf{Ver}_{pk}(m, \sigma)$
```

Figure 4.6: Construction of an EU-CMA stateless signature scheme from an EU-n-wCSA stateful scheme.

Proof. Let be an adversary in the CMA experiment against Γ' and $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{CMA}}$ at most $N = qn \cdot O(2^{-\lambda/n})$ times. A reduction $\mathcal{R}^{\mathcal{A}}$ for the EU-n-wCSA experiment against Γ runs as follows. Upon receiving the challenge pk and st_0, $\mathcal{R}$ randomly picks N random values $r_1, ..., r_N$ from the set $[q]$. $\mathcal{R}$ halts immediately if there exists a value $r \in [q]$ that is picked at least $n+1$ times. After that, $\mathcal{R}$ runs $\mathcal{A}(1^{\lambda}, pk)$. When $\mathcal{A}$ queries with the message m in the ith query, $\mathcal{R}$ runs $st_i \leftarrow \mathsf{Samp}_q(pk, r_i)$. Then, $\mathcal{R}$ queries (m, st_i) to $\mathcal{O}_{\Gamma}^{n\text{-}\mathsf{wCSA}}$, obtains (σ, st_i'), and returns σ to $\mathcal{A}$. After the queries are completed, $\mathcal{A}$ returns a forgery $(m^{\star}, \sigma^{\star})$ of Γ', and $\mathcal{R}$ returns $(m^{\star}, \sigma^{\star})$ as a forgery of Γ.

Note that r_i is chosen randomly from $[q]$, and $\mathcal{R}$ perfectly simulates $O_{\Gamma'}^{\mathsf{CMA}}$ for $\mathcal{A}$ (if it does not halt). Additionally, since there is no $r \in [q]$ appearing more than n times in $\{r_1, ..., r_N\}$, $\mathcal{R}$ does not query a state more than n times from $\mathcal{O}_{\Gamma}^{n\text{-}\mathsf{wCSA}}$. The only hindrance is that $\mathcal{R}$ may halt when picking random values. Let $E(N, q, d)$ be the event such that out of N samples, a given element in $[q]$ is chosen d times. We have

$$\Pr[E(N,q,d)] = \binom{N}{d}(1-\frac{1}{q})^{N-d}\frac{1}{q^d} < \frac{N!}{(N-d)!d!q^d}. \tag{4.14}$$

Since $\frac{N!}{(N-d)!} < N^d$ and $d! > d^d e^{-d}$ (where e is the base of the natural logarithm), we have

$$\Pr[E(N,q,d)] < \frac{N^d}{d^d e^{-d} q^d} = (\frac{eN}{qd})^d. \tag{4.15}$$

Let $\alpha = \frac{eN}{qn} = \frac{O(2^{-\lambda/n})}{q^{1/n}}$. The probability that a given element is chosen more than n times is

$$\sum_{d=n+1}^{N} \Pr[E(N,q,d)] < \sum_{d=n+1}^{N} (\frac{eN}{qd})^d < \sum_{d=n+1}^{N} (\frac{eN}{qn})^d = \sum_{i=n+1}^{N} \alpha^d < \alpha^n = \frac{O(2^{-\lambda})}{q}. \tag{4.16}$$

Thus, the probability that there is an element in $[q]$ chosen more than n times is at most $O(2^{-\lambda})$, which is the probability that $\mathcal{R}$ halts before it runs $\mathcal{A}$. Therefore,

$$\Pr[\mathbf{Exp}_{\Gamma',\mathcal{A}}^{\mathsf{EU\text{-}CMA}}(\lambda)] \leq \Pr[\mathbf{Exp}_{\Gamma,\mathcal{R}^{\mathcal{A}}}^{\mathsf{EU\text{-}}n\mathsf{\text{-}wCSA}}(\lambda)] + O(2^{-\lambda}), \tag{4.17}$$

which is negligible if Γ is EU-n-wCSA. □

4.2 Constructions for samplable wCSA security

In the previous section, we showed that a q-samplable EU-wCSA or EU-n-wCSA-secure stateful signature scheme can be used to construct an EU-KSA stateful signature scheme or an EU-CMA stateless scheme. Here, q must be a large number (even though it is not required to be super-polynomial), since the numbers of signing operations in our constructions are linearly related to q. Then, a natural question is how to construct a q-samplable EU-wCSA or EU-n-wCSA-secure stateful signature scheme, where q is large enough. In this section, we provide some constructions for these schemes using primitives in "lower levels".

4.2.1 wCSA-secure Stateful Signature Schemes with Higher Samplability

A stateful signature scheme is often constructed from a one-time signature scheme (OTS), where a key pair can only be used to sign once. A CMA-secure OTS can be considered a 1-samplable wCSA-secure stateful signature scheme whose state space has only one element. A wCSA adversary can only query for one signature in this case since it cannot provide two distinct states in the state space. In the literature, there are some practical OTSs based on different kinds of assumptions. For example, Hülsing [67] proposed W-OTS+, which is based on the second preimage resistance of hash functions. Lyubashevsky and Micciancio [85] proposed an efficient lattice-based OTS. They are all efficient but constrained by the number of signing events.

Merkle [87] proposed the Merkle tree structure, which can be used to construct a multi-time signature scheme from an OTS. It essentially enlarges the state space of a stateful signature scheme, resulting in a q-samplable scheme with a larger q than before. We simply call this a scheme with higher samplability. A Merkle tree can be generalized by any static accumulator, as defined in Definition 4. In this section, we demonstrate two general constructions of wCSA-secure stateful signature schemes with higher samplability. The first construction is from a q-samplable EU-wCSA stateful signature scheme to a kq-samplable scheme for any polynomial k. It can be used to construct a polynomial-samplable stateful signature scheme from an OTS. The second construction is from a q-samplable wCSA-secure stateful signature scheme to a q^d-samplable wCSA-secure scheme for any constant integer d. This can be used to construct a higher-samplable stateful signature scheme.

Let $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be a q-samplable EU-wCSA stateful signature scheme and Samp_q be a sampling algorithm for Γ. The first construction is shown in Figure 4.7. It runs the key generation algorithm for Γ k times and accumulates the public keys with a static accumulator A. When signing a message, it picks one of the key pairs of Γ according to the current state, signs the message with the secret key and generates the witness of A for the public key. A state consists of two counters (i, j), where $i \in [q]$ and $j \in [k]$. The former records the number of signatures issued with regard to a key pair, while the latter is the index of the key pair.

Theorem 16 *Let Γ be a q-samplable EU-wCSA stateful signature scheme in which the public key is of length $l = l(\lambda)$, Samp_q be a sampling algorithm for Γ, and A be a static accumulator, where the accumulation domain includes $\{0,1\}^{l(\lambda)}$. Let $G_k : \{0,1\}^\lambda \to \{0,1\}^{k\lambda}$ be a pseudorandom generator. Then, the $\Gamma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ depicted in Figure 4.7 is a kq-samplable EU-wCSA stateful signature scheme.*

Proof. First, we show that Γ' is kq-samplable. For $i' \in [kq]$, let $i' = a + bq$, where $a, b \in [q]$. Then, $\mathsf{Samp}_{kq}^{\Gamma'}(pk, i') = (a, b)$ is a sampling algorithm for Γ'.

Next, we show that Γ' is EU-wCSA if G_k is a pseudorandom generator and $\mathcal{A}$ is a static accumulator. We consider the following sequence of experiments:

- **Game 0** is the original EU-wCSA experiment against Γ'.
- **Game 1** differs from **Game 0** in the following ways. In KeyGen', $(c_0, ..., c_{k-1})$ is randomly chosen from $\{0,1\}^{k\lambda}$ instead of through the pseudorandom generator G_k. After that, $(c_0, ..., c_{k-1})$ is contained in sk'. In KeyGen', $(c_0, ..., c_{k-1})$ are not computed by the pseudorandom generator. Instead, they are directly obtained from sk'.

```
$\mathsf{KeyGen}'(1^\lambda; c)$
$(c_0, c_1, ..., c_{k-1}) \leftarrow G_k(c)$
**For** $j = 0$ to $k - 1$:
    $(pk_j, sk_j, st_j) \leftarrow \Gamma.\mathsf{KeyGen}(1^\lambda; c_j)$
$ak \leftarrow \mathsf{A.Gen}(1^\lambda, k)$
$z \leftarrow \mathsf{A.Eval}_{ak}(pk_0, ..., pk_{k-1})$
$pk' := (z, ak)$
$sk' := (c, pk)$
$st'_0 := (0, 0)$
**return** $(pk', sk', st'_0)$
```

```
$\mathsf{Sig}'_{sk'}(m, st)$
Parse $st = (i, j)$, $sk' := (c, pk)$
**if** $i \notin [q] \vee j \notin [k]$ **return** $(\perp, st)$
$(c_0, c_1, ..., c_{k-1}) \leftarrow G_k(c)$
$(pk_j, sk_j) \leftarrow \Gamma.\mathsf{KeyGen}(1^\lambda; c_j)$
$st_i \leftarrow \mathsf{Samp}_q(pk_j, i)$
$(\sigma, st'_i) \leftarrow \Gamma.\mathsf{Sign}_{sk_j}(m, st_i)$
$w_j \leftarrow \mathsf{A.Wit}_{ak}(pk_j)$
$\sigma' := (\sigma, pk_j, w_j)$
$st' = (i + 1, j)$
**if** $i = q - 1$ **then** $st' = (0, j + 1)$
**return** $(\sigma', st')$
```

```
$\mathsf{Ver}'_{pk'}(m, \sigma)$
Parse $pk' = (z, ak)$, $\sigma' = (\sigma, pk, w)$
$b_1 \leftarrow \Gamma.\mathsf{Ver}_{pk}(m, \sigma)$
$b_2 \leftarrow \mathsf{A.Ver}_{ak}(pk, w, z)$
**return** $b_1 \wedge b_2$
```

Figure 4.7: Construction of a kq-samplable EU-wCSA stateful signature scheme.

Since G_k is a pseudorandom generator, the difference between the success probabilities of **Game 0** and **Game 1** is negligible. Otherwise, we can construct a distinguisher D to break the pseudorandomness of G_k. This reduction is the same as that in previous theorems, so we omit the details. As a result, we have

$$|\Pr[\textbf{Game 0}] - \Pr[\textbf{Game 1}]| \leq Adv_{G_k, D^{\mathcal{A}}}^{\text{Ind-PRG}}(\lambda). \tag{4.18}$$

- In **Game 1**, KeyGen' runs KeyGen k times with k randomness values and accumulates the public keys by using a static accumulator A. We show that if there exists an adversary $\mathcal{A}$ that can succeed in **Game 1**, we can construct a reduction $\mathcal{R}^{\mathcal{A}}$ that can break the wCSA security of Γ or the soundness of A.

  Given the challenges pk and ak, $\mathcal{R}$ runs as follows. Firstly, it randomly picks $t \in [k]$ and sets $pk_t = pk$. After that, it picks k randomness values from $\{0,1\}^\lambda$ and runs KeyGen k times, obtaining k key pairs of Γ. Let pk_i and sk_i be the corresponding key pairs except for that when $i = t$. Note that all the pk_is (including pk_t) are generated by KeyGen with a random input, and $\mathcal{R}$ does not know sk_i. Then, $\mathcal{R}$ runs $z \leftarrow \mathsf{Eval}_{ak}(pk_1, ..., pk_k)$ and runs $\mathcal{A}(1^\lambda, (z, ak), (0, 0))$.

  When $\mathcal{A}$ queries with (m, st), where $st = (i, j)$, $\mathcal{R}^{\mathcal{A}}$ replies as follows. If $i \notin [q]$ or $j \notin [k]$, it returns $(\perp, st)$. Otherwise, it computes $st_i \leftarrow \mathsf{Samp}_q(pk, i)$. If $j \neq t$, it runs $(\sigma, st') \leftarrow \mathsf{Sign}_{sk_j}(m, st_i)$. If $j = t$, it queries $(\sigma, st') \leftarrow \mathcal{O}_{\Gamma}^{\mathsf{wCSA}}(m, st_i)$. After that, it runs $w_j \leftarrow \mathsf{A.Wit}_a k(pk_j)$ and lets $\sigma' = (\sigma, pk_j, w_j)$. Then, $\mathcal{R}$ returns (σ', st) to $\mathcal{A}$.

  Then, $\mathcal{A}$ returns a forgery $(m^\star, \sigma'^\star)$. Suppose it is a valid forgery; let $\sigma'^\star = (\sigma^\star, pk^\star, w^\star)$ and **NewPK** be the event that $pk^\star \notin \{pk_1, ..., pk_k\}$. We consider the following two cases:

  – If **NewPK** occurs, this means $\mathcal{A}$ generates a $pk^\star \notin \{pk_1, ..., pk_k\}$ and $w^\star$ such that $\mathsf{A.Ver}(pk^\star, w, z) = 1$. $\mathcal{R}$ returns $(pk_1, ..., pk_k, pk^\star, w^\star)$, which breaks the soundness of A.

- If **NewPK** does not occur, this means $pk^\star = pk_{t'}$ for some $t' \in [k]$. Since all the pk_is are generated by random coins, the probability that $t' = t$ is $1/k$. If $t' = t$, $\mathcal{R}$ returns $(m^\star, \sigma^\star)$ as a forgery of Γ, thereby breaking the wCSA security of Γ. Thus, the probability of this case is $k \cdot \Pr[\mathbf{Exp}^{\mathsf{wCSA}}_{\Gamma,\mathcal{R}^\mathcal{A}}(\lambda)]$.

We have

$$\begin{aligned}\Pr[\textbf{Game 1}] &= \Pr[\textbf{Game 1} \wedge \textbf{NewPK}] + \Pr[\textbf{Game 1} \wedge \overline{\textbf{NewPK}}]\\ &= Adv^{Accu}_{\mathsf{A},\mathcal{R}^\mathcal{A}}(\lambda) + k \cdot \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}wCSA}}_{\Gamma,\mathcal{R}^\mathcal{A}}(\lambda)]\end{aligned}$$

To summarize, we have

$$\Pr[\mathbf{Exp}^{\mathsf{wCSA}}_{\Gamma',\mathcal{A}}(\lambda)] \leq Adv^{\texttt{Ind-PRG}}_{G_k,D^\mathcal{A}}(\lambda) + Adv^{Accu}_{\mathsf{A},\mathcal{R}^\mathcal{A}}(\lambda) + k \cdot \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}wCSA}}_{\Gamma,\mathcal{R}^\mathcal{A}}(\lambda)], \quad (4.19)$$

which is negligible if G_k is a pseudorandom generator, A is a static accumulator, and Γ is EU-wCSA.

$\square$

The construction above is not practical enough in terms of the running times of the key generation and signing processes, since KeyGen is invoked k times. This implies that it can only extend the samplability property by a polynomial factor. The second construction depicted in Figure 4.8 is a q^d-samplable stateful signature scheme from a q-samplable scheme, and this can extend the samplability property to the dth power. The cost is the linear expansion of a signature.

The construction in Figure 4.8 uses a tree structure of depth d and out-degree q. Each node of the tree contains a key pair of Γ that is generated by a pseudorandom coin. The key pairs at a leaf are used to sign the message, and those at other nodes are used to sign the public keys of child nodes. The state consists of d counters, $(q_1, q_2, ..., q_d)$, where $q_i \in [q]$. $(q_2, q_3, ..., q_d)$ points to an address of a leaf, and q_1 records the number of signatures issued with regard to that leaf. The public key of Γ' is the public key of the root node, and the secret key is the seed generating the tree. Given a message and a state $(q_1, q_2, ..., q_d)$, Γ' signs with the corresponding key pair and generates the $d-1$ signatures from the leaf to the

```
$\mathsf{KeyGen}'(1^\lambda; c)$
$(pk, sk, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda; c)$
$sk' := c$
$st'_0 := (0, 0, ..., 0) \in [q]^d$
**return** $(pk, sk', st'_0)$

$\mathsf{Sig}'_{sk'}(m, st)$
Parse $st = (q_1, q_2, ..., q_d)$
$l := \lceil \log q \rceil$
Let $B_i \in \{0,1\}^l$ be the binary expression of $i$
**If** $\exists q_i \notin [q]$ **return** $(\perp, st)$
$pk_0 := m$, $c = sk'$
**For** $i = 1$ to $d$:
    $c_i := F_c(B_{q_{i+1}} || ... || B_{q_d})$
    $(pk_i, sk_i, st_0) \leftarrow \mathsf{KeyGen}(1^\lambda; c_i)$
    $(\sigma_i, st_i) \leftarrow \mathsf{Sign}_{sk_i}(pk_{i-1}, \mathsf{Samp}_q(pk_i, q_i))$
$\sigma := (\sigma_1, ..., \sigma_d, pk_1, ..., pk_{d-1})$
**If** $q_1 = q_2 = ... = q_d = q - 1$ **then** $st' = \perp$
**else** let $j$ be the smallest $i$ such that $q_i \neq q - 1$
    $st' = (\underbrace{0, ..., 0}_{j-1}, q_j + 1, q_{j+1}, ..., q_d)$
**return** $(\sigma, st')$

$\mathsf{Ver}'_{pk'}(m, \sigma)$
Parse $\sigma = (\sigma_1, ..., \sigma_d, pk_1, ..., pk_{d-1})$
$h_0 = m$, $pk_d = pk'$
**For** $i = 1$ to $d$: $b_i \leftarrow \mathsf{Ver}_{pk_i}(pk_{i-1}, \sigma_i)$
**return** $\bigwedge_{i=1}^{d} b_i$
```

Figure 4.8: Construction of a q^d-samplable EU-wCSA stateful signature scheme.

root. The signature of Γ' contains a signature of m, $d-1$ public keys and $d-1$ signatures in the path.

Theorem 17 *Let $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be a q-samplable EU-wCSA stateful signature scheme, where the public key is of length $l = l(\lambda)$, the message space M_λ includes $\{0,1\}^{l(\lambda)}$, and q is polynomial with respect to λ. Let Samp_q be a sampling algorithm for Γ. Let $F : \{0,1\}^\star \times COIN_\lambda \to COIN_\lambda$ be a pseudorandom function where the input is of arbitrary length. For a constant d, the $\Gamma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ depicted in Figure 4.8 is a q^d-samplable EU-wCSA stateful signature scheme.*

Proof. First, we show that Γ' is q^d-samplable. For $i \in [q^d]$, let $i = \sum_{i=1}^{d} q_i q^{d-1}$, where $q_i \in [q]$. Then, $\mathsf{Samp}'_{pk,q^d}(i) = (q_1, ..., q_d)$ is a sampling algorithm for Γ'.

The proof of wCSA security is similar to Theorem 16. To begin with the EU-wCSA experiment against Γ', we first replace the pseudorandom function with a truly random function. That is, all the key pairs in the tree are generated by random coins, which are selected in the key generation algorithm. All the coins are contained in sk'. When signing messages, the scheme directly uses the coins contained in sk' instead of computing them with a pseudorandom function. The difference between the success probabilities of these two experiments is negligible if F is pseudorandom.

After that, we reduce the EU-wCSA security of Γ' to the EU-wCSA security of Γ. Given a public key pk of Γ, $\mathcal{R}^{\mathcal{A}}$ runs KeyGen' and generates $\frac{q^d-1}{q-1}$ key pairs in the tree. After that, $\mathcal{R}$ randomly chooses a node in the tree. This is called "the target node". Then, $\mathcal{R}$ replaces the public key at the target node with pk. Let pk' be the public key at the root. $\mathcal{R}$ runs $\mathcal{A}(1^\lambda, pk', 0^d)$. When $\mathcal{A}$ queries with the message (m, st), $\mathcal{R}$ signs the message using the secret keys at corresponding nodes except the target node (since $\mathcal{R}$ does not know the secret key of the target node). When it reaches the target node, $\mathcal{R}$ obtains the corresponding signature by querying $\mathcal{O}_\Gamma^{\mathsf{w\text{-}CSA}}$. Note that the public keys at the nodes are all generated by random coins. After the queries are completed, $\mathcal{A}$ returns a forgery $(m^\star, \sigma^\star)$. $\sigma^\star$ contains at least one forgery of the public key at a node. The probability that $\sigma^\star$

contains a forgery of the target node is at least $\frac{q-1}{q^d-1}$. Thus, we have

$$\Pr[\mathbf{Exp}^{\mathsf{wCSA}}_{\Gamma',\mathcal{A}}(\lambda)] \leq Adv^{\text{Ind-PRF}}_{F,D^{\mathcal{A}}}(\lambda) + \frac{q^d-1}{q-1}\Pr[\mathbf{Exp}^{\mathsf{EU\text{-}wCSA}}_{\Gamma,\mathcal{R}^{\mathcal{A}}}(\lambda)], \tag{4.20}$$

which is negligible if F is a pseudorandom function and Γ is EU-wCSA. □

4.2.2 From wCSA-security to n-wCSA-security

Now, we present a construction for the EU-n-wCSA stateful signature scheme, and it is depicted in Figure 4.9. It is similar to the construction in Figure 4.8, where the tree has only 2 layers. The root contains a key pair of a q-samplable EU-wCSA stateful signature scheme. The root has q leaves, each of which contains a key pair of an n-time stateless signature scheme. The secret key of the root is used to sign the public keys of leaf nodes. Thus, a signature of Γ' consists of a signature of an n-time stateless signature scheme, the corresponding public key, and a stateful signature of this public key.

Theorem 18 *Let $\widetilde{\Gamma} = (\widetilde{\mathsf{KeyGen}}, \widetilde{\mathsf{Sign}}, \widetilde{\mathsf{Ver}})$ be an n-time EU-CMA stateless signature scheme in which the public keys are of length $l = l(\lambda)$. Let $\Gamma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be a q-samplable EU-wCSA stateful signature scheme in which the message space M_λ includes $\{0,1\}^{l(\lambda)}$, where q is polynomial with respect to λ. Let $F : [q] \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a pseudorandom function. Then, the $\Gamma' = (\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ depicted in Figure 4.9 is a q-samplable EU-n-wCSA stateful signature scheme.*

Proof. Γ' is q-samplable since $\mathsf{Samp}_q(pk, i) = i$ is a sampling algorithm. Next, we prove that Γ' is EU-n-wCSA. The first step is similar to those of Theorems 16 and 17. First, we replace the pseudorandom function used in Γ' with a truly random function. That is, KeyGen' additionally picks $c_0, ..., c_{q-1}$ randomly from $\{0,1\}^\lambda$ and contains them in sk'. In Sign', $F_{st}(sk')$ is replaced by the c_{st} contained in sk'. Suppose the success probability differs from that of the original EU-n-wCSA experiment against Γ'; we can construct a distinguisher D to break the pseudorandomness of F, as we did in previous proofs.

```
$\mathsf{KeyGen}'(1^\lambda)$
    $(pk, sk, *) \leftarrow \mathsf{KeyGen}(1^\lambda)$
    $s \xleftarrow{\$} \{0,1\}^\lambda$
    $pk' := pk, sk' := (pk, sk, s)$
    return $(pk', sk', 0)$

$\mathsf{Ver}'_{pk'}(m, \sigma')$
    Parse $\sigma' = (\sigma_1, \widetilde{pk}, \sigma_2)$
    $b_2 \leftarrow \mathsf{Ver}_{\widetilde{pk}}(m, \sigma_2)$
    $b_1 \leftarrow \mathsf{Ver}_{pk'}(\widetilde{pk}, \sigma_1)$
    return $b_1 \wedge b_2$

$\mathsf{Sig}'_{sk'}(m, st)$
    If $st \notin [q]$ return $(\perp, st)$
    Parse $sk' = (pk, sk, s)$
    $(\widetilde{pk}, \widetilde{sk}) \leftarrow \widetilde{\mathsf{KeyGen}}(1^\lambda; F_s(st))$
    $\sigma_2 \leftarrow \widetilde{\mathsf{Sign}}_{\widetilde{sk}}(m)$
    $(\sigma_1, *) \leftarrow \mathsf{Sign}_{sk}(\widetilde{pk}, \mathsf{Samp}_q(pk, st))$
    $\sigma' = (\sigma_2, \widetilde{pk}, \sigma_1)$
    return $(\sigma', st + 1)$
```

Figure 4.9: Construction of a q-samplable EU-n-wCSA stateful signature scheme.

Next, we show that if there exists an adversary $\mathcal{A}$ that succeeds in this experiment with nonnegligible probability, then there exists a reduction $\mathcal{R}^{\mathcal{A}}$ that can break the wCSA security of Γ or the CMA security of $\widetilde{\Gamma}$. First, $\mathcal{R}$ is given a challenge pk for the EU-wCSA experiment against Γ and a challenge $\widetilde{pk}$ for the EU-CMA experiment against $\widetilde{\Gamma}$. $\mathcal{R}$ randomly picks $c_0, ..., c_{q-1}$ from $\{0,1\}^\lambda$ and runs $(\widetilde{pk}_i, \widetilde{sk}_i) \leftarrow \widetilde{\mathsf{KeyGen}}(1^\lambda, c_i)$ for $i \in [q]$. After that, it randomly chooses $t \leftarrow [q]$ and replaces $\widetilde{pk}_t$ with $\widetilde{pk}$. Then, $\mathcal{R}$ runs $\mathcal{A}(1^\lambda, pk, 0)$. When $\mathcal{A}$ queries for the signature of (m, st), $\mathcal{R}$ checks whether $st \notin [q]$ or st has been queried more than n times. If so, it returns $(\perp, st)$. Otherwise, it runs $\sigma_2 \leftarrow \widetilde{\mathsf{Sign}}_{\widetilde{sk}_{st}}(m)$ when $st \neq t$ or $\sigma_2 \leftarrow \mathcal{O}^{\mathsf{CMA}}_{\widetilde{\Gamma}}(m)$ when $st = t$. Then, it runs $\sigma_1 \leftarrow \mathcal{O}^{\mathsf{wCSA}}_{\Gamma}(\widetilde{pk}_{st}, \mathsf{Samp}_q(pk, st))$. Finally, $\mathcal{R}$ answers $(\sigma_2, \widetilde{pk}_{st}, \sigma_1)$ to $\mathcal{A}$.

After the queries are completed, the adversary returns a forgery $(m^\star, \sigma^\star)$, where $\sigma^\star = (\sigma_2^\star, \widetilde{pk}^\star, \sigma_1^\star)$. If it is a valid forgery, let **NewPK** be the event that $\widetilde{pk}^\star \notin \{\widetilde{pk}_0, ..., \widetilde{pk}_{q-1}\}$.

- If **NewPK** occurs, this means that $\widetilde{pk}^\star$ is never input into $\mathcal{O}^{\mathsf{wCSA}}_{\Gamma}$. Furthermore, since $\mathsf{Ver}_{pk}(m^\star, \sigma^\star) = 1$, we have $\mathsf{Ver}_{pk}(\widetilde{pk}^\star, \sigma_1^\star) = 1$. Thus, $(\widetilde{pk}, \sigma_1^\star)$ is a forgery of Γ, and $\mathcal{R}$ breaks the wCSA security of Γ. The probability of this case is bounded by $\Pr[\mathbf{Exp}^{\mathsf{EU\text{-}wCSA}}_{\Gamma,\mathcal{R}}(\lambda)]$.

- If **NewPK** does not occur, this means that $\widetilde{pk}^\star = \widetilde{pk}_i$ for some $i \in [q]$. If $i = t$, $\mathcal{R}$ returns $(m^\star, \sigma_2^\star)$ as a forgery of $\widetilde{\Gamma}$, and it returns $fail$ otherwise. Since all the $\widetilde{pk}_i$s are generated by random coins, the probability that $i = t$ is $1/q$. Thus, $\mathcal{R}$ breaks the CMA security of $\widetilde{\Gamma}$ with probability $1/q$. The probability of this case is bounded by $q \cdot \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}CMA}}_{\widetilde{\Gamma},\mathcal{R}}(\lambda)]$.

To summarize, we have

$$\Pr[\mathbf{Exp}^{\mathsf{EU\text{-}}n\mathsf{\text{-}wCSA}}_{\Gamma',\mathcal{A}}(\lambda)] \le Adv^{\text{Ind-PRF}}_{F,D^{\mathcal{A}}}(\lambda) + \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}wCSA}}_{\Gamma,\mathcal{R}^{\mathcal{A}}}(\lambda)] + q \cdot \Pr[\mathbf{Exp}^{\mathsf{EU\text{-}CMA}}_{\widetilde{\Gamma},\mathcal{R}^{\mathcal{A}}}(\lambda)], \tag{4.21}$$

which is negligible if F is a pseudorandom function, Γ is EU-wCSA and $\widetilde{\Gamma}$ is EU-CMA. □

4.3 Generality of Our Frameworks

In the previous sections, we discussed constructions in a black-box manner. In this section, we generalize our frameworks to instantiations based on concrete assumptions. Indeed, our frameworks capture several existing constructions.

- **Hash-based Assumptions**.

  Since all the additional primitives used in our constructions, such as PRFs and PRGs, can be instantiated by hash functions, the most natural instantiations are HBS schemes. There are several hash-based one-time signature schemes, such as the Winternitz OTS (W-OTS) [87, 43], the Bleichenbacher-Maurer OTS [20], the Leighton-Micali OTS [83], BiBa [90] and W-OTS+ [67]. W-OTS+ is probably the most widely-used OTS. We can instantiate the accumulator in Figure 4.7 by using a Merkle tree. According to the constructions in Figures 4.7 and 4.8, any of the one-time signature schemes above can be converted to q-samplable EU-wCSCMA and q-time EU-KSCMA stateful signature schemes with large values of q.

Note that XMSS^{MT} [70] can be considered as an example of the above scheme. It uses the construction in Figure 4.7 from W-OTS+ to XMSS and then uses the construction in Figure 4.8 from XMSS to XMSS^{MT}. The slight difference between XMSS^{MT} and the construction in Figure 4.8 is that the signature of XMSS^{MT} does not contain the public keys in the path. For W-OTS+, a public key can be calculated from a message and its valid signature and thus is omitted in XMSS^{MT}.

In addition, according to the constructions in Figures 4.9 and 4.6, it is possible to construct a hash-based stateless signature scheme (with a large number of signing operations). In the construction in Figure 4.9, an n-time stateless signature scheme is needed, and it can be instantiated with hash-based few-time signature schemes, such as HORS [94], HORS++ [92], HORST [15], PORS [8], FORS [17] and DFORS [1]. HORS++ is based on the one-way property of hash functions and a cover-free family, and others are based on the one-way and hardness properties of subset resilience problems. By comparison, HORS++ needs a weaker assumption but yields larger signatures (although the running time is shorter).

SPHINCS [15] and its variants can be considered examples of combining the constructions in Figures 4.9 and 4.6. They use a highly samplable stateful signature scheme based on hash functions, such as XMSS^{MT}, and then combine it with different kinds of few-time stateless signature schemes. SPHINCS, Gravity-SPHINCS [9] and SPHINCS+ [17] choose HORST, PORS and FORS, respectively, as their n-time stateless signature schemes. Similar to WOTS+, the public key of the few-time signatures is omitted in SPHINCS/ Gravity-SPHINCS/ SPHINCS+, since it can be calculated from a message and its signature.

- **RSA** & **CDH** & **SIS Assumption**.

  Another efficient instantiation is based on the (standard, not strong) RSA assumption. There are several stateful signature schemes based on the RSA assumption; these include the Dwork-Noar scheme[46] and the Cramer-Damgård scheme[37]. They are EU-KSCMA based on our security notions. In detail, they both use a tree of depth d and out-degree l. The tree can be divided into two parts. The lower part contains the leaves of the tree, where each leaf is labeled by a message. The upper part contains the other nodes of the tree, where each node is labeled by a randomness value. Every time the scheme signs for a message, it picks the left-most unused leaf in the lower part, labels the leaf with the message, and yields the authentication path from the leaf to the root. As the upper part is determined during the signing operation and can be calculated in advance, it can be seen as a part of the secret key. Then, these schemes can be seen as stateful signature schemes whose states are the indices of the leaves that have been used. In this case, these schemes are also EU-wCSCMA and l^d-samplable.

In [21], the authors proposed a general construction for a stateless signature scheme from a tag-based signature scheme with EU-naCMA$^\star_n$ security, and it can be considered a non-adaptive version of the EU-n-wCSCMA defined in this chapter. Then, the constructions in [21] can be considered as examples of those in Figure 4.6.

Note that the construction in [21] works well only if (a) there exists an efficient non-adaptive-wCSA-secure stateful signature scheme with high samplability, (b) there exists an aggregation algorithm for signatures, and (c) there exists an efficient chameleon hash function. The authors gave examples based on the RSA, CDH and SIS assumptions. In comparison, HBS schemes do not satisfy the second and third conditions, so the construction in [21] fell short in terms of hash-based cryptography.

- **Other Assumptions**.

  There are some stateful signature schemes that fit our framework, such as the code-based one-time signature schemes proposed in [51, 91]. Unfortunately, few-time code-based signature schemes still lack research, especially those that are secure in the standard model. With our construction, we can transform a code-based few-time EU-CMA stateless signature scheme into a general stateless EU-CMA scheme in the standard model. It is still an open question whether our construction can be instantiated by schemes based on other assumptions, such as multivariate quadratic problems.

4.4 Specific Stateful Signature Schemes

Finally, we compare our definition with other stateful signature schemes mentioned in related work. For convenience, stateful signature schemes are abbreviated as SS in this subsection.

- **Key-Evolving Signature Scheme.**

  A key-evolving signature scheme (KES) consists of four algorithms: KeyGen, Sign, KeyUpd, and Ver. KeyGen generates a pair of keys (pk, sk). A signer who owns a secret key sk can sign a message m by running $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$. After that, the signer runs $sk' \leftarrow \mathsf{KeyUpd}(sk)$ to update the secret key with a new key. (In some definitions of a KES, the public key is also updated during this step. In this work, we only consider the case where the secret key is updated and the public key is static.) Anyone who owns the public key pk can verify the signature by running $\mathsf{Ver}_{pk}(m, \sigma)$. As long as the secret key is updated in a correct way, a signer can generate a valid signature. A KES can be seen as an SS if we let the state of the SS be the updated secret key of the KES and the secret key of the SS be empty, and Sign together with KeyUpd in the KES can be considered the signing algorithm of the SS. Thus, an SS is a more general object than a KES.

  Thus, the security of a KES can only hold under HSAs. It will never be secure under a KSA or CSA since the updated secret keys are given to the adversary in these cases. However, it is possible to convert an EU-CMA KES (that is, the EU-HSA in our definition) to an EU-wCSA signature scheme by the conversions in Figures 4.3 and 4.4.

Some cases of KESs are of forward security. In these cases, the signature contains the time slot when it was issued. Forward security implies that even if an adversary obtains the entire secret key at a time slot t, it cannot forge a valid signature with a time slot before t. It is meaningless to discuss forward security in our definition of SS, since the syntax of SS does not mention the time slot. The conversion of a EU-HSA/KSA/wCSA stateful signature scheme into a forward-secure KES in a black-box manner is still an open question.

- **Tag-based Signature Scheme.[21]**

  A tag-based signature scheme (TS) consists of three algorithms: KeyGen, Sign, and Ver. It is the same as a stateless signature scheme except that both Sign and Ver take as input an additional tag in a tag space. A tag has a similar role as that of a state in an SS. For a TS, the signing algorithm always outputs a valid signature when taking a secret key and a tag as inputs. However, for a general SS, the signing algorithm may output an invalid signature if the state is invalid. Thus, a TS can also be considered a samplable SS where all the states are valid.

  In [21], the authors defined EU-naCMA$^{\star}_{n}$ security for a TS, and this is very similar to our definition of EU-n-wCSA for an SS. An EU-naCMA$^{\star}_{m}$ adversary against a TS is allowed to query the signing oracle with any (m, tag), as long as the tag has not been queried more than n times. Namely, it is the same as an n-wCSA adversary in the case of an SS except that the messages are chosen non-adaptively. In addition, a TS does not update the tag or the keys when signing messages. Thus, it is meaningless to discuss HSA/KSA security for TSs. It is natural to convert it into an EU-KSA SS by using the construction in Figure 4.5 (if the TS is EU-wCSA).

  Furthermore, there are several specific tag-based signature schemes, such as double-authentication signature schemes [93], that can also be considered stateful signature schemes with special properties.

4.5 Conclusion and Open Questions

In this work, we formally define the syntax and security notions of stateful signature schemes and demonstrate the conversions among them. After that, we show the conversions from stateful signature schemes to stateless schemes. However, many conversions require an additional property of stateful signature schemes called q-samplability. Even though many stateful signature schemes indeed have this property, it is still a strong condition, as it does not hold for all stateful signature schemes. In this work, we attempt to solve this problem by presenting some general constructions of q-samplable stateful signature schemes from OTSs, but this is not efficient enough in practice. It is still an open question whether it is possible to remove samplability from the requirements of constructions or to replace it with a weaker property.

In addition, the construction in Theorem 6 is proven to be limited-time EU-CMA. Indeed, it is weaker than being general EU-CMA. In [97, 98], the author proves that two limited-time EU-CMA stateless signature schemes are also general EU-CMA by using a prefix-guessing strategy in the reductions. Since the two schemes have similar structures to ours, it is potential to implement this technique in our black-box construction. Unfortunately, it seems hard to directly obtain a positive result. It is probably because the prefix-guessing strategy behaves well in some special cases. It is an interesting open question whether we can improve our construction in Theorem 6 to general EU-CMA one. (This may require some additional property of the stateful signature schemes besides samplability.)

Finally, it is interesting to look for the relationship between stateful signature scheme and other specific signature schemes. For example, is it possible to construct a forward-secure key-evolving signature scheme from an EU-KSCMA or EU-wCSCMA stateful signature scheme? As far as we know, although the$f$ two schemes are closely related, there does not exist a black-box construction.

Chapter 5

Quantum-access Security of Stateless Hash-based Signature Schemes

5.1 Tweakable Hash Functions

In this chapter, we particularly focus on $H : \{0,1\}^* \to [t]^k$, that is, hash functions mapping to k (ordered) elements of set $[t]$ (where $t = 2^\tau$ for some integer τ). We can simply sample such a function by sampling a hash function $H' : \{0,1\}^* \to \{0,1\}^{k \cdot \log t}$, splitting the output into k short strings of length τ, and then mapping the short strings into integers in $[t]$. In other words, the above H and H' are two equivalent expressions of one hash function.

In the following, we denote a function $H : \{0,1\}^* \to [t]^k$ by $H = (h_1, ..., h_k)$, where $h_i : \{0,1\}^* \to [t]$ denotes the "partial" function mapping to the ith element of the output of H. To avoid ambiguity, we always use capital letters (such as H) to denote a hash function and their corresponding small letters with a subscript (such as h_i) as the partial functions. For instance, in the case that $\mathcal{H}$ is a hash function family mapping to $[t]^k$, $(h_1, ..., h_k) \leftarrow \mathcal{H}$ means sampling $H \leftarrow \mathcal{H}$ and letting $H = (h_1, ..., h_k)$, rather than sampling k functions from $\mathcal{H}$.

A tweakable hash function is a special hash function taking as input a message m together with a tweak T and a public parameter P. Especially, let $\mathcal{T}$ be the tweak space, $\mathcal{P}$ be the public parameter space and $\mathcal{M}$ be the message space, a tweakable hash function **Th** is defined as

$$\mathbf{Th} : \mathcal{P} \times \mathcal{T} \times \mathcal{M} \to \{0,1\}^n. \tag{5.1}$$

Definition 27 *(SM-TCR.[17]) Let* ***Th*** *be a tweakable hash function defined above. Let* $p \leq |\mathcal{T}|$*. For an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\mathcal{A}_1$ *is allowed to give* p *queries to an oracle* $\boldsymbol{Th}(P, \cdot, \cdot)$ *where* P *is uniformly chosen from* $\mathcal{P}$*. Denote the set of* $\mathcal{A}_1$*'s queries be* $Q = \{(T_i, M_i)\}_{i=1}^p$ *and define the predicate* $\mathsf{DIST}(\{T_i\}_{i=1}^p) = 1$ *iff all tweaks are distinct. Then,* $\mathcal{A}_2$ *takes as input* (Q, P) *and the state of* $\mathcal{A}_1$*, and finally outputs* (j, M)*. Define*

$$Adv_{\boldsymbol{Th},p,q}^{\mathsf{SM\text{-}TCR}}(\mathcal{A}) \triangleq \Pr_{P \leftarrow \mathcal{P}}[\boldsymbol{Th}(P, T_j, M_j) = \boldsymbol{Th}(P, T_j, M) \wedge M \neq M_j \wedge \mathsf{DIST}(\{T_i\}_{i=1}^p) = 1], \tag{5.2}$$

where q *denotes the maximum number of queries to* ***Th****.*

We say that ***Th*** *is single-function multi-target target-collision-resistant for distinct tweaks (SM-TCR) if for any polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* $\epsilon(\cdot)$ *such that* $Adv_{\boldsymbol{Th},p}^{\mathsf{SM\text{-}TCR}}(\mathcal{A}) \leq \epsilon(n)$ *for large enough* $n \in \mathbb{N}$*.*

Definition 28 *(SM-DSPR.[17]) Let* ***Th****,* DIST*,* Q *be as defined above. Denote a predicate* $SP_{P,T}(M) = 1$ *iff there exists another* $M' \neq M$ *such that* $\boldsymbol{Th}(P, T, M) = \boldsymbol{Th}(P, T, M')$*. For adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\mathcal{A}_1$ *is allowed to give* p *queries to an oracle* $\boldsymbol{Th}(P, \cdot, \cdot)$ *where* P *is uniformly chosen from* $\mathcal{P}$*. Then,* $\mathcal{A}_2$ *takes as input* (Q, P) *and the state of* $\mathcal{A}_1$*, and finally outputs* (j, b)*. Define*

$$succ \triangleq \Pr_{P \leftarrow \mathcal{P}}[\mathsf{SP}_{P,T_j}(M_j) = b \wedge \mathsf{DIST}(\{T_i\}_{i=1}^p)], \tag{5.3}$$

$$triv \triangleq \Pr_{P \leftarrow \mathcal{P}}[\mathsf{SP}_{P,T_j}(M_j) = 1 \wedge \mathsf{DIST}(\{T_i\}_{i=1}^p)] \tag{5.4}$$

and

$$Adv_{\boldsymbol{Th},p,q}^{\mathsf{SM\text{-}DSPR}}(\mathcal{A}) \triangleq \max\{0, succ - triv\}, \tag{5.5}$$

where q denotes the maximum of queries to $\boldsymbol{Th}$*.*

We say that $\boldsymbol{Th}$ *is single-function multi-target decisional second-preimage-resistant for distinct tweaks (SM-DSPR) if for any polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* $\epsilon(\cdot)$ *such that* $Adv^{\textsf{SM-DSPR}}_{\boldsymbol{Th},p}(\mathcal{A}) \leq \epsilon(n)$ *for large enough* $n \in \mathbb{N}$*.*

Then, we give a definition of preimage resistance for tweakable hash functions. It is a tweakable version of Open-PRE [16] and has been implicitly used in security proof of SPHINCS+[17].

Definition 29 *(SM-OpenPRE.) Let* $\boldsymbol{Th}$ *be a tweakable hash function defined above. Let* $p \leq |\mathcal{T}|$*. For an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\mathcal{A}_1$ *is allowed to give* p *queries to an oracle* $\mathcal{O}$ *initialized by a random* $P \in \mathcal{P}$*. Taking as input* $T_i \in \boldsymbol{T}$*,* $\mathcal{O}$ *randomly choose* $M_i \leftarrow \boldsymbol{M}$ *and output* $Y_i = \boldsymbol{Th}(P, T_i, M_i)$*. Define* $Q = \{(T_i, Y_i)\}_{i=1}^{p}$ *be the input/output pairs of* $\mathcal{O}$ *and* $\textsf{DIST}(\{T_i\}_{i=1}^{p})$ *as above. Then,* $\mathcal{A}_2$ *takes as input* (Q, P) *and the state of* $\mathcal{A}_1$*, and is given the access of an oracle* $\textsf{Open}(i) = M_i$*. Let* L *be the list of the queries to* $\textsf{Open}(\cdot)$*. Finally,* $\mathcal{A}_2$ *outputs* (j, M)*. Define*

$$Adv^{\textsf{SM-OpenPRE}}_{\boldsymbol{Th},p,q}(\mathcal{A}) = \Pr_{P \leftarrow \mathcal{P}}[\boldsymbol{Th}(P, T_j, M) = Y_j \wedge \textsf{DIST}(\{T_i\}_{i=1}^{p}) = 1 \wedge j \notin L], \quad (5.6)$$

where q denotes the maximum number of queries to $\boldsymbol{Th}$*.*

We say that $\boldsymbol{Th}$ *is SM-OpenPRE if for any polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* ϵ *such that* $Adv^{\textsf{SM-OpenPRE}}_{\boldsymbol{Th},p}(\mathcal{A}) \leq \epsilon(n)$ *for large enough* $n \in \mathbb{N}$*.*

The following lemma shows that the insecurity of SM-OpenPRE can be reduced to SM-TCR or SM-DSPR.

Lemma 17 *([16, 17]) Let* $\boldsymbol{Th}$ *be a tweakable hash function family and* $\mathcal{A}$ *be an adversary against SM-OpenPRE of* $\boldsymbol{Th}$*. Then, there exist polynomial-time reductions* $\mathcal{B}^{\mathcal{A}}$ *and* $\mathcal{C}^{\mathcal{A}}$ *such that*

$$Adv^{\textsf{SM-OpenPRE}}_{\boldsymbol{Th},p,q}(\mathcal{A}) \leq 3 \cdot Adv^{\textsf{SM-TCR}}_{\boldsymbol{Th},p,q}(\mathcal{B}) + Adv^{\textsf{SM-DSPR}}_{\boldsymbol{Th},p,q}(\mathcal{C}). \quad (5.7)$$

In particular, if the tweakable hash function **Th** is modeled as a random oracle (which means that $\mathbf{Th}(P, T, M) = H(P||T||M)$ where H is a quantum random oracle), the security has been evaluated in [17].

Lemma 18 *([16, 17]) Let **Th** be constructed above mapping to $\{0,1\}^n$. For any quantum adversary $\mathcal{A}$ making at most q queries to the quantum random oracle, it holds that*

$$Adv^{\textsf{SM-TCR}}_{\mathbf{Th},p,q}(\mathcal{A}) \leq 8(2q+1)^2/2^n. \tag{5.8}$$

$$Adv^{\textsf{SM-DSPR}}_{\mathbf{Th},p,q}(\mathcal{A}) \leq 32pq^2/2^n. \tag{5.9}$$

Remark 6 *It is conjectured that equation (5.9) is loose. In [17], the author conjectures that $Adv^{\textsf{SM-DSPR}}_{\mathbf{Th},p,q}(\mathcal{A})$ should be bounded by $O(q^2/2^n)$ (without the factor p).*

In practice, the hash function **Th** is sampled by choosing the public parameter $P \in \mathcal{P}$. Since P is public, we omit P and simply write $\mathbf{Th}(T, M)$ in the following.

An SM-TCR tweakable hash function can be used to construct a tree structure [17]. For convenience, we only show the syntax and the security notion and omit the detailed construction.

Proposition 1 *Let τ be an ingeter, $(y_1, ..., y_t)$ be a t-tuple of n-bit strings where $t = 2^\tau$ and $\mathbf{Th} : \{0,1\}^l \times \{0,1\}^* \to \{0,1\}^n$ be a tweakable hash function where $l \geq \tau + 1$. There exists a tuple of algorithms $\textsf{Tree} = (\textsf{TreeGen}, \textsf{TreeProv}, \textsf{TreeVer})$ where*

- *$\textsf{TreeGen}(\mathbf{Th}, t, (y_1, ..., y_t))$ generates a hash tree with leaf $(y_1, ..., y_t)$ and function **Th**, where the tweak of **Th** is the address of the node. Then, it outputs the root of the hash tree.*
- *$\textsf{TreeProv}(\mathbf{Th}, t, (y_1, ..., y_t), i)$ runs $\textsf{TreeGen}(\mathbf{Th}, t, (y_1, ..., y_t))$, obtains the hash tree, and outputs the authentication path π_i for leaf y_i.*

- *TreeVer*$(\boldsymbol{Th}, t, i, y_i, \pi_i)$ *uses y_i and its authentication path π_i to generate the root of the hash tree, and then outputs the root.*

Lemma 19 *Let $\boldsymbol{Th} : \{0,1\}^l \times \{0,1\}^* \to \{0,1\}^n$ and Tree be depicted above. For any $(y_1, ..., y_t)$ and adversary $\mathcal{A}$, there exists a reduction $\mathcal{M}^{\mathcal{A}}$ such that*

$$\Pr_{\boldsymbol{Th},\mathcal{A}}\left[\begin{array}{c|c} y^* \neq y_{i^*} & y_0 \leftarrow \textit{TreeGen}(\boldsymbol{Th}, t, (y_1, ..., y_t)) \\ y_0 = \textit{TreeVer}(\boldsymbol{Th}, t, i^*, y^*, \pi^*) & (i^*, y^*, \pi^*) \leftarrow \mathcal{A}(\boldsymbol{Th}, t, (y_1, ..., y_t)) \end{array}\right] \leq Adv^{\textit{SM-TCR}}_{\boldsymbol{Th},t,q_H}(\mathcal{M}^{\mathcal{A}}). \tag{5.10}$$

5.2 Revisiting Subset-resilient Hash Function Families

In this chapter, the definition of subset-resilient hash function families is slightly different from that in Chapter 3. The range of a hash function is $[t]$ instead of $\{0,1\}^{l(n)}$ where $t = 2^\tau$ for some τ. The new definitions are essentially equivalent to the previous ones but more concise in constructions.

In addition, this section proposes a new variant called weak subset resilience. It is a weaker security notion than restricted subset resilience and helpful to the following analysis.

5.2.1 Definitions

Definition 30 *(Subset Cover.[94]) Let $H = (h_1, ...h_k)$ be a hash function mapping $\{0,1\}^m$ to $[t]^k$ and $r \geq 0$ be an integer. We say that $(r+1)$-tuple $(x, x_1, ..., x_r) \in \{0,1\}^{m(r+1)}$ is an (r,k)-subset cover of H if it holds that $\{h_i(x)\}_{i\in[k]} \subset \{h_i(x_j)\}_{i\in[k],j\in[r]}$ and $x \notin \{x_j\}_{j\in[r]}$.*

Definition 31 *(Restricted Subset Cover.) Let $H = (h_1, ...h_k)$ be a hash function mapping $\{0,1\}^m$ to $[t]^k$ and $r \geq 0$ be an integer. We say that $(r+1)$-tuple $(x, x_1, ..., x_r) \in \{0,1\}^{m(r+1)}$ is an (r,k)-restricted subset cover of H if it holds that $x \notin \{x_j\}_{j\in[r]}$, and for $\forall i \in [k]$, $h_i(x) \in \{h_i(x_j)\}_{j\in[r]}$.*

Next, we show a weaker statement called *weak subset cover* that is useful in the following sections.

Definition 32 *(Weak Subset Cover.) Let $H = (h_1, ...h_k)$ be a hash function mapping $\{0,1\}^m$ to $[t]^k$ and $r \geq 0$ be an integer. We say an $(r+1)$-tuple $(x_1, ..., x_{r+1}) \in \{0,1\}^{m(r+1)}$ is an (r,k)-weak subset cover of H if for $\forall i \in [k]$, it holds that $|\{h_i(x_j)\}_{j\in[r+1]}| \leq r$ and x_j's are distinct. In other words, there exists a collision in $x_1, ..., x_{r+1}$ w.r.t. each h_i.*

Corollary 2 *If $(x, x_1, ..., x_r)$ is an (r,k)-restricted subset cover of H, it is also an (r,k)-weak subset cover of H.*

If it is hard for any polynomial-time adversary to find a (restricted/weak-)subset cover, then we say that the hash function family is (restricted/weak-)subset-resilient.

Definition 33 *Let $\mathcal{H} = \{H : \{0,1\}^m \to [t]^k\}$ be a hash function family. Let $\mathcal{A}$ be an adversary that takes as input $H = (h_1, ..., h_k) \leftarrow \mathcal{H}$, runs at most q hash queries and finally outputs $(x, x_1, ..., x_r) \in \{0,1\}^{m(r+1)}$. Define*

$$Adv_{\mathcal{H},q}^{(r,k)\text{-}SR}(\mathcal{A}) \triangleq \Pr_{\mathcal{H},\mathcal{A}}\left[\{h_i(x)\}_{i\in[k]} \subset \{h_i(x_j)\}_{i\in[k],j\in[r]} \wedge x \notin \{x_j\}_{j\in[r]}\right] \tag{5.11}$$

and

$$Adv_{\mathcal{H},q}^{(r,k)\text{-}rSR}(\mathcal{A}) \triangleq \Pr_{\mathcal{H},\mathcal{A}}\left[\forall i \in [k], h_i(x) \in \{h_i(x_j)\}_{j\in[r]} \wedge x \notin \{x_j\}_{j\in[r]}\right]. \tag{5.12}$$

Let $\mathcal{A}$ be an adversary that takes as input $H = (h_1, ..., h_k) \leftarrow \mathcal{H}$, runs at most q hash queries and finally outputs $(x_1, ..., x_{r+1}) \in \{0,1\}^{m(r+1)}$. Define

$$Adv_{\mathcal{H},q}^{(r,k)\text{-}wSR}(\mathcal{A}) \triangleq \Pr_{\mathcal{H},\mathcal{A}}\left[\forall i \in [k], \left|\{h_i(x_j)\}_{j\in[r+1]}\right| \leq r \wedge x_1, x_2, ..., x_{r+1} \text{ are distinct}\right]. \tag{5.13}$$

We say that $\mathcal{H}$ is a secure (r,k)(-restricted/weak) subset resilient hash function family or (r,k)-SRH(/rSRH/wSRH) if $Adv_{\mathcal{H},q}^{(r,k)\text{-}SR}(\mathcal{A})/Adv_{\mathcal{H},q}^{(r,k)\text{-}rSR}(\mathcal{A})/Adv_{\mathcal{H},q}^{(r,k)\text{-}wSR}(\mathcal{A})$ is negligible for any probabilistic polynomial-time quantum adversary $\mathcal{A}$.

5.2.2 Generic Security with Quantum Queries

This subsection gives generic security of subset resilience and its variants with quantum queries.

Theorem 19 *Let $\mathcal{H} = \{H : \{0,1\}^m \to [t]^k\}$ be a random function family and r be a positive integer. For any quantum probabilistic polynomial-time adversary $\mathcal{A}$, it holds that*

$$Adv_{\mathcal{H},q}^{(r,k)\text{-}wSR}(\mathcal{A}) \leq (2q+1)^{2(r+1)}\left(\frac{r^2}{t}\right)^k, \tag{5.14}$$

$$Adv_{\mathcal{H},q}^{(r,k)\text{-}rSR}(\mathcal{A}) \leq (2q+1)^{2(r+1)}(r+1)\left(\frac{r}{t}\right)^k, \tag{5.15}$$

and

$$Adv_{\mathcal{H},q}^{(r,k)\text{-}SR}(\mathcal{A}) \leq (2q+1)^{2(r+1)}(r+1)\left(\frac{rk}{t}\right)^k. \tag{5.16}$$

Proof. Since $\mathcal{H}$ is a random function family, the success probability of adversaries can be evaluated by Lemma 2. For $i \in [k]$, denote by $y^{(i)} \in [t]$ the i-th element of $y \in [t]^k$.

1. (Proof of (5.14).)

   For $H = (h_1, ..., h_k) : \{0,1\}^* \to [t]^k$, define $R_1 \subseteq [t]^{k(r+1)}$ as follows:

   $$R_1 \triangleq \{(y_1, y_2, ..., y_{r+1}) : \forall i \in [k], \left|\{y_j^{(i)}\}_{j\in[r+1]}\right| \leq r\}. \tag{5.17}$$

   We analyze the size of R_1. From $(y_1, ..., y_{r+1}) \in R_1$, for every $i \in [k]$, (at least) two of $y_1^{(i)}, ..., y_{r+1}^{(i)}$ are equal. Fix an $i \in [k]$, we can tranverse all the possible $(y_1^{(i)}, ..., y_{r+1}^{(i)})$ w.r.t. i as follows:

   (a) Pick a pair of indices a_1, a_2 from $[r]$.

   (b) Pick $y \in [t]$, let $y_{a_1} = y_{a_2} = y$.

   (c) Tranverse all possible values of $y_j^{(i)}$ for all $j \notin \{a_1, a_2\}$ and $j \in [r+1]$.

The numbers of choices in the three steps are $\binom{r+1}{2}$, t and t^{r-1} respectively. Thus, for all $i \in [k]$, the total number of possible values of $(y_1, ..., y_{r+1}) \in R_1$ is at most

$$\left(\binom{r+1}{2} \cdot t \cdot t^{r-1}\right)^k = \left(\frac{(r+1)r}{2} t^r\right)^k \leq (r^2 t^r)^k. \tag{5.18}$$

In addition, it is not hard to see that relation R_1 is not ordered (which means that for any $\pi \in \mathsf{Perm}([r+1])$, the statement $(y_1, ..., y_{r+1}) \in R_1$ is equivalent to the statement $(y_{\pi(1)}, ..., y_{\pi(r+1)})$). Due to Lemma 2, we have

$$\mathrm{Adv}_{\mathcal{H},q}^{(r,k)\text{-wSR}}(\mathcal{A}) \leq (2q+1)^{2(r+1)} \frac{(r^2 t^r)^k}{t^{(r+1)k}} = (2q+1)^{2(r+1)} \left(\frac{r^2}{t}\right)^k, \tag{5.19}$$

which is what we expected.

2. (Proof of (5.15).)

   Note that in Lemma 2, the elements in a solution have to be distinct, but those in a restricted subset cover do not. (In a restricted subset cover, only $x \notin \{x_j\}_{j\in[r]}$ is demanded, and thus x_j can be equal to another $x_{j'}$.) We divide a restricted subset cover into several cases.

   Fix r, $\mathcal{H}$ and $\mathcal{A}$. Recall that $H = (h_1, ..., h_k) \leftarrow \mathcal{H}$ and $(x, x_1, ..., x_r) \leftarrow \mathcal{A}(H)$. Let $1 \leq s \leq r$ be some integer. Define

$$f(s) \triangleq \Pr_{\mathcal{H},\mathcal{A}}\left[\forall i \in [k], h_i(x) \in \{h_i(x_j)\}_{j\in[r]} \wedge x \notin \{x_j\}_{j\in[r]} \wedge \left|\{x_j\}_{j\in[r]}\right| = s\right]. \tag{5.20}$$

   Then, we have

$$\mathrm{Adv}_{\mathcal{H},q}^{(r,k)\text{-rSR}}(\mathcal{A}) = \sum_{s\in[r]} f(s), \tag{5.21}$$

   and

$$f(r) = \Pr_{\mathcal{H},\mathcal{A}}\left[\{h_i(x)\}_{i\in[k]} \subset \{h_i(x_j)\}_{i\in[k],j\in[r]} \wedge x, x_1, ..., x_r \text{ are distinct}\right]. \tag{5.22}$$

First, we give a bound for the case that $r = s$.

Lemma 20 $f(r) \leq (2q+1)^{2(r+1)}(r+1)(\frac{r}{t})^k$.

Proof. For $H = (h_1, ..., h_k) : \{0,1\}^* \to [t]^k$, define $R_2 \subseteq [t]^{k(r+1)}$ as follows:

$$R_2 \triangleq \big\{(y, y_1, ..., y_r) : \forall i \in [k], y^{(i)} \in \{y_j^{(i)}\}_{j\in[r]}\big\}. \quad (5.23)$$

Next, we analyze the size of R_2. For convenience, we call the first element of $(y, y_1, ..., y_r) \in R_2$ as y_0.

First, there are exactly t^k possible values of y_0. Then, for any fixed $y_0 = (y_0^{(1)}, ..., y_0^{(k)})$, it holds that $y_0^{(i)} \in \{y_j^{(i)}\}_{j\in[r]}$ for each $i \in [k]$. This implies that $y_j^{(i)} = y_0^{(i)}$ for some $j \in [r]$. We can tranverse all the possible value of $(y_1^{(i)}, ..., y_r^{(i)})$ for each i w.r.t. y_0 by the following steps:

(a) Pick $j \in [r]$ and let $y_j^{(i)} = y_0^{(i)}$.

(b) Tranversing all the possible value of $y_{j'}^{(i)}$ for all $j' \in [r]$ and $j' \neq j$.

The number of possible values of $(y_1^{(i)}, ..., y_r^{(i)})$ w.r.t. y_0 is at most $r \cdot t^{r-1}$ for each i. Thus, considering all $i \in [k]$ and tranversing all possible values of y_0, the total number of $(y_0, y_1, ..., y_r)$ is at most

$$(r \cdot t^{r-1})^k \cdot t^k = (rt^r)^k. \quad (5.24)$$

Unlike R_1, relation R_2 is ordered. Define

$$R_2^* \triangleq \{(y_1, ..., y_{r+1}) : \exists \pi \in \mathsf{Perm}([r+1]) \text{ s.t. } (y_{\pi(1)}, ..., y_{\pi(r+1)}) \in R_2\}. \quad (5.25)$$

Observe that for any $\pi \in \mathsf{Perm}([r])$, the statement $(y, y_1, ..., y_r) \in R_2$ is equivalent to the statement $(y, y_{(\pi(1))}, ..., y_{\pi(r)}) \in R_2$. This implies that the order of R_2 is only determined by the first element. Thus, we can tranverse all the possible values of $(y_1, ..., y_{r+1}) \in R_2^*$ by the following steps:

(a) Pick $(y, y_1, ..., y_r) \in R_2$.

(b) Pick $j \in [r+1]$, and insert y between $(j-1)$-th element and the j-th element of $(y_1, ..., y_r)$. In other words, tranverse $(y, y_1, ...y_r)$, $(y_1, y, y_2, ..., y_r)$, ..., $(y_1, ..., y_r, y)$.

Thus, we have

$$|R_2^*| \leq (r+1)|R_2| \leq (r+1)(rt^r)^k. \tag{5.26}$$

Due to Lemma 2, we have

$$f(r) \leq (2q+1)^{2(r+1)} \frac{(r+1)(rt^r)^k}{t^{(r+1)k}} = (2q+1)^{2(r+1)}(r+1)\left(\frac{r}{t}\right)^k. \tag{5.27}$$

□

Next, we consider the case that $s < r$.

If the adversary output an (r,k)-restricted subset cover $(x, x_1, ..., x_r)$ such that $|\{x_j\}_{j\in[r]}| = s < r$. Let $\{x_j\}_{j\in[r]} = \{x'_{j'}\}_{j'\in[s]}$ after reordering. Then, it is not hard to see that $(x, x'_1, ..., x'_s)$ is an (s,k)-restricted cover and all the elements are distinct. The probability of this event is also bounded by Lemma 20. That is, for all $1 \leq s \leq r$ it holds that

$$f(s) \leq (2q+1)^{2(s+1)}(s+1)\left(\frac{s}{t}\right)^k \leq (2q+1)^{2(s+1)}(r+1)\left(\frac{r}{t}\right)^k. \tag{5.28}$$

Thus, we have

$$\mathrm{Adv}_{\mathcal{H},q}^{(r,k)\text{-}\mathsf{rSR}}(\mathcal{A}) = \sum_{s\in[r]}(2q+1)^{2(s+1)}(r+1)\left(\frac{r}{t}\right)^k \leq (2q+1)^{2(r+1)}(2r+2)\left(\frac{r}{t}\right)^k. \tag{5.29}$$

3. (Proof of (5.16).)

   Similarly, fix $r, \mathcal{H}, \mathcal{A}$. For $s \in [r]$, we define

$$g(s) \triangleq \Pr_{\mathcal{H},\mathcal{A}}\left[x \notin \{x_1, ..., x_r\} \wedge \{h_i(x)\}_{i\in[k]} \subset \{h_i(x_j)\}_{i\in[k],j\in[r]} \wedge \left|\{x_j\}_{j\in[r]}\right| = s\right], \tag{5.30}$$

   and thus

$$\mathrm{Adv}_{\mathcal{H},q}^{(r,k)\text{-}\mathsf{SR}}(\mathcal{A}) = \sum_{s\in[r]} g(s). \tag{5.31}$$

   As above, we first consider the case that $s = r$.

For $H = (h_1, ..., h_k) : \{0,1\}^* \to [t]^k$, define $R_3 \subseteq [t]^{k(r+1)}$ as follows:

$$R_3 \triangleq \{(y, y_1, ..., y_r) : \{y^{(i)}\}_{i\in[k]} \subseteq \{y_j^{(i)}\}_{i\in[k],j\in[r]}\}. \tag{5.32}$$

We divide R_3 into k subsets $R_{3,1}, ..., R_{3,k}$, where, for $m \in [k]$,

$$R_{3,m} \triangleq \{((y, y_1, ..., y_r) \in R_3 : \left|\{y^{(i)}\}_{i\in[k]}\right| = m\}. \tag{5.33}$$

Observe that $R_{3,m}$'s are disjoint and that $R_3 = \bigcup_{m\in[k]} R_{3,m}$. More precisely, the statement $(y, y_1, ..., y_r) \in R_{3,m}$ implies that $\{y^{(i)}\}_{i\in[k]}$ contains exactly m elements in $[t]$, and $\{y_j^{(i)}\}_{i\in[k],j\in[r]}$ covers them. Since there are at most rk elements in set $\{y_j^{(i)}\}_{i\in[k],j\in[r]}$, there are rk "chances" to cover the m target elements. We can tranverse all the elements of $R_{3,m}$ by the following steps:

(a) Pick m distinct elements $x_1, ..., x_m$ from $[t]$. Let $X = \{x_1, ..., x_m\}$.

The number of choices in this step is $\binom{t}{m}$.

(b) Pick $y^{(1)}, ..., y^{(k)}$ from X^k such that $\{y^{(i)}\}_{i\in[k]} = X$.

This step is equivalent to the experiment of putting k different balls into m different bins such that there is at least one ball in each bin. The number of choices is $\left\{ {k \atop m} \right\} \cdot m!$, where $\left\{ {k \atop m} \right\}$ denotes Stirling number of the second kind.

(c) Next, we require that $\{y_j^{(i)}\}_{i\in[k],j\in[r]}$ covers $X = \{y^{(i)}\}_{i\in[k]}$. Since $|X| = m$, we only need to choose m elements of $\{y_j^{(i)}\}_{i\in[k],j\in[r]}$, make them equal to a permutation of X, and do not have any demand for the remaining $(rk - m)$ elements.

The number of choices in the two substeps are $\binom{rk}{m}$ and $m!$ respectively.

(d) Finally, since the remaining $(rk-m)$ elements have no demand, tranverse all the possible $y_j^{(i)}$ that have not been assigned in the above steps.

The number of choices in this step is $t^{(rk-m)}$.

To sum up, the total number of elements in $R_{3,m}$ is

$$\begin{aligned}|R_{3,m}| =&\binom{t}{m}\left\{{k \atop m}\right\} \cdot m! \cdot \binom{rk}{m} \cdot m! \cdot t^{rk-m}\\ \leq&\frac{t^m}{m!} \cdot \left\{{k \atop m}\right\} \cdot m! \cdot \binom{rk}{m} \cdot m! \cdot t^{rk-m}\\ =&\left\{{k \atop m}\right\} \cdot \binom{rk}{m} \cdot m! \cdot t^{rk}\\ =&\left\{{k \atop m}\right\} \cdot (rk)_m \cdot t^{rk},\end{aligned}$$

where $(\cdot)_m$ denotes the falling factorial:

$$(x)_m = x \cdot (x-1) \cdot \ldots \cdot (x-m+1). \tag{5.34}$$

Thus, we have

$$|R_3| = \sum_{m=1}^{k} |R_{3,m}| \leq \sum_{m=1}^{k} \left\{{k \atop m}\right\} \cdot (rk)_m \cdot t^{rk} = (rk)^k \cdot t^{rk}, \tag{5.35}$$

where the last equality uses the fact that $\sum_{m=1}^{k} \left\{{k \atop m}\right\}(x)_m = x^k$.

Similar to R_2, we define

$$R_3^* \triangleq \{(y_1, \ldots, y_{r+1}) : \exists \pi \in \mathsf{Perm}([r+1]) \text{ s.t. } (y_{\pi(1)}, \ldots, y_{\pi(r+1)}) \in R_3\}, \tag{5.36}$$

and then we have

$$|R_3^*| \leq (r+1)|R_3| \leq (r+1)(rk)^k \cdot t^{rk}. \tag{5.37}$$

Due to Lemma 2, we have

$$g(r) \leq (2q+1)^{2(r+1)} \frac{(r+1)(rk)^k \cdot t^{rk}}{t^{(r+1)k}} = (2q+1)^{2(r+1)}(r+1)\left(\frac{rk}{t}\right)^k, \tag{5.38}$$

and for $s \in [r]$,

$$g(s) \leq (2q+1)^{2(s+1)}(s+1)\left(\frac{sk}{t}\right)^k \leq (2q+1)^{2(s+1)}(r+1)\left(\frac{rk}{t}\right)^k. \tag{5.39}$$

Thus,

$$\mathrm{Adv}_{\mathcal{H},q}^{(r,k)\text{-}\mathsf{SR}}(\mathcal{A}) = \sum_{s\in[r]}(2q+1)^{2(s+1)}(r+1)\left(\frac{rk}{t}\right)^k \leq (2q+1)^{2(r+1)}(2r+2)\left(\frac{rk}{t}\right)^k. \tag{5.40}$$

□

5.2.3 Target Subset Resilience

Target subset resilience (TSR) [94] is a variant of subset resilience. In (r,k)-TSR experiment, the adversary is given a hash function $H = (h_1, ..., h_k)$ and r random targets $x_1, ...x_r$. Then, the adversary is required to output a single element x such that $\{h_i(x)\}_{i\in[k]} \subset \{h_i(x_j)\}_{i\in[k],j\in[r]}$. It is not hard to see that (r,k)-TSR is a weaker notion than (r,k)-SR.

This subsection proposes a target version of restricted subset resilience, which is called *extended target subset resilience* (eTSR). Unlike TSR, the adversary in eTSR can adaptively control the target to some extent. In detail, the adversary is able to adaptively query a (classical) oracle Box. For a query x_j, Box randomly chooses $z_j \in \{0,1\}^n$ and returns $(z_j, H(z_j||x_j))$. After r queries, the adversary is required to output (x,z) such that for each $i \in [k]$, $h_i(z||x) \in \{h_i(z_j||x_j)\}_{j\in[r]}$ and $(x,z) \notin \{(x_j,z_j)\}_{j\in[r]}$ hold. Note that (r,k)-eTSR is a weaker notion than than (r,k)-rSR.

Definition 34 *(Extended Target Subset Resilience.) Let $\mathcal{H} = \{H = (h_1, ..., h_k) : \{0,1\}^{m+n} \to [t]^k\}$ be a hash function family. Let $\mathcal{A}^{\mathsf{Box}}$ be an adversary that queries Box at most r times, computes H at most q times and then outputs $(x,z) \in \{0,1\}^{m+n}$. Define*

$$Adv_{\mathcal{H},q}^{(r,k)\text{-}eTSR}(\mathcal{A}) \triangleq \Pr_{Box,\mathcal{H},\mathcal{A}}\left[\forall i \in [k], h_i(z||x) \in \{h_i(z_j||x_j)\}_{j\in[r]} \land (x,z) \notin \{(x_j,z_j)\}_{j\in[r]}\right]. \tag{5.41}$$

We say that hash function family $\mathcal{H}$ is an (r,k)-extended target-subset-resilient hash function family ((r,k)-eTSRH) if $Adv_{\mathcal{H},q}^{(r,k)\text{-}eTSR}(\mathcal{A})$ is negligible for any probabilistic polynomial-time quantum adversary $\mathcal{A}$.

Here we model $\mathcal{H}$ as a quantum-accessible $\mathsf{H} : \{0,1\}^{m+n} \to [t]^k$ and $\mathsf{h}_1, ..., \mathsf{h}_k : \{0,1\}^{m+n} \to [t]$ be the partial oracle. Then, the experiment of eTSR is depicted as **Game 0** in Figure 5.1.

```
Game 0
$(x, z) \leftarrow \mathcal{A}^{|\mathsf{H}\rangle,\mathsf{Box}}()$
if $\forall i \in [k], \mathsf{h}_i(z||x) \in \{\mathsf{h}_i(z_j||x_j)\}_{j\in[r]}$
    if $(x, z) \notin \{(x_j, z_j)\}_{j\in[r]}$ return 1
return 0

Box($x_j$)
$z_j \leftarrow \{0, 1\}^n$
return $(z_j, \mathsf{H}(z_j||x_j))$

Game 1
$(x, z) \leftarrow \mathcal{A}^{|\mathsf{H}\rangle,\mathsf{Box'}}()$
if $\forall i \in [k], \mathsf{h}_i(z||x) \in \{y_j^{(i)}\}_{j\in[r]}$
    if $(x, z) \notin \{(x_j, z_j)\}_{j\in[r]}$ return 1
return 0

Box'($x_j$)
$z_j \leftarrow \{0, 1\}^n$
$y_j := y_j^{(1)}||...||y_j^{(k)} \leftarrow [t]^k$
$\mathsf{H} := \mathsf{H}^{z_j||x_j \to y_j}$
return $(z_j, y_j)$
```

Figure 5.1: Hybrid arguments in the proof of Theorem 20.

Theorem 20 *Let $\mathcal{H}$ be modeled as a quantum-accessible random oracle H and r be a positive integer. For any quantum probabilistic polynomial-time adversary $\mathcal{A}$, it holds that*

$$Adv_{H,q}^{(r,k)\text{-}eTSR}(\mathcal{A}) \leq \frac{3r}{2}\sqrt{\frac{q+r+1}{2^n}} + 8(q+r+2)^2\left(\frac{r}{t}\right)^k. \tag{5.42}$$

Proof. We use the technique in security proof of (multi-target) extended collision resistance in [72] and [59]. We prove this theorem by showing the following games:

- **Game 0** is the original experiment of eTSR.
- **Game 1** differs from **Game 0** in that Box is replaced by Box'. Every time $\mathcal{A}$ queries x_j to the oracle Box', Box' randomly picks z_j, randomly chooses $y_j^{(1)}, ..., y_j^{(k)} \in [t]$ and reprograms $\mathsf{h}_i(z_j||x_j) := y_j^{(i)}$ for each $i \in [k]$. (In other words, it reprograms $\mathsf{H}(z_j||x_j) := y_j = y_j^{(1)}||...||y_j^{(k)}$.) Then, it returns (z_j, y_j) to the adversary. See details in Figure 5.1.

  Next, we show that the probabilities of **Game 0** and **Game 1** are negligible close by Lemma 1. If not, the adversary $\mathcal{A}$ can be used to distinguish

Repro_0 and Repro_1 in Figure 2.1. In **Game 0**, H is simulated by O_0 and Box is simulated by $\mathsf{Reprogram}_0$ with additional classical query to O_0. In **Game 1**, Box' is simulated by $\mathsf{Reprogram}_1$ with additional classical query to O_1. In total, it issues $(q+r+1)$ queries to O_b (q for simulating H, r for simulating Box/Box' and 1 for the final verification). Due to Lemma 1, we have

$$|\Pr[\textbf{Game 0}] - \Pr[\textbf{Game 1}]| \le \frac{3r}{2}\sqrt{\frac{q+r+1}{2^n}}. \tag{5.43}$$

- In **Game 1**, the adversary outputs (x, z) such that for all $i \in [k]$, $\mathsf{h}_i(z||x)$ is covered by $\{y_j^{(i)}\}_{j\in[r]}$. Define $S = \{y_{a_1}^{(1)}||...||y_{a_k}^{(k)}\}_{(a_1,...,a_k)\in[r]^k}$. In other words, S contains all $y = y^{(1)}||...||y^{(k)}$ where $y^{(i)} \in \{y_j^{(i)}\}_{j\in[r]}$ for each i. Thus, the adversary is to output (x, z) such that $H(z||x) \in S$ and (x, z) is not equal to any (x_j, z_j).

  Note that $|S| \le r^k$. Without loss of generality, we suppose $|S| = r^k$. (If $|S| < r^k$, the success probability is obviously smaller. Here our purpose is to find an upper bound of the probability.) Reorder $S = \{y'_1, ..., y'_{r^k}\}$.

  Next, we use an adversary succeeding in **Game 1** to construct a reduction breaking $\mathsf{Avg\text{-}Search}_\lambda$ in Lemma 5.

  Let $f \leftarrow D_\lambda : \{0,1\}^{m+n} \to \{0,1\}$ and $\lambda = (\frac{r}{t})^k$. Let $I : \{0,1\}^{m+n} \to [r^k]$ and $g : \{0,1\}^{m+n} \to [t]^k \backslash S$ be random functions. Construct $\tilde{\mathsf{H}} : \{0,1\}^{m+n} \to [t]^k$ as follows: for any $(z||x) \in \{0,1\}^{m+n}$, define:

$$\tilde{\mathsf{H}}(z||x) = \begin{cases} y_j, & x = x_j \wedge z = z_j, \\ y'_{I(z||x)}, & f(z||x) = 1, \\ g(x), & otherwise. \end{cases} \tag{5.44}$$

  Note that the outputs of $\tilde{\mathsf{H}}$ distributes uniformly. Thus, an adversary in **Game 2** finds (x, z) such that $f(z||x) = 1$. Due to Lemma 5, we have

$$\Pr[\textbf{Game 1}] \le 8(q+r+1+1)^2\left(\frac{r}{t}\right)^k = 8(q+r+2)\left(\frac{r}{t}\right)^k. \tag{5.45}$$

From equation (5.43) and (5.45), we complete the proof. □

Interleaved target subset resilience (ITSR) [17] is another variant of target resilience and has been used in constructing SPHINCS+. It can be considered an extended version of eTSR. We introduce another hash function $h_0 : \{0,1\}^{m+n} \to \{0,1\}^h$ (where h is a constant) and modify the oracle Box to an interleaved version iBox. Given x_j as input, iBox picks random z_j and outputs $(z_j, h_0(z_j||x_j), H(z_j||x_j))$. After queries, the adversary outputs (x,z) that, for each $i \in [k]$, $(h_i(z||x), h_0(z||x)) \in \{(h_i(z_j||x_j), h_0(z_j||x_j))\}_{j\in[r]}$ and $(x,z) \notin \{(x_j,z_j)\}_{j\in[r]}$ hold. Note that if $h = 0$, ITSR becomes the same as eTSR.

Definition 35 *(Interleaved Target Subset Resilience.) Let $\mathcal{H} = \{H = (h_1,...,h_k) : \{0,1\}^{m+n} \to [t]^k\}$ and $\mathcal{H}_0 = \{h_0 : \{0,1\}^{m+n} \to \{0,1\}^h\}$ be hash function families. Let $\mathcal{A}^{\mathsf{iBox}}$ be an adversary that queries iBox at most r times, computes (H,h_0) at most q times and then outputs $(x,z) \in \{0,1\}^{m+n}$. Define*

$$Adv^{(r,k)\text{-}ITSR}_{\mathcal{H},\mathcal{H}_0,q}(\mathcal{A}) \triangleq \Pr_{iBox,\mathcal{H},\mathcal{H}_0,\mathcal{A}}\left[\begin{array}{c}\forall i \in [k], (h_i(z||x), h_0(z||x)) \in \{(h_i(z_j||x_j), h_0(z_j||x_j))\}_{j\in[r]} \\ (x,z) \notin \{(x_j,z_j)\}_{j\in[r]}\end{array}\right]. \tag{5.46}$$

We say that hash function family pair $(\mathcal{H},\mathcal{H}_0)$ is an (r,k)-interleaved target subset resilient hash function family ((r,k)-ITSRH) if $Adv^{(r,k)\text{-}ITSR}_{\mathcal{H},\mathcal{H}_0,q}(\mathcal{A})$ is negligible for any probabilistic polynomial-time quantum adversary $\mathcal{A}$.

Remark 7 *In practice (e.g., in SPHINCS+[17]), $\mathcal{H}_0$ and $\mathcal{H}$ are instantiated by a separation of a single hash function family. That is, pick a hash function H_{msg} mapping to $\{0,1\}^{h+k\log t}$ and let $H_{msg}(z||x) = (MD||idx)$. H denotes the map from $(z||x)$ to MD and h_0 denotes the map to idx.*

In [17], the authors give an attack on ITSR and conjecture a bound for the security. Here we give a concrete lower bound of the security in the quantum-accessible random oracle model. The idea mainly follows [17] except using Lemma 1.

```
**Game 0**
$(x, z) \leftarrow \mathcal{A}^{|\mathsf{h}_0\rangle, |\mathsf{H}\rangle, \mathsf{iBox}}()$
**if** $\forall i \in [k], (\mathsf{h}_0(z||x), \mathsf{h}_i(z||x)) \in \{(\mathsf{h}_0(z||x), \mathsf{h}_i(z_j||x_j))\}_{j \in [r]}$
  **if** $(x, z) \notin \{(x_j, z_j)\}_{j \in [r]}$ **return** 1
**return** 0

$\mathsf{Box}(x_j)$
$z_j \leftarrow \{0,1\}^n$
**return** $(z_j, \mathsf{h}_0(z_j||x_j), \mathsf{H}(z_j||x_j))$

**Game 1**
$(x, z) \leftarrow \mathcal{A}^{|\mathsf{h}_0\rangle, |\mathsf{H}\rangle, \mathsf{iBox'}}()$
**if** $\forall i \in [k], (\mathsf{h}_0(z||x), \mathsf{h}_i(z||x)) \in \{(y_j^{(0)}, y_j^{(i)})\}_{j \in [r]}$
  **if** $(x, z) \notin \{(x_j, z_j)\}_{j \in [r]}$ **return** 1
**return** 0

$\mathsf{iBox'}(x_j)$
$z_j \leftarrow \{0,1\}^n$
**for** $\forall i \in [k], y_j^{(i)} \leftarrow [t], y_j^{(0)} \leftarrow \{0,1\}^h$
$y_j := y_j^{(1)}||...||y_j^{(k)}$
$\mathsf{H} := \mathsf{H}^{z_j||x_j \to y_j}, \mathsf{h}_0 := \mathsf{h}_0^{z_j||x_j \to y_j^{(0)}}$
**return** $(z_j, y_j^{(0)}, y_j)$
```

Figure 5.2: Hybrid arguments in the proof of Theorem 21.

Theorem 21 *Let H and h_0 be modeled as quantum-accessible random oracles H and h_0 respectively and r be a positive integer. For any quantum probabilistic polynomial-time adversary $\mathcal{A}$, it holds that*

$$Adv_{\mathsf{H},\mathsf{h}_0,q}^{(r,k)\text{-}ITSR}(\mathcal{A}) \le \frac{3r}{2}\sqrt{\frac{q+r+1}{2^n}} + 8(q+r+2)^2\Gamma, \tag{5.47}$$

where $\Gamma = \sum_\gamma \left(1-\left(1-\frac{1}{t}\right)^\gamma\right)^k \binom{r}{\gamma}\left(1-\frac{1}{2^h}\right)^{r-\gamma}\frac{1}{2^{h\gamma}}$.

Proof. We prove the statement by the following games, which is very similar to Theorem 20. We write $\mathsf{H}'(z||x) = (\mathsf{H}(z||x), \mathsf{h}_0(z||x))$.

- **Game 0** is the original experiment of ITSR as depicted in Figure 5.2.
- **Game 1** differs from **Game 0** expect that iBox is replaced by iBox' in Figure 5.2. Every time iBox' is queried with x_j, it randomly samples $y_j^{(0)}$,...,$y_j^{(k)}$ where $y_j^{(0)} \in \{0,1\}^h$ and $y_j^{(i)} \in [t]$ for other $i \in [k]$, and does reprogramming. As in Theorem 20, we have

$$|\Pr[\textbf{Game 0}] - \Pr[\textbf{Game 1}]| \le \frac{3r}{2}\sqrt{\frac{q+r+1}{2^n}}. \tag{5.48}$$

- We give a bound for the probability of **Game 1**. Let $\mathbf{y} = (y_j^{(0)},...,y_j^{(k)})_{j\in[r]}$ and $\mathbf{z} = (z_j)_{j\in[r]}$ be all the choices of $y_j^{(i)}$'s and z_j's in Box' respectively. For $y \in \{0,1\}^h$, let J_y be the set of index j that $h_0(z_j||x_j) = y$. That is

$$J_y \triangleq \{j : h_0(z_j||x_j) = y\}. \tag{5.49}$$

  Then, define

$$S(\mathbf{y}) \triangleq \bigcup_{y\in\{0,1\}^h : J_y \neq \emptyset} \{(y_{a_1}^{(1)}||...||y_{a_k}^{(k)}, y)\}_{(a_1,...,a_k)\in J_y^k}. \tag{5.50}$$

  The adversary succeed if and only if it finds an (x, z) such that $H'(z||x) \in S(\mathbf{y})$ and x is not equal to any x_j. The success probability is taken over the choice of $\mathbf{y}$, $\mathbf{z}$, H' and the randomness of $\mathcal{A}$. That is

$$\Pr[\textbf{Game 2}] \le \Pr_{\mathbf{y},\mathbf{z},\mathsf{H}',\mathcal{A}}[\mathsf{H}'(z||x) \in S(\mathbf{y})] = \mathbb{E}_{\mathbf{y}} \Pr_{\mathbf{z},\mathsf{H}',\mathcal{A}}[\mathsf{H}'(z||x) \in S(\mathbf{y})]. \tag{5.51}$$

Again, we use the adversary in **Game 1** to construct a reduction breaking $\mathsf{Avg\text{-}Search}_\lambda$ in Lemma 5. Fix **y** (and also $S(\mathbf{y})$). Let $f \leftarrow D_\lambda : \{0,1\}^{m+n} \to \{0,1\}$ and $\lambda = |S|/t^k$. Let $I : \{0,1\}^{m+n} \to |S|$ and $g : \{0,1\}^{m+n} \to [t]^k \times \{0,1\}^h \backslash S(\mathbf{y})$ be random functions. Reorder $S(\mathbf{y}) = \{y'_1, ..., y'_{|S(\mathbf{y})|}\}$. Construct $\tilde{\mathsf{H}}' : \{0,1\}^{m+n} \to [t]^k \times \{0,1\}^h$ as follows: for any $(z||x) \in \{0,1\}^{m+n}$, define

$$\tilde{\mathsf{H}}'(z||x) = \begin{cases} y_j & (x = x_j \wedge z = z_j), \\ y'_{I(z||x)} & (f(z||x) = 1), \\ g(x) & (otherwise). \end{cases} \tag{5.52}$$

If $\mathcal{A}$ succeeds in **Game 1**, then the reduction succeeds in $\mathsf{Avg\text{-}Search}$ with the same probability. From Lemma 5, we have

$$\Pr_{\mathbf{z},\mathsf{H}',\mathcal{A}}[\mathsf{H}'(z||x) \in S(\mathbf{y})] \leq 8(q+r+2)^2 \frac{|S(\mathbf{y})|}{2^h t^k}. \tag{5.53}$$

From the analysis in [17], for any $y' \in [t]^k \times \{0,1\}^h$, the probability of $y' \in S$ (over the choice of **y**) is at most Γ. Accordingly the expectation of $|S(\mathbf{y})|/(2^h t^k)$ over the choice of **y** is at most Γ. Thus, we have

$$\Pr[\textbf{Game 1}] \leq \mathbb{E}_{\mathbf{y}}\left[8(q+r+2)^2 \frac{|S(\mathbf{y})|}{2^h t^k}\right] \leq 8(q+r+2)^2\Gamma. \tag{5.54}$$

From equation (5.48) and (5.54), we complete the proof.

□

Remark 8 *We give an exact bound for the generic security of ITSR. Compared to the conjecture in [17], we have an additional term* $\frac{3r}{2}\sqrt{\frac{q+r+1}{2^n}}$ *here. This term does not have an essential role if a mild security argument based on the notion of security level is sufficient. Note that for HBS schemes, the security level is defined as the complexity of* q_H*, making the probability of breaking the security reach a constant. For example, in SPHINCS+-256s (*$r = 2^{64}$ *and* $n = 256$*), this additional term will cause 128-bit security, which is the same as the original security level of SPHINCS+-256s. Thus, this additional term has a small impact on the security level.*

5.3 Security Analysis of Few-time HBS Schemes

5.3.1 Few-time Hash-based Signature Schemes

In this section, we analyze two few-time HBS schemes.

The first is Hash to Obtain Random Subsets (HORS) [94]. The outline of HORS is as follows. In the key generation algorithm, it picks a one-way function $f : \{0,1\}^{l(n)} \to \{0,1\}^n$ and a (r,k)-subset-resilient function $H = (h_1, ..., h_k) : \{0,1\}^m \to [t]^k$. Then, it picks t random strings $s_1, ..., s_t$ from $\{0,1\}^{l(n)}$ and computes $y_j = f(s_j)$ for each $j \in [t]$. Let $(s_1, ..., s_t)$ be the secret key and $(y_1, ..., y_t)$ be the public key. In the signing algorithm, it reveals k elements from $\{s_j\}_{j\in[t]}$ determined by $H(m)$. Due to (r,k)-subset resilience of H, it is hard to find a message m^* such that the secret values in the corresponding signature are covered by r (classical) queries to the signing oracle. The formal description is as follows.

Construction 1 *(Hash to Obtain Random Subsets (HORS) [94].) Let $\mathcal{F} = \{\mathcal{F}_n : \{0,1\}^{l(n)} \to \{0,1\}^n\}$ and $\mathcal{H} = \{H : \{0,1\}^m \to [t]^k\}$ be hash function families. $\mathsf{HORS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ is depicted in Figure 5.3.*

SPHINCS [15] introduces a variant of HORS called HORST (HORS with trees). HORST compresses the public key with a (bitmarked) hash tree, and thus the signature needs to contain the corresponding authentication path. This operation does not hurt the security except that it introduces a second-preimage-resistant hash function.

Furthermore, SPHINCS+ [17] introduces an improvement of HORST, which is called FORS (Forest of Random Subsets). The main differences between HORST and FORS are as follows. First, the key generation algorithm picks kt random strings from $\{0,1\}^{l(n)}$ (rather than t strings) and divides them into k groups of t strings. In the signing algorithm, instead of revealing k elements from t strings, FORS reveals *one* element from each group. Second, FORS uses a tweakable hash function $\mathbf{F}$ instead of the one-way function f, where the tweaks are the indices of the strings. Third, instead of using bitmarked hash functions in the hash

```
HORS.KeyGen($1^\lambda$)
$F \leftarrow \mathcal{F}_n$, $H = (h_1, ..., h_k) \leftarrow \mathcal{H}$.
**for** $j \in [t]$, $s_j \leftarrow \{0,1\}^{l(n)}$, $y_j = f(s_j)$
$Y = (y_1, ..., y_t)$, $S = (s_1, ..., s_t)$
$pk = (Y, f, H)$, $sk = (S, f, H)$.
**return** $(pk, sk)$.

HORS.Sig($sk, m$)
Parse $S = (s_1, ..., s_t)$ and $H = (h_1, ..., h_k)$
**for** $i \in [k]$, $x_i = s_{h_i(m)}$.
**return** $\sigma = (x_1, ..., x_k)$.

HORS.Ver($pk, m, \sigma$)
Parse $\sigma = (x_1, ..., x_k)$ and $H = (h_1, ..., h_k)$.
**if for** $\forall i \in [k]$, $y_{h_i(m)} = f(x_i)$ **return** 1
**return** 0
```

Figure 5.3: Construction of Hash to Obtain Subsets (HORS).

tree, FORS uses a tweakable hash function **Th** in generating hash trees, where the tweaks are the addresses of the nodes. Finally, since there are k hash trees in FORS, it compresses the k roots by calling **Th** and denote the value as the public key.

In FORS, the message is not hashed. In the signing operation, it directly splits the message m into k digits with size $\log t$ and then proceeds the following steps. Thus, the scheme is not EU-CMA secure. (One can forge a signature and on message $m_1||m_2$ given signatures on $m_1||m_2^*$ and $m_1^*||m_2$.) In practice, FORS has to be used on hashed messages to achive EU-CMA. We call FORS with integrated hashing as *simplified* FORS (sFORS).

In SPHINCS+ [17], sFORS is never directly used. In each signing operation, it introduces a (pseudo-)randomizer z. The message is then hashed with z, and z is contained in the signature. It results in a new scheme that achieves higher security. We call it *randomized* FORS (rFORS).

Construction 2 *(Simplified & Randomized Forest of Random Subsets (sFORS & rFORS) [17].) Let* $\boldsymbol{T} = [k] \times [t]$ *and* $\alpha \geq \log t + 2$. *Let* $\boldsymbol{F} : \boldsymbol{T} \times \{0,1\}^{l(n)} \to \{0,1\}^n$ *and* $\boldsymbol{Th} : \{0,1\}^\alpha \times \{0,1\}^* \to \{0,1\}^n$ *be tweakable hash functions where* $\alpha \geq$ *and* $\mathcal{H} = \{H : \{0,1\}^* \to [t]^k\}$ *be a hash function family. Let PRF be a pseudorandom function. sFORS and rFORS are depicted in Figure 5.4 and 5.5.*

5.3.2 Security Analysis

In this subsection we analyze the security of few-time signature schemes. Some of the reductions have been given in previous work [94, 17] and Section 3.1.3. Particularly, we analyze the EU-qCMA security of sFORS.

In the security analysis, we use insecurity functions to show the maximum probability of breaking the security notions. For $* \in \{\mathsf{SR}, \mathsf{rSR}, \mathsf{wSR}, \mathsf{eTSR}, \mathsf{OW}\}$ and hash function family $\mathcal{H}$, $\mathrm{InSec}^{*}_{\mathcal{H},q_H}(\xi)$ denotes the maximum of $Adv^{*}_{\mathcal{H},q_H}(\mathcal{A})$ for all ξ-time adversary $\mathcal{A}$. In addition, for $* \in \{\mathsf{SM\text{-}TCR}, \mathsf{SM\text{-}DSPR}\}$ and tweakable hash function **Th**, $\mathrm{InSec}^{*}_{\mathbf{Th},q_1,q_2}(\xi)$ denotes the maximum of $Adv^{*}_{\mathcal{H},q_1,q_2}(\mathcal{A})$ for all ξ-time adversary $\mathcal{A}$.

```
$\mathsf{sFORS.KeyGen}(1^\lambda)$
    $H = (h_1, ..., h_k) \leftarrow \mathcal{H}$
    for $(i, j) \in [k] \times [t]$, $s_{i,j} \leftarrow \{0,1\}^{l(n)}$
    $y_{i,j} = \mathbf{F}((i,j), s_{i,j})$
    for $\forall i \in [k]$, $y_i \leftarrow \mathsf{TreeGen}(\mathbf{Th}, t, (y_{i,1}, ..., y_{i,t}))$.
    $y_0 = \mathbf{Th}(0, (y_1, ...y_k))$, $S = (s_{1,1}, ..., s_{k,t})$,
    $pk = (y_0, H)$, $sk = (S, H)$.
    return $(pk, sk)$.

$\mathsf{sFORS.Sig}(sk, m)$
    Parse $S = (s_{1,1}, ..., s_{k,t})$ and $H = (h_1, ..., h_k)$
    for $i \in [k]$, $x_i = s_{i,h_i(m)}$.
    for $(i, j) \in [k] \times [t]$, $y_{i,j} = \mathbf{F}((i,j), s_{i,j})$.
    for $\forall i \in [k]$, $\pi_i \leftarrow \mathsf{TreeProv}(\mathbf{Th}, t, (y_{i,1}, ..., y_{i,t}), h_i(m))$
    return $\sigma = (x_1, ..., x_k, \pi_1, ..., \pi_k)$.

$\mathsf{sFORS.pkFromSig}(m, \sigma, H)$
    Parse $\sigma = (x_1, ..., x_k, \pi_1, ..., \pi_k)$ and $H = (h_1, ..., h_k)$.
    for $\forall i \in [k]$, $y_i \leftarrow \mathsf{TreeVer}(\mathbf{Th}, t, h_i(m), \mathbf{F}((i,j), x_i), \pi_i)$.
    return $y_0' = \mathbf{Th}(0, (y_1, ..., y_k))$

$\mathsf{sFORS.Ver}(pk, m, \sigma)$
    Parse $pk = (y_0, H)$
    return $[\![\mathsf{sFORS.pkFromSig}(m, \sigma, H) = y_0]\!]$
```

Figure 5.4: Construction of Simplified FORS.

```
rFORS.KeyGen($1^\lambda$)
    $(pk, sk) \leftarrow$ sFORS.KeyGen($1^\lambda$)
    $k \leftarrow \{0,1\}^n$
    return $(pk, (sk, k))$.

rFORS.Sig($(sk, k), m$)
    $z = \mathsf{PRF}(k, m)$
    $\sigma \leftarrow$ sFORS.Sig($sk', z||m$)
    return $(z, \sigma)$.

rFORS.pkFromSig($m, (z, \sigma), H$)
    return sFORS.pkFromSig($z||m, \sigma, H$)

rFORS.Ver($pk, m, (z, \sigma)$)
    Parse $pk = (y_0, H)$
    return $[\![$rFORS.pkFromSig$(m, (z, \sigma), H) = y_0]\!]$
```

Figure 5.5: Construction of Randomized FORS.

First we show the EU-CMA security of HORS and sFORS. The reduction is straightforward and has been given in previous work [94] and Section 3.1.3.

Lemma 21 *For any ξ-time adversary $\mathcal{A}$, it holds that*

$$\begin{aligned} Adv^{\textsf{EU-CMA}}_{\textsf{HORS},r,q_H}(\mathcal{A}) &\le \textit{InSec}^{(r,k)\text{-}\textsf{SR}}_{\mathcal{H},q_H}(\xi) + kt \cdot \textit{InSec}^{\textsf{OW}}_{\mathcal{F},q_H}(\xi), \\ Adv^{\textsf{EU-CMA}}_{\textsf{sFORS},r,q_H}(\mathcal{A}) &\le \textit{InSec}^{\textsf{SM-TCR}}_{\textbf{Th},rkt,q_H}(\xi) + \textit{InSec}^{(r,k)\text{-}\textsf{rSR}}_{\mathcal{H},q_H}(\xi) \\ &\quad + 3\textit{InSec}^{\textsf{SM-TCR}}_{\textbf{F},kt,q_H}(\xi) + \textit{InSec}^{\textsf{SM-DSPR}}_{\textbf{F},kt,q_H}(\xi), \end{aligned}$$

Next, we focus on EU-qCMA security of sFORS.

Theorem 22 *For any ξ-time adversary $\mathcal{A}$, it holds that*

$$\begin{aligned} Adv^{\textsf{EU-qCMA}}_{\textsf{sFORS},r,q_H}(\mathcal{A}) \le \textit{InSec}^{\textsf{SM-TCR}}_{\textbf{Th},rkt,q_H}(\xi) &+ \textit{InSec}^{(r,k)\text{-}\textsf{wSR}}_{\mathcal{H},q_H}(\xi) \\ &+ kt^r\left(3\textit{InSec}^{\textsf{SM-TCR}}_{\textbf{F},t,q_H}(\xi) + \textit{InSec}^{\textsf{SM-DSPR}}_{\textbf{F},t,q_H}(\xi)\right). \end{aligned}$$

Proof. Our proof follows the idea of proving the EU-qCMA security of Lamport's scheme in [24]. Let $\mathcal{A}$ be a quantum adversary, we reduce $\text{Adv}^{\textsf{EU-qCMA}}_{\textsf{sFORS},r,q_H}(\mathcal{A})$ by the following hybrid arguments.

- **Game 0** is the original EU-qCMA experiment of sFORS.
- **Game 1** differs from **Game 0** as follows. In **Game 1**, the challenger stores all the $y_{i,j}$ in key generation algorithm. **Game 1** returns 0 if $\mathcal{A}$ outputs a $(m, \sigma) = (m, (x_1, ..., x_k, \pi_1, ..., \pi_k))$ such that $f(x_i) \neq y_{i,h_i(m)}$ for some $i \in [k]$.

  The probabilities of **Game 0** and **Game 1** differ only if the adversary generates a different hash tree of which the root collides with the real one. By Lemma 19, this implies a reduction to SM-TCR of **Th**. That is, there exists a reduction $\mathcal{M}_1$ such that

$$|\Pr[\textbf{Game 0}] - \Pr[\textbf{Game 1}]| \le Adv^{\textsf{SM-TCR}}_{\textbf{Th},rkt,q_H}(\mathcal{M}_1). \qquad (5.55)$$

- **Game 2** differs from **Game 1** as follows. Given forgeries $(m_j, \sigma_j)_{j\in[r+1]}$, **Game 2** outputs 0 if for $\forall i \in [k]$, $|\{h_i(m_j)\}_{j\in[r+1]}| \le r$ holds. If the output of the game differs, it implies that the adversary outputs a weak subset cover of $H = (h_1, ..., h_k)$, and thus succeeds in attacking the weak subset resilience of $\mathcal{H}$. There exists a reduction $\mathcal{M}_2$ such that

$$|\Pr[\textbf{Game 1}] - \Pr[\textbf{Game 2}]| \leq \mathrm{Adv}^{(r,k)\text{-wSR}}_{\mathcal{H},q_H}(\mathcal{M}_2). \tag{5.56}$$

If $\mathcal{A}$ succeeds in **Game 2**, there exists $i^* \in [k]$ such that $|\{h_{i^*}(m_j)\}_{j\in[r+1]}| = r+1$ holds, which means that $h_{i^*}(m_j)$'s are distinct for $j \in [r+1]$. In other words, $\mathcal{A}$ outputs $(r+1)$ preimages of $\{y_{i^*,j}\}_{j\in[t]}$ after only r queries to the signing oracle.

- **Game 3** differs from **Game 2** as follows. Before running $\mathcal{A}$, **Game 3** randomly guesses $i' \in [k]$. **Game 3** outputs 1 if and only if **Game 2** outputs 1 and $i' = i^*$. We have

$$\Pr[\textbf{Game 2}] \leq k \cdot \Pr[\textbf{Game 3}]. \tag{5.57}$$

- **Game 4** differs from **Game 3** as follows. Every time $\mathcal{A}$ queries to the signing oracle, the signing oracle performs a partial measurement after computing h_{i^*}. Since each measurement has at most t outcomes and there are at most r measurements, due to Lemma 4 we have

$$\Pr[\textbf{Game 3}] \leq t^r \cdot \Pr[\textbf{Game 4}]. \tag{5.58}$$

- In **Game 4**, the adversary can only obtain r elements of $\{s_{i^*,j}\}_{j\in[t]}$ with pure states, but it is required to output $(r+1)$ preimages in the forgeries. The forgeries must contain an x_{i^*,u^*} where $u^* \in [t]$ is never the outcome of the partial measurements on h_{i^*} (and thus s_{i^*,u^*} is never revealed), but x_{i^*,u^*} is a preimage of y_{i^*,u^*}.

  Given a successful adversary in **Game 4**, we construct a reduction $\mathcal{M}$ to attack SM-OpenPRE of $\mathbf{F}$. First, $\mathcal{M}$ queries all $(i,j) \in [k] \times [t]$ to oracle $\mathcal{O}$ and obtains $y_{i,j} = \mathbf{F}((i,j), s_{i,j})$ for $(i,j) \in [k] \times [t]$ where $s_{i,j}$'s are random strings picked by $\mathcal{O}$. In the second stage, $\mathcal{M}$ obtains parameter P of $\mathbf{F}$ and uses P and $y_{i,j}$'s to generate a public key of sFORS. Then, it queries the oracle Open with all $(i,j) \in [k] \times [t]$ except $i = i^*$. In response, it obtains all $s_{i,j}$'s expect $i = i^*$ and sends the public key to $\mathcal{A}$. When $\mathcal{A}$ sends a quantum signing query, $\mathcal{M}$ performs a partial measurement after computing h_{i^*} and suppose the outcome is u. It sends (i^*, u) to Open and obtains $s_{i^*,u}$ in response. It then simulates the signing oracle in **Game 4** and sends the corresponding signature to $\mathcal{A}$. Finally, when $\mathcal{A}$ successfully returns $(r+1)$ forgeries, there must be an x_{i^*,u^*} such that (i^*, u^*) is never sent to Open and is a preimage of y_{i^*,u^*}. Sending x_{i^*,u^*} to the challenger completes the SM-OpenPRE attack on **F**. We thus have

$$\Pr[\textbf{Game 4}] \leq \mathrm{Adv}_{\mathbf{F},t,q_H}^{\mathsf{SM\text{-}OpenPRE}}(\mathcal{M}). \tag{5.59}$$

By Lemma 17, there exist $\mathcal{M}_3$ and $\mathcal{M}_4$ such that

$$\Pr[\textbf{Game 4}] \leq 3\mathrm{Adv}_{\mathbf{F},t,q_H}^{\mathsf{SM\text{-}TCR}}(\mathcal{M}_3) + \mathrm{Adv}_{\mathbf{F},t,q_H}^{\mathsf{SM\text{-}DSPR}}(\mathcal{M}_4). \tag{5.60}$$

From equation (5.55), (5.56), (5.57), (5.58), and (5.60), we complete the proof.

□

By introducing Theorem 22, Corollary 1, Lemma 18 and the conjectured bound of SM-DSPR in Remark , we immediately obtain the security bounds of few-time siganture schemes in the quantum random oracle model.

Corollary 3 *Let hash funcions in HORS and sFORS be modeled as quantum random oracles. It holds that*

$$Adv_{\mathsf{HORS},r,q}^{\mathsf{EU\text{-}CMA}}(\mathcal{A}) \leq O\left(q^{2(r+1)}\left(\frac{rk}{t}\right)^k + \frac{q^2kt}{2^n}\right), \tag{5.61}$$

$$Adv_{\mathsf{sFORS},r,q}^{\mathsf{EU\text{-}CMA}}(\mathcal{A}) \leq O\left(q^{2(r+1)}\left(\frac{r}{t}\right)^k + \frac{q^2kt}{2^n}\right), \tag{5.62}$$

and

$$Adv_{\mathsf{sFORS},r,q}^{\mathsf{EU\text{-}qCMA}}(\mathcal{A}) \leq O\left(q^{2(r+1)}\left(\frac{r^2}{t}\right)^k + \frac{q^2kt^{r+1}}{2^n}\right). \tag{5.63}$$

Remark 9 *In the above equalities, we use the loose bound of SM-DSPR. If we use the conjectured bound of SM-DSPR in Remark 6, then the second and third bounds are*

$$Adv^{\textit{EU-CMA}}_{\textit{sFORS},r,q}(\mathcal{A}) \leq O\left(q^{2(r+1)}\left(\frac{r}{t}\right)^k + \frac{q^2}{2^n}\right), \tag{5.64}$$

and

$$Adv^{\textit{EU-qCMA}}_{\textit{sFORS},r,q}(\mathcal{A}) \leq O\left(q^{2(r+1)}\left(\frac{r^2}{t}\right)^k + \frac{q^2kt^r}{2^n}\right). \tag{5.65}$$

We always follow the above bounds in the rest of the dissertation.

Next, we give an example of the concrete security of few-time signatures by implementing the parameters used in SPHINCS-256 [15] ($t = 2^{16}$, $k = 32$, $n = 256$). The concrete security of few-time signature schemes is summarized in Figure 5.6. Here the security levels denote the logarithm of q_H such that the probabilities depicted in Corollary 3 reach a constant. It implies lower bounds of the generic security of the schemes.

In addition, note that an attack on restricted-subset resilience immediately implies a chosen message attack on the corresponding sFORS. In Chapter 3, a generic quantum attack is shown on (r, k)-restricted subset resilience with $O(kt^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})})$ quantum queries to H. It implies an upper bound of the generic security of sFORS (and HORS) against CMA (and qCMA, respectively). The comparison is also depicted in Figure 5.6.

Next, we analyze the security of rFORS.

Theorem 23 *For any ξ-time adversary $\mathcal{A}$, it holds that*

$$\begin{aligned} Adv^{\textit{EU-CMA}}_{\textit{rFORS},r,q_H}(\mathcal{A}) \leq\ & \textit{InSec}^{\textit{SM-TCR}}_{\boldsymbol{Th},rkt,q_H}(\xi) + \textit{InSec}^{(r,k)\textit{-eTSR}}_{\mathcal{H},q_H}(\xi) + \textit{InSec}^{\textit{Ind-PRF},r}_{\textit{PRF}}(\xi) \\ & + 3\textit{InSec}^{\textit{SM-TCR}}_{\boldsymbol{F},kt,q_H}(\xi) + \textit{InSec}^{\textit{SM-DSPR}}_{\boldsymbol{F},kt,q_H}(\xi), \end{aligned}$$

$$\begin{aligned} Adv^{\textit{EU-qCMA}}_{\textit{rFORS},r,q_H}(\mathcal{A}) \leq\ & \textit{InSec}^{\textit{SM-TCR}}_{\boldsymbol{Th},rkt,q_H}(\xi) + \textit{InSec}^{(r,k)\textit{-wSR}}_{\mathcal{H},q_H}(\xi) \\ & + kt^r\left(3\textit{InSec}^{\textit{SM-TCR}}_{\boldsymbol{F},t,q_H}(\xi) + \textit{InSec}^{\textit{SM-DSPR}}_{\boldsymbol{F},t,q_H}(\xi)\right). \end{aligned}$$

Proof.

r	CMA-HORS	CMA-FORS	qCMA-FORS	Attacks
1	87.0	116.5	108.5	128.0
2	52.3	79.0	73.6	114.8
3	36.6	56.6	50.3	87.2
4	27.8	43.8	37.4	67.0
5	22.1	35.4	29.2	60.2
6	18.2	29.6	23.7	52.6
7	15.3	25.3	19.7	44.9

Figure 5.6: Concrete security of r-time HBS schemes derived from Corollary 3 with concrete parameters in SPHINCS-256. The three columns in the left hand show lower bounds of the logarithm of hash queries needed in breaking the generic security, and the rightmost column shows upper bounds.

1. (Proof of EU-CMA.)

   The reduction of rFORS is similar to sFORS. The main difference is that rFORS is not reduced to rSR. Instead, it is reduced to eTSR and the security of PRF. Since the signing algorithm is deterministic, we suppose the adversary does not require repeated messages in the experiment. Let $(m^*, (z^*, \sigma^*))$ be a forgery that $\mathcal{A}$ outputs.

   The hybrid argument is as follows.

   - **Game 0** is the original EU-CMA experiment of rFORS.
   - **Game 1** diffes from **Game 0** in that it returns 0 if σ^* contains a different hash tree with the real one. As shown in the proof of Theorem 22, there exists a reduction $\mathcal{M}_1$ such that

   $$|\Pr[\textbf{Game 0}] - \Pr[\textbf{Game 1}]| \leq Adv^{\mathsf{SM\text{-}TCR}}_{\mathbf{Th},rkt,q_H}(\mathcal{M}_1) \tag{5.66}$$

   - **Game 2** differs from **Game 1** in that it returns 0 if σ^* contains a preimage which has not been revealed in the signing queries. Also as shown in the proof of Theorem 22, a difference of success probability with **Game 1** implies a breaker of SM-PRE of **F**. Then, there exist reduction $\mathcal{M}_2$ and $\mathcal{M}_3$ such that

$$|\Pr[\textbf{Game 1}] - \Pr[\textbf{Game 2}]| \leq 3\mathrm{Adv}^{\textsf{SM-TCR}}_{\mathbf{F},t,q_H}(\mathcal{M}_2) + \mathrm{Adv}^{\textsf{SM-DSPR}}_{\mathbf{F},t,q_H}(\mathcal{M}_3). \tag{5.67}$$

- **Game 3** differs from **Game 2** in that the signing oracle does not calculate pseudorandom functions. Instead, it uses a truly random function (which can be instantiated by querying a random oracle). If the probability differs, there exists a distingusher $\mathcal{M}_4$ breaking the security of PRF:

$$|\Pr[\textbf{Game 3}] - \Pr[\textbf{Game 2}]| \leq Adv^{\textsf{Ind-PRF}}_{\textsf{PRF},r}(\mathcal{M}_4). \tag{5.68}$$

- In **Game 3**, the adversary succeeds if for $\forall i \in [k]$, it always holds that $h_i(z^*||m^*) \in \{h_i(z_j||m_j)\}_{j\in[r]}$. We construct a reduction $\mathcal{M}_5$ that breaks eTSR of $\mathcal{H}$. Given challenge $\mathcal{H}$, it generates the key pair of rFORS and give the public key to $\mathcal{A}$. When the adversary queries m_j to the signing oracle, $\mathcal{M}_5$ queries m_j to the oracle Box and obtains $(z_j, H(z_j))$ in response. Then, $\mathcal{M}_5$ generates the corresponding signature and gives it to $\mathcal{A}$. Finally, $\mathcal{M}_5$ returns (m^*, z^*) to the challenger. If $\mathcal{A}$ succeeds, then $\mathcal{M}_5$ succeeds with the same probability. We thus have

$$\Pr[\textbf{Game 3}] \leq Adv^{(r,k)\textsf{-eTSR}}_{\mathcal{H},q_H}(\mathcal{M}_5). \tag{5.69}$$

From equation (5.66), (5.67), (5.68) and (5.69), we complete the proof.

2. (Proof of EU-qCMA.)

We claim that rFORS is at least as secure as sFORS (if we suppose that the input of $\mathcal{H}$ is of arbitrary length). That is, if there exists an adversary $\mathcal{A}$ breaks EU-qCMA security of rFORS, then we construct $\mathcal{M}$ breaking EU-qCMA security of sFORS.

Given the public key pk of sFORS as the challenge, $\mathcal{M}$ randomly picks $k \in \{0,1\}^n$ and then sends pk to $\mathcal{A}$. Every time $\mathcal{A}$ queries $\sum_{m,t} |m,t\rangle$, $\mathcal{M}$ computes $\sum_{m,t} |m, \mathsf{PRF}_k(m), t\rangle$ and queries $\sum_{m,t} \left|\left(\mathsf{PRF}_k(m)||m\right), t\right\rangle$ to the signing oracle. Then, $\mathcal{M}$ sends to $\mathcal{A}$ what it receives from the signing oracle. Finally, when $\mathcal{A}$ returns $\{(m_j^*, (z_j^*, \sigma_j^*))\}_{j\in[r+1]}$, $\mathcal{M}$ returns $\{(z_j^*||m_j^*, \sigma_j^*)\}_{j\in[r+1]}$. It is a set of valid forgeries of sFORS if $\mathcal{A}$ succeeds. We thus have

$$\mathrm{Adv}^{\mathsf{EU\text{-}qCMA}}_{\mathsf{rFORS},r}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{EU\text{-}qCMA}}_{\mathsf{sFORS},r}(\mathcal{M}). \tag{5.70}$$

By Theorem 22, we complete the proof.

□

By imcoporating Theorem 20 to Theorem 23, we obtain the generic security of rFORS in the random oracle model as follows.

Corollary 4 *Let the hash functions in rFORS be modeled as quantum random oracles. It holds that*

$$Adv^{EU\text{-}CMA}_{rFORS,r,q}(\mathcal{A}) \leq O\left(\sqrt{\frac{q}{2^n}} + q^2\left(\frac{r}{t}\right)^k + \frac{q^2}{2^n}\right), \tag{5.71}$$

and

$$Adv^{EU\text{-}qCMA}_{rFORS,r,q}(\mathcal{A}) \leq O\left(q^{2(r+1)}\left(\frac{r^2}{t}\right)^k + \frac{q^2ktr^r}{2^n}\right). \tag{5.72}$$

5.4 Quantum-access Attacks on Existing Many-time Stateless HBS Schemes

5.4.1 Many-time Stateless HBS Schemes

SPHINCS [15] and SPHINCS+[17] are many-time stateless HBS schemes. Briefly speaking, they introduce a stateful HBS with trees (denoted by HT) and a few-time HBS. In each signing execution, the signer first pseudorandomly picks a leaf from HT, which authenticate the public key of the few-time HBS. The signature contains (1) the few-time signature of the message, (2) the HT signature of the public key of the corresponding few-time HBS, and (3) the (pseudo-)randomizer.

Note that in SPHINCS and its variants, the few-time signatures are never explicitly verified. Instead, it uses pkFromSig to recover the public key from the message and signature. The security of HT guarantees that only the real public key can be verified in the next steps.

Remark 10 *Essentially, SPHINCS and SPHINCS+ uses the constructions in Figures 4.9 and 4.6 in Chapter 4. In this chapter, the syntax of stateful signature is slightly different from that in Chapter 4. The verification algorithm also takes as input the state. In addition, the security notion of HT is also different from that in 4. The new security notion is described in the following.*

Although SPHINCS and SPHINCS+ use different HT, the security notions for HT are the same and implicitly contained in [15, 17]. The security is a stateful version of existential unforgeability under non-adaptive chosen message attacks. We call it *existential unforgeability under non-adaptive chosen message attacks with states* (EU-sNACMA). In detail, the security experiment is defined as follows.

```
**Experiment** $\mathsf{Exp}^{\mathsf{EU\text{-}sNACMA}}_{\mathsf{HT},q_s,q_H}(1^n, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$
    $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$
    $(\{(m_i, \mathsf{st}_i)\}_{i\in[q_s]}, S) \leftarrow \mathcal{A}_1()$
    If $\mathsf{st}_i$ are not distinct, **return** 0
    **For** $i \in [q_s]$, $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i, \mathsf{st}_i)$
    $(m^*, \sigma^*, \mathsf{st}^*) \leftarrow \mathcal{A}_2(pk, S, \{\sigma_i\}_{i\in[q_s]})$
    If $\mathsf{st}^* = \mathsf{st}_j$ for some $j \wedge m^* \neq m_j \wedge \mathsf{Ver}(pk, m^*, \sigma^*, \mathsf{st}^*) = 1$, **return** 1,
otherwise **return** 0.
```

In this chapter, we do not depict the detailed construction of HT in SPHINCS and SPHINCS+. We use it as a black box and only care about the security.

In SPHINCS, the few-time signature scheme is HORST (HORS with trees), a variant of HORS. It compresses the HORS public key by a Merkle tree structure, and the compressed public key can be generated by an algorithm pkFromSig from the message and the signature. In this section, we also use HORS.Sig and HORS.pkFromSig to denote the algorithms of HORS with trees.

```
SPHINCS.KeyGen($1^\lambda$)
$sk_{\mathsf{seed}} \leftarrow \{0,1\}^n$
$(pk_{\mathsf{HT}}, sk_{\mathsf{HT}}) \leftarrow \mathsf{HT.KeyGen}(1^\lambda)$
$sk = (sk_{\mathsf{seed}}, sk_{\mathsf{HT}}), pk = pk_{\mathsf{HT}}$
**Output** $(pk, sk)$.

SPHINCS.Sig($sk, m$)
$z = \mathsf{PRF}_{\mathsf{msg}}(sk_{\mathsf{seed}}, m), idx = \mathsf{PRF}_{\mathsf{idx}}(sk_{\mathsf{seed}}, m)$
$s_{idx} = \mathsf{PRF}_{\mathsf{seed}}(sk_{\mathsf{seed}}, idx)$
$(pk_{\mathsf{HORS}}, sk_{\mathsf{HORS}}) \leftarrow \mathsf{HORS.KeyGen}(1^\lambda; s_{idx})$
$\sigma_{\mathsf{HORS}} \leftarrow \mathsf{HORS.Sig}(sk_{\mathsf{HORS}}(z||m))$
$\sigma_{\mathsf{HT}} \leftarrow \mathsf{HT.Sig}(sk_{\mathsf{HT}}, pk_{\mathsf{HORS}}, idx)$
**return** $(idx, z, \sigma_{\mathsf{HT}}, \sigma_{\mathsf{HORS}})$.

SPHINCS.Ver($m, (idx, z, \sigma_{\mathsf{HT}}, \sigma_{\mathsf{HORS}})$)
$pk_{\mathsf{HORS}} \leftarrow \mathsf{HORS.pkFromSig}(z||m, \sigma_{\mathsf{HORS}})$
**return** $\mathsf{HT.Ver}(pk_{\mathsf{HT}}, pk_{\mathsf{HORS}}, \sigma_{\mathsf{HT}}, idx)$
```

Figure 5.7: The framework of SPHINCS

In detail, the frameworks of SPHINCS and SPHINCS+ are depicted in Figure 5.7 and 5.8.

Remark 11 *Figure 5.8 shows the deterministic version of SPHINCS+. It can be converted into a probabilistic version by adding a random salt to the input of $\mathsf{PRF}_{\mathsf{msg}}$. In this section, we mainly focus on the deterministic version and will discuss the probabilistic version in the next section.*

5.4.2 Quantum Chosen Message Attacks on SPHINCS

From this subsection, we analyze the quantum-access security of SPHINCS and SPHINCS+. Let q_s be the maximum number of signing queries and q_H be the maximum number of hash queries. For classical-accessible security (EU-CMA),

```
$\mathsf{SPHINCS+.KeyGen}(1^\lambda)$
    $sk_{\mathsf{seed}} \leftarrow \{0,1\}^n$, $h_0 \leftarrow \mathcal{H}_0$
    $(pk_{\mathsf{HT}}, sk_{\mathsf{HT}}) \leftarrow \mathsf{HT.KeyGen}(1^\lambda)$
    $sk = (sk_{\mathsf{seed}}, sk_{\mathsf{HT}})$, $pk = (pk_{\mathsf{HT}}, h_0)$
    **Output** $(pk, sk)$.

$\mathsf{SPHINCS+.Sig}(sk, m)$
    $z = \mathsf{PRF}_{\mathsf{msg}}(sk_{\mathsf{seed}}, m)$
    $idx = h_0(z||m)$
    $s_{idx} = \mathsf{PRF}_{\mathsf{seed}}(sk_{\mathsf{seed}}, idx)$
    $(pk_{\mathsf{FORS}}, sk_{\mathsf{FORS}}) \leftarrow \mathsf{sFORS.KeyGen}(1^\lambda; s_{idx})$
    $\sigma_{\mathsf{FORS}} \leftarrow \mathsf{sFORS.Sig}(sk_{\mathsf{FORS}}, z||m)$
    $\sigma_{\mathsf{HT}} \leftarrow \mathsf{HT.Sig}(sk_{\mathsf{HT}}, pk_{\mathsf{FORS}}, idx)$
    **return** $(z, \sigma_{\mathsf{HT}}, \sigma_{\mathsf{FORS}})$.

$\mathsf{SPHINCS+.Ver}(pk, m, (z, \sigma_{\mathsf{HT}}, \sigma_{\mathsf{FORS}}))$
    $idx = h_0(z||m)$
    $pk_{\mathsf{FORS}} \leftarrow \mathsf{sFORS.pkFromSig}(z||m, \sigma_{\mathsf{FORS}})$
    **return** $\mathsf{HT.Ver}(pk_{\mathsf{HT}}, pk_{\mathsf{FORS}}, \sigma_{\mathsf{HT}}, idx)$
```

Figure 5.8: The framework of SPHINCS+

it needs approximately 2^{128} hash queries to break the security of SPHINCS-256 when $q_s = 2^{50}$ and also approximately the same number of hash queries to break that of SPHINCS+-256s. One may consider that the success probability may increase when the signing queries become quantum. In this subsection, we prove this conjecture by showing some concrete quantum chosen message attacks on SPHINCS and SPHINCS+ that can succeed with a larger probability than classical chosen message attacks.

The outline of the attack is as follows.

In SPHINCS, $idx = \mathsf{PRF}_{\mathsf{idx}}(sk_{\mathsf{seed}}, \mathsf{PRF}_{\mathsf{msg}}(sk_{\mathsf{seed}}, m))$. Fix sk_{seed}. Then, it implies a function $f(m)$ mapping m to idx. Although an adversary cannot calculate f without knowing sk_{seed}, idx is a part of the signature and thus f can be computed by querying the signing oracle. By using Grover's algorithm in Lemma 3, one can search a message m mapping to any specialized index after $O(2^{h/2})$ queries to the signing oracle.

Note that any index authenticates a key pair of the few-time signature scheme. By repeating the above steps r times, one can obtain r message-signature pairs w.r.t. the same few-time signature key pair. If the secret key of the few-time signature is used too many times, the security level will be degraded rapidly.

The formal description of the quantum chosen message attack on SPHINCS is as follows:

1. Given pk and the signing oracle SigO, for some $idx^* \in \{0,1\}^h$, denote function $F(m) = 1$ iff $\mathsf{SigO}(m, 0^h, 0^*) = (m, idx^*, *)$. Here, F is quantum-computable by querying SigO.

2. Run Grover's algorithm on $F(m)$. It outputs a random m such that $F(m) = 1$. It implies a HORS signature labeled by idx^*. This requires $O(2^{h/2})$ quantum queries to the signing oracle.

3. Repeat the previous step r times. This requires $O(r2^{h/2})$ quantum queries to the signing oracle. Let S be the set of labels of which the preimages have appeared in the HORS signatures. In other words, let $m^{(1)}, ..., m^{(r)}$ be the outputs of the Grover's algorithm, then $S = \{h_i(m^{(j)})\}_{i\in[k],j\in[r]}$.

Note that for each s_j, the probability of appearing in a HORS signature is k/t. The probability of appearing in r random signatures is $1-(1-k/t)^r$. Thus, the expectation of $|S|$ is

$$\mathbb{E}[|S|] = \left(1 - \left(1 - \frac{k}{t}\right)^r\right) \cdot t \geq (1 - e^{-\frac{kr}{t}}) \cdot t, \tag{5.73}$$

where the equality comes from $(1-x)^\alpha \leq e^{-\alpha x}$.

4. Denote function $G(z||m) = 1$ iff $\{h_i(z||m)\}_{i\in[k]} \subset S$. Run Grover's algorithm on G. It outputs $z^*||m^*$ whose corresponding preimages have appeared in the HORS signatures. The expected number of quantum hash queries in this step is

$$O\left(\sqrt{\left(\frac{t}{|S|}\right)^k}\right) = O((1-e^{\frac{kr}{t}})^{-\frac{k}{2}}). \tag{5.74}$$

5. Note that the signing oracle has been queried $q_s = O(r2^{h/2})$ times. The forgery needs to contain at least one-more message-signature pairs. Step 4 needs to be repeated $(q_s + 1 - r)$ times. (It is unnecessary to compute σ_{HT} for the new forgeries since all the forgeries share a common σ_{HT}.) The total number of hash queries is

$$q_H = O(q_s + 1 - r) \cdot O((1 - e^{-\frac{kr}{t}})^{-\frac{k}{2}}) = O((1 - e^{-\frac{kr}{t}})^{-\frac{k}{2}} \cdot r2^{\frac{h}{2}}). \tag{5.75}$$

In SPHINCS-256, $k = 32$, $t = 2^{16}$ and $h = 60$. When $r = 2^{10}$, q_s and q_H are approximately 2^{40} and 2^{61}, respectively. When $r = 2^{14}$, q_s reaches 2^{43} and q_H decreases to 2^{43} as well. It is much lower than the level of EU-CMA security, where $q_s = 2^{50}$, and q_H is expected to be 2^{128}.

Remark 12 *The purpose of Step 1 and 2 is to obtain r messages mapping to the single index, idx^*. Essentially, what we need here is an r-collision of the function mapping m to idx^*. Intuitively, finding an r-collision is easier than searching r preimages for a special index. However, when r is large, the quantum query complexities of the two problems are close. Finding an r-collision* *requires $O(r2^{\frac{h}{2}(1-\frac{1}{2^{r+1}-1})})$ hash queries [84] (with additional huge memory cost) and finding r preimages by Grover's algorithm requires $O(r2^{\frac{h}{2}})$ queries. Since it greatly increases memory cost and only slightly decreases the time cost, we will not use the multi-collision finding algorithm here.*

5.4.3 Quantum Chosen Message Attacks on SPHINCS+

In SPHINCS+, the index is not directly contained in the signature. It is computed by $idx = h_0(z||m)$, avoiding forgeries with a malicious index. It is not a big issue since we can add some additional conditions on Grover's algorithm to find a malicious randomizer z mapping to our malicious index. In this section, we simply denote by H the hash computation of h_0 and $(h_1, ..., h_k)$. (In practice, h_0 and $(h_1, ..., h_k)$ are parts of a function H.)

Our attack on SPHINCS+ is as follows:

1. Given pk and signing oracle SigO. Let $\mathsf{SigO} : (m, 0^n, 0^*) \to (m, z, *)$. Let function $z(m)$ be the map from m to the corresponding z. For some $idx^* \in \{0,1\}^h$, denote predicate $F(m) = 1$ iff $h_0(z(m)||m) = idx^*$. Here, F is quantum-computable by querying SigO and a quantum query to h_0.

2. Run Grover's algorithm on $F(m)$. It outputs a random m such that $F(m) = 1$. It implies a sFORS signature labeled by idx^*. This requires $O(2^{h/2})$ quantum queries to the signing oracle and $O(2^{h/2})$ quantum queries to h_0.

3. Repeat the previous step r times. This requires $O(r2^{h/2})$ quantum queries to the signing oracle. For $i \in [k]$, let S_i be the set of labels of which the preimages have appeared in the i-th tree of sFORS signatures. In other words, $S_i = \{h_i(m^{(j)})\}_{j\in[r]}$.

   For each $s_{i,j}$, the probability of appearing in an sFORS signature is $1/t$. The probability of appearing in r random signatures is $1-(1-1/t)^r$. Thus, the expectation of $|S_i|$ is

$$\mathbb{E}[|S_i|] = \left(1 - \left(1 - \frac{1}{t}\right)^r\right) \cdot t \geq (1 - e^{-\frac{r}{t}}) \cdot t, \qquad (5.76)$$

and thus

$$\mathbb{E}\left[\prod_{i\in[k]} |S_i|\right] \geq (1-e^{-\frac{r}{t}})^k \cdot t^k. \quad (5.77)$$

4. Denote function $G(z||m) = 1$ iff $\forall i \in [k], h_i(z||m) \in S_i \wedge h_0(z||m) = idx^*$. Run Grover's algorithm on G. It outputs $z^*||m^*$ whose corresponding preimages have appeared in sFORS signatures. The expected number of quantum hash queries in this step is

$$O\left(\sqrt{\left(\frac{2^h \cdot t^k}{\prod_{i\in[k]} |S_i|}\right)}\right) = O((1-e^{-\frac{r}{t}})^{-\frac{k}{2}} \cdot 2^{\frac{h}{2}}). \quad (5.78)$$

5. The signing oracle has been queried $q_s = O(r2^{h/2})$ times. The forgery needs to contain at least one more message-signature pair. Step 4 needs to be repeated $(q_s + 1 - r)$ times. The total number of hash queries is

$$q_H = O(r2^{\frac{h}{2}}) + O(q_s+1-r)\cdot O((1-e^{-\frac{r}{t}})^{-\frac{k}{2}}\cdot 2^{\frac{h}{2}}) = O((1-e^{-\frac{r}{t}})^{-\frac{k}{2}}\cdot r2^h). \quad (5.79)$$

In SPHINCS+-256s, $k = 22$, $t = 2^{14}$ and $h = 64$. When $r = 2^{16}$, q_s and q_H are approximately 2^{48} and 2^{80}, respectively. It is also lower than the level of EU-CMA security, where $q_s = 2^{64}$ and q_H is expected to be 2^{128}.

5.4.4 Attack on PRF

Another quantum-access attack focuses on $\mathsf{PRF}_{\mathsf{msg}}$. It does not work well in practice but still reminds a potential risk under quantum chosen message attacks.

In SPHINCS and SPHINCS+, the security notion of $\mathsf{PRF}_{\mathsf{msg}}$ and $\mathsf{PRF}_{\mathsf{idx}}$ are (post-quantum) pseudorandomness. That is, any polynomial-time (quantum) adversary cannot tell the difference between oracle $\mathsf{PRF}_{\mathsf{msg}}(k,\cdot)$ (where k is randomly chosen) and a random oracle $O(\cdot)$. Note that here the oracle is classical-accessible, meaning that the adversary can only query the oracle with pure states in the experiment. However, if the oracle becomes quantum-accessible, the security may differ. In [106], the author shows a separating example PRF that is secure under classical queries but completely insecure under quantum queries. That is, there exists an adversary that queries the quantum oracle $\mathsf{PRF}(k, \cdot)$ polynomial times and then becomes able to calculate $\mathsf{PRF}(k, \cdot)$ without querying the oracle anymore.

If the pseudorandom functions are instantiated by such quantum-insecure instances, the resulting SPHINCS and SPHINCS+ will be fragile under quantum-access attacks. Let us give an example of SPHINCS+. Since $z = \mathsf{PRF}_{\mathsf{msg}}(sk_{\mathsf{seed}}, m)$ is contained in the signature, $\mathsf{PRF}_{\mathsf{msg}}(sk_{\mathsf{seed}}, \cdot)$ is implied in the signing oracle. A quantum adversary mentioned above can send quantum queries to the signing oracle and then obtain the ability of calculating $\mathsf{PRF}_{\mathsf{msg}}(sk_{\mathsf{seed}}, \cdot)$. As a result, the adversary can calculate $f(m) = h_0(\mathsf{PRF}_{\mathsf{msg}}(sk_{\mathsf{seed}}, m)||m)$ without querying the signing oracle and use Grover's algorithm to find a message m mapping to a special index idx^*. Then, the adversary sends m with the pure state to the signing oracle and obtains a few-time signature labeled by idx^*. By repeating the previous step r times, the adversary can obtain r few-time signatures and then generate a forgery for any message if r is large enough (approximately $r = ct$ where $c > 1$ is a small constant). Finally, the adversary uses Grover's algorithm to find $(q_s + 1)$ number of (z, m) pairs such that $h_0(z||m) = idx^*$ (this requires $O(q_s 2^{\frac{h}{2}})$ hash queries). If the number of queries needed in attacking $\mathsf{PRF}_{\mathsf{msg}}$ is smaller than $O(2^{\frac{h}{2}})$, the signing queries and hash queries will be smaller than the attacks in the previous subsections.

In practice, $\mathsf{PRF}_{\mathsf{msg}}$ is instantiated by hash functions such as SHA-256 and SHAKE-256. They are conjectured secure in quantum-access settings. Thus, it is likely that there does not exist such an efficient quantum-access attacker on $\mathsf{PRF}_{\mathsf{msg}}$. However, to make the scheme remain secure under quantum chosen message attacks, it is natural to adapt the security notion for $\mathsf{PRF}_{\mathsf{msg}}$ to be resistant to quantum-access adversaries (e.g., QPRF [106]).

5.4.5 Attacks in the BU model

In the above attacks, when r is large enough, the adversary can forge a signature for any message that it wants. However, because of the rule of EU-CMA experiment, the adversary has to generate q_s forgeries, which leads to large q_H in the attacks.

Now that the adversary has already obtained the ability to forge any signatures, intuitively, we should admit that the security has been broken and that the adversary succeeds. Thus, it then seems meaningless to "force" the adversary to repeatedly generate such a large number of forgeries.

In Section **??**, we have introduced a security notion called blind unforgeability (BU). In this subsection, we call the original security notion (existential unforgeability under quantum chosen message attacks) PO security, which means Plus-One. It has been proven that the BU security implies the PO security, meaning that if there exists an adversary $\mathcal{A}$ that breaks the PO security, then, there exists a general reduction $\mathcal{M}^{\mathcal{A}}$ that can break the BU security. Thus, it is natural that our attacks also work in the BU model. If we use the BU model, the adversary does not need to execute the "meaningless" hash queries as mentioned above. That is, after the adversary gains the ability to forge signatures, it only needs to forge one signature for a message in the blind region $B_{\varepsilon,n}$. The number of hash queries will be much lower than that in the PO model.

In the following, we omit the parameter n and use B_ϵ to denote $B_{\epsilon,n}$. In detail, the strategy of breaking the BU security of SPHINCS is as follows.

1. Given pk and quantum access to signing oracle $B_\varepsilon\mathsf{SigO}$, for some $idx^* \in \{0,1\}^h$, denote function $F(m) = 1$ iff $B_\varepsilon\mathsf{SigO}(m, 0^h, 0^*) = (m, idx^*, *)$. (Note that $F(m) = 1$ implies that $m \notin B_\varepsilon$.) Here, F is quantum-computable by querying $B_\varepsilon\mathsf{SigO}$.
2. Run Grover's algorithm on $F(m)$. This requires $O\Big(\sqrt{\frac{2^h}{1-\varepsilon}}\Big)$ queries to $B_\varepsilon\mathsf{SigO}$.
3. Repeat the last step r times and denote S as above. The expectation of $|S|$ is also $(1 - e^{-\frac{kr}{t}}) \cdot t$.
4. Denote predicate $F'(m) = 1$ iff $B_\varepsilon\mathsf{SigO}(m) = \perp$. It outputs a message $m^* \in B_\varepsilon$. This requires approximately $O(\varepsilon^{-1/2})$ signing queries.
5. Denote function $G(z) = 1$ iff $\{h_i(z||m^*)\}_{i\in[k]} \subset S$. Run Grover's algorithm on G. It outputs z^* such that the preimages corresponding to (z^*, m^*) have

appeared in S. The expected number of quantum hash queries in this step is also $O((1-e^{\frac{kr}{t}})^{-\frac{k}{2}})$.

Then, the adversary successfully forges $\sigma^* = (idx^*, z^*, \sigma^*_{\mathsf{HT}}, \sigma^*_{\mathsf{HORS}})$ for $m^* \in B_\varepsilon$, where σ^*_{HT} is obtained by an additional signing query and σ^*_{HORS} is obtained by S.

The total number of required queries is

$$q_s = O(r2^{\frac{h}{2}}(1-\varepsilon)^{-\frac{1}{2}}) + O(\varepsilon^{-\frac{1}{2}}), \quad q_H = O((1-e^{-\frac{kr}{t}})^{-\frac{k}{2}}). \tag{5.80}$$

Note that q_s is slightly larger than the original attack (increased by a polynomial $\sqrt{\varepsilon}$) and q_H decreases to $1/r2^{h/2}$ of the original one.

The above attack also works on SPHINCS+, but we have another one that requires less queries. The main idea is similar. First, we find a message m^* in the blind region. Then, to sign a message for m^*, we need an sFORS signature associated with some index idx^*. Note that the sFORS signature includes k elements. We directly use Grover's algorithm to search k messages (outside of the blind region) that respectively map to the k target elements. It can be done by quantum signing queries. Finally, the sFORS signature of m^* is covered by the k signatures, and a forgery is generated. The attack is as follows.

1. Given pk and quantum access to signing oracle $B_\varepsilon\mathsf{SigO}$, find m^* such that $B_\varepsilon\mathsf{SigO}(m^*) = \perp$. This requires $O(\varepsilon^{-\frac{1}{2}})$ quantum hash queries.

2. Let $z^* \in \{0,1\}^n$ and $idx^* = h_0(z^*||m^*)$. Let $\mathsf{SigO} : (m, 0^n, 0^*) \to (m, z, *)$. Let function $z(m)$ be the map from m to the corresponding z. For $i \in [k]$, denote predicate $F_i(m) = 1$ iff (1) $z(m) \neq \perp$, (2) $h_0(z(m)||m) = idx^*$ and (3) $h_i(z(m)||m) = h_i(z^*||m^*)$. F_i is quantum-computable by querying SigO and a quantum query to h_0, h_i. Run Grover's algorithm on F_i. The expected number of F_i computations is $O(\sqrt{(1-\varepsilon)^{-1} \cdot 2^h \cdot t})$.

3. After that, the secret values in sFORS signature on m^* is covered the k signatures. A forgery is then generated.

In total, the number of required queries is

$$q_s = O(k2^{\frac{h+\log t}{2}}(1-\varepsilon)^{-\frac{1}{2}}) + O(\varepsilon^{-\frac{1}{2}}), \quad q_H = O(k2^{\frac{h+\log t}{2}}(1-\varepsilon)^{-\frac{1}{2}}). \tag{5.81}$$

With the parameters in SPHINCS+-256s, q_s and q_H are both approximately 2^{43}, which is lower than out attack in the PO model.

Remark 13 *The above attack can also be simply modified to attack SPHINCS, but the number of hash queries is greatly higher than the former attack.*

The concrete complexity of our attacks is summarized in Figure 1.3.

5.5 SPHINCS-FORS: A Provably Secure Hash-based Signature Scheme against Quantum Chosen Message Attacks

5.5.1 Discussion: How to Avoid Quantum Attacks?

In the previous section, we introduce quantum-access attacks on deterministic SPHINCS and SPHINCS+ and show that their security levels in the EU-qCMA/BU-qCMA model are much lower than that in the EU-CMA model. The key point of the above attacks is searching messages that map to some index idx^*. The search is done by iteratively running a function F in Grover's algorithm. A simple improvement to avoid these attacks is making F randomized. That is, in each signing operation, the signer adds a random nonce in calculating the pseudorandomness $z = \mathsf{PRF}_{\mathsf{msg}}(sk_{\mathsf{seed}}, m)$. This is indeed the probabilistic version of SPHINCS+ [17]. Note that this nonce is not necessary to be uniformly distributed. This nonce does not affect the security reduction of EU-CMA, but it affects EU-qCMA security.

However, we do not find security proof of EU-qCMA security for probabilistic SPHINCS or SPHINCS+, even if the random nonce is well sampled. Although the above attacks do not work on these probabilistic versions, there is no evidence to show that they are secure against *any* quantum chosen message attacks.

Note that the security under quantum chosen message attacks of SPHINCS and SPHINCS+ is more complicated than that in the classical setting for the following reasons. First, in the classical setting, a response of the signing query contains only one few-time signature. Since idx may differ in superpositions in the quantum-access setting, a SPHINCS/SPHINCS+ signature with quantum states may contain many few-time signatures for many key pairs. This multi-instance case exceeds our discussion about the security of few-time signature schemes. Second, a quantum SPHINCS/SPHINCS+ signature may also contain a large number of HT signatures σ_{HT} in superpositions. It makes the analysis even more difficult.

So how do we construct a *provably secure* HBS scheme under quantum-access attacks? Our solution is simple. The first step is to make each signing response only contain few-time signatures related to *one* key pair. For this purpose, we make the index of the few-time signature irrelevant to the message. In each signing query, the signing algorithm randomly picks a leaf from $\{0,1\}^h$ instead of running the pseudorandom function on the message. Since the randomness of a signing query is global, the resulting signatures in superpositions share common randomness and thus a common idx, implying a common few-time signature key pair. In addition, note that σ_{HT} is the signature on the few-time signature public key. Since all superpositions share a common public key, the resulting σ_{HT} is also identical in all superpositions. The security is then reduced to the quantum-access security of the few-time signature scheme in the single-instance case, which has been evaluated in Section 4.

This variant can avoid the above quantum-access attacks since the index is independent of the message. However, note that the random index needs to be directly contained in the signature, so an adversary can arbitrarily choose an index in the forgeries. It causes lower security than SPHINCS+, especially in the EU-CMA model. Thus, the classical security also needs re-analyzed.

In the next subsection, we present SPHINCS-FORS, a variant of SPHINCS+ that follows the approach and provides provable security in the EU-qCMA model.

```
$\mathsf{SPHINCS\text{-}FORS.KeyGen}(1^\lambda)$
    $sk_{\mathsf{seed}} \leftarrow \{0,1\}^n$,
    $(pk_{\mathsf{HT}}, sk_{\mathsf{HT}}) \leftarrow \mathsf{HT.KeyGen}(1^\lambda)$
    $sk = (sk_{\mathsf{seed}}, sk_{\mathsf{HT}})$, $pk = pk_{\mathsf{HT}}$
    **Output** $(pk, sk)$.

$\mathsf{SPHINCS\text{-}FORS.Sig}(sk, m)$
    $idx \leftarrow \{0,1\}^h$
    $s_{idx} = \mathsf{PRF}_{\mathsf{seed}}(sk_{\mathsf{seed}}, idx)$
    $(pk_{\mathsf{FORS}}, sk_{\mathsf{FORS}}) \leftarrow \mathsf{rFORS.KeyGen}(1^\lambda; s_{idx})$
    $(z, \sigma_{\mathsf{FORS}}) \leftarrow \mathsf{rFORS.Sig}(sk_{\mathsf{FORS}}, m)$
    $\sigma_{\mathsf{HT}} \leftarrow \mathsf{HT.Sig}(sk_{\mathsf{HT}}, pk_{\mathsf{FORS}}, idx)$
    **return** $(idx, z, \sigma_{\mathsf{HT}}, \sigma_{\mathsf{FORS}})$.

$\mathsf{SPHINCS\text{-}FORS.Ver}(pk, m, (idx, z, \sigma_{\mathsf{HT}}, \sigma_{\mathsf{FORS}}))$
    $pk_{\mathsf{FORS}} \leftarrow \mathsf{rFORS.pkFromSig}(m, (z, \sigma_{\mathsf{FORS}}))$
    **return** $\mathsf{HT.Ver}(pk_{\mathsf{HT}}, pk_{\mathsf{FORS}}, \sigma_{\mathsf{HT}}, idx)$
```

Figure 5.9: The framework of SPHINCS-FORS

5.5.2 SPHINCS-FORS

Now we introduce the variant of SPHINCS discussed by the end of the last section.

Construction 3 *Let PRF$_{seed}$: $\{0,1\}^n \times \{0,1\}^h \to \{0,1\}^n$ be a pseudorandom function, and HT and rFORS be depicted in above sections. SPHINCS-FORS is depicted in Figure 5.9.*

The difference with SPHINCS+ is as follows:

- The main difference is the strategy of choosing the index. In SPHINCS-FORS, the index is randomly chosen from $\{0,1\}^h$ while in SPHINCS+ it is pseudorandomly related to m. The motiation has been mentioned in

Subsection 5.5.1. The random index is directly contained in the signature in SPHINCS-FORS.

- In SPHINCS-FORS, we uses rFORS as the few-time signature. The slight difference is that in SPHINCS+, the (pseudo-)randomizer z is calculated by a global function $\mathsf{PRF}_{\mathsf{msg}}(\mathsf{sk}_{\mathsf{seed}}, m)$ while in SPHINCS-FORS, sk_{seed} differs in different indices. This will be helpful for security proofs.

5.5.3 Security Analysis

In this subsection, we analyze the security of SPHINCS-FORS under (quantum) chosen message attacks.

At first, we need to discuss the security for HT. As we use the same HT as SPHINCS+ and its (EU-sNACMA) security has been evaluated in [17, 69], we omit the formal analysis of HT. The success probability of breaking the security is at most $O(q^2/2^n)$, where q denotes the number of hash queries.

For a signature scheme Γ, let $\mathrm{InSec}^{*}_{\Gamma,r,q_H}(\xi)$ be the maximum of $\mathrm{Adv}^{*}_{\Gamma,r,q_H}(\mathcal{A})$ for all ξ-time adversary $\mathcal{A}$ and $* \in \{\mathsf{EU\text{-}CMA}, \mathsf{EU\text{-}qCMA}, \mathsf{EU\text{-}sNACMA}\}$.

We first prove the security under quantum chosen message attacks. We treat rFORS as a black box.

Theorem 24 *For any ξ-time adversary $\mathcal{A}$, it holds that*

$$\begin{aligned}Adv^{\textsf{EU-qCMA}}_{\textsf{SPHINCS-FORS},r,q_H}(\mathcal{A}) \leq InSec^{\textsf{EU-sNACMA}}_{\textsf{HT},2^h,q_H}(\xi) + InSec^{\textsf{Ind-PRF}}_{\textsf{PRF}_{\textsf{seed}},2^h}(\xi) \\ + \sum_{r=0}^{q_s} p(r,q_s) \cdot InSec^{\textsf{EU-qCMA}}_{\textsf{rFORS},r,q_H}(\xi),\end{aligned}$$

where $p(r,q_s) = \min\{2^{r(\log q_s - h)+h-\log r!}, 2^h\}$.

As mentioned in Section 5.5.1, all the signatures contained in a response from the signing oracle share a common index and thus a common rFORS key pair. In each superposition, idx and σ_{HT} are identical. The only "quantum state" of a response is $(z, \sigma_{\mathsf{FORS}})$, the signature of rFORS. It implies that the EU-qCMA security of SPHINCS-FORS is reduced to the EU-qCMA security of rFORS and the classical security of HT.

Proof.

The statement can be proved by the following hybrid arguments.

- **Game 0** is the original EU-qCMA experiment of SPHINCS-FORS.
- **Game 1** differs from **Game 0** in that, in the signing oracle, s_{idx} is calculated by $\mathsf{TRF}(idx)$ where $\mathsf{TRF} : \{0,1\}^h \to \{0,1\}^\lambda$ is a truly random function. If the success probability differs, it implies a reduction distinguishing $\mathsf{PRF}_{\mathsf{seed}}$ and TRF. Note that there are at most 2^h calls to PRF. We have

$$|\Pr[\textbf{Game 1}] - \Pr[\textbf{Game 0}]| \leq \mathrm{InSec}^{\mathsf{Ind\text{-}PRF}}_{\mathsf{PRF}_{\mathsf{seed}},2^h}(\xi). \tag{5.82}$$

- **Game 2** differs from **Game 1** as follows. After $\mathcal{A}$ outputs (q_s+1) message-signature pairs, **Game 2** checks whether there exists a forgery $(m^*, \Sigma^*) = (m^*, (idx^*, z, \sigma^*_{\mathsf{FORS}}, \sigma^*_{\mathsf{HT}}))$ such that $pk^*_{\mathsf{FORS}} \neq pk_{\mathsf{FORS}}$, where $s_{idx^*} = \mathsf{TRF}(idx^*)$, $pk^*_{\mathsf{FORS}} \leftarrow \mathsf{rFORS.pkFromSig}(m^*, (z, \sigma^*_{\mathsf{FORS}})), pk_{\mathsf{FORS}} \leftarrow \mathsf{rFORS.KeyGen}(1^n; s_{idx^*})$. If so, it returns 0.

  Game 2 differs from **Game 1** only if the adversary generates a HT signature for a "fake" pk^*_{FORS} which is not consistent to the real one. It implies a reduction attacking the EU-sNACMA security of HT. At the beginning, the reduction generates the rFORS public keys w.r.t. all the indices in $\{0,1\}^h$ and sends them with the corresponding indices to the challenger. Then, it obtains the HT signatures and the public key pk_{HT} from the challenger. When signing a message m from the adversary, it picks a random $idx \in \{0,1\}^h$, generates the corresponding rFORS signature $(z, \sigma_{\mathsf{FORS}})$. Then, it replies with $(idx, z, \sigma_{\mathsf{FORS}})$ and the corresponding σ_{HT} from the challenger. Finally, the adversary outputs a pk^*_{FORS} that is different from the real one. It implies a forgery of HT with state idx^*. We have

$$\Pr[\textbf{Game 1}] \leq \Pr[\textbf{Game 2}] + Adv^{\mathsf{EU\text{-}sNACMA}}_{\mathsf{HT},2^h,q_H}(\mathcal{A}). \tag{5.83}$$

  In **Game 2**, the adversary wins only if it generates (q_s+1) rFORS forgeries (of multiple instances) after q_s signing queries. Note that in each signing query, the signing oracle picks one $idx \in \{0,1\}^h$ and signs the message by the rFORS key pair associated with idx. Due to the pigeonhole principle, there must exist a special $idx^* \in \{0,1\}^h$ that has been used r times in signing queries and is used at least $(r+1)$ times in the forgeries for some $r \geq 0$.

- **Game 3** differs from **Game 2** in that it guesses $idx' \in \{0,1\}^h$ at the beginning of the experiment and outputs 0 if $idx' \neq idx^*$. We have

$$\Pr[\textbf{Game 2}] \leq 2^h \cdot \Pr[\textbf{Game 3}]. \tag{5.84}$$

- In **Game 3**, $\mathcal{A}$ wins if it generates $(r+1)$ forgeries for the rFORS key pair associated with idx' conditioned that it is chosen r times in q_s signing queries. It breaks the EU-qCMA security of rFORS with r signing queries. In addition, let $\mathbf{E}_r$ be the event that a leaf is chosen r times. The probability of $\mathbf{E}_r$ is $\binom{q_s}{r}(1-2^{-h})^{q_s-r}(2^{-h})^r < 2^{r(\log q_s - h) - \log r!}$. Thus, we have

$$\begin{aligned}\Pr[\textbf{Game 3}] &= \sum_{r=0}^{q_s} \Pr[\textbf{Game 3}|\mathbf{E}_r] \cdot \Pr[\mathbf{E}_r] \\ &\leq \sum_{r=0}^{q_s} \text{InSec}^{\text{EU-qCMA}}_{FORS,r,q_H}(\xi) \cdot \min\{2^{r(\log q_s - h) - \log r!}, 1\}.\end{aligned}$$

This completes the proof. □

From Corollary 3, we have

Corollary 5 *Let hash funcions in SPHINCS-FORS be modeled as quantum random oracles. It holds that*

$$Adv^{\textit{EU-qCMA}}_{\textit{SPHINCS-FORS},q_s,q_H}(\mathcal{A}) \leq O\left(\frac{q_H^2}{2^n} + \sum_{r=0}^{q_s} p(r,q_s) \cdot \min\left\{q_H^{2(r+1)}\left(\frac{r^2}{t}\right)^k + \frac{q_H^2 k t^r}{2^n}, 1\right\}\right). \tag{5.85}$$

In terms of EU-CMA security, the proof is very similar to EU-qCMA and thus omitted.

Theorem 25 *For any ξ-time adversary $\mathcal{A}$, it holds that*

$$Adv^{\textit{EU-CMA}}_{\textit{SPHINCS-FORS},r,q_H}(\mathcal{A}) \leq \textit{InSec}^{\textit{EU-sNACMA}}_{\textit{HT},2^h,q_H}(\xi) + \textit{InSec}^{\textit{Ind-PRF}}_{\textit{PRF}_{seed},2^h}(\xi) + \sum_{r=0}^{q_s} p(r) \cdot \textit{InSec}^{\textit{EU-CMA}}_{\textit{rFORS},r,q_H}(\xi),$$

where $p(r, q_s) = \min\{2^{r(\log q_s - h) + h - \log r!}, 2^h\}$.

From Theorem 4, we have

Corollary 6 *Let hash funcions in SPHINCS-FORS be modeled as quantum random oracles. It holds that*

$$Adv^{\textit{EU-CMA}}_{\textit{SPHINCS-FORS},q_s,q_H}(\mathcal{A}) \leq O\left(q_s\sqrt{\frac{q_H + q_s}{2^n}} + \frac{q_H^2}{2^n} + q_H^2 \sum_{r=0}^{q_s} p(r, q_s)\left(\frac{r}{t}\right)^k\right). \quad (5.86)$$

Note that here the term caused by adaptive reprogramming is $O\big(q_s\sqrt{\frac{q_H+q_s}{2^n}}\big)$ rather than $\sum_{r=0}^{q_s} p(r, q_s) \cdot \frac{3r}{2}\sqrt{\frac{q_H+r+1}{2^n}}$. It is because that the random oracle is reprogrammed at most q_s times in the reduction. For the same reason the terms caused by SM-TCR, SM-DSPR and Ind-PRF are gathered to $O(q_H^2/2^n)$.

5.5.4 Concrete Security

As a variant of SPHINCS+, SPHINCS-FORS is expected to reach the same security level as SPHINCS+. However, we note that the EU-CMA security level of SPHINCS-FORS is lower than SPHINCS+ with the same parameters. It is because of the different strategies of choosing the index. Since the index is directly contained in the signature (just like what SPHINCS does), the adversary is able to arbitrarily choose a target index to forge a signature. Recall that in SPHINCS+, the index can be verified by the message and thus makes the adversary harder to succeed. In the next section, we show a concrete attack on SPHINCS-FORS, implying the difference in security levels with SPHINCS+.

For example, in SPHINCS+-256s, $h = 64$, $k = 22$, $t = 14$, and $q_s = 2^{64}$. The expected number of q_H breaking EU-CMA security is 2^{128}. With the same parameters on SPHINCS-FORS, q_H is only 2^{96}, much lower than what we expect.

Parameters	$\log q_s$	$\log q_H$
SPHINCS-FORS-256s ($n = 256, h = 64, t = 2^{17}, k = 22$)	10	91
	20	88
	30	86
	40	52
	50	28
	60	10
SPHINCS-FORS-256f ($n = 256, h = 68, t = 2^{11}, k = 32$)	10	93
	20	91
	30	88
	40	50
	50	32
	60	13

Figure 5.10: Concrete security against qCMA of SPHINCS-FORS. q_H denotes a lower bound of the hash queries needed in all quantum-access attacks.

Thus, we suggest the parameters be $h = 64$, $k = 22$, and $t = 2^{17}$. When $q_s = 2^{64}$, $\sum_{r=0}^{q_s} p(r, q_s)(\frac{r}{t})^k$ in the equation (5.86) becomes approximately 2^{-256}. The scheme can then provide 128-bit security against chosen message attacks. This causes approximately 7.09% larger signatures.

In addition, we adapt the parameters in SPHINCS+-256f from $h = 68$, $k = 32$, $t = 2^9$ to $h = 68$, $k = 32$, $t = 2^{11}$ and obtain a version SPHINCS-FORS-256f. This causes approximately 4.49% larger signatures than SPHINCS+-256f.

As for EU-qCMA security, our reduction provides security levels in Figure 5.10 by Corollary 5.

We observe that the EU-qCMA security does not provide 128-bit security even if q_s is lower than 2^{64}. Accordingly, one can flexibly adapt the parameters to provide higher security levels. As a result, the signature size and running time will become larger. For instance, our attack in the previous section shows that the quantum-access bit security of SPHINCS+-256s is at most 80 when $q_s = 2^{48}$.

We can set the parameters ($k = 32$, $t = 2^{17}$, $h = 128$, $n = 256$, SPHINCS-FORS v.2 in Figure 1.4) to achieve the same security level with security proof. It implies that the bit security is at least (rather than at most) 80. The price is that the signature size is larger than SPHINCS+-256s. In addition, we can simply increase the parameters of SPHINCS+-256s so that our attacks require $q_s = 2^{64}$ and $q_H = 2^{128}$ (say SPHINCS+-256s* in Figure 1.4). Similarly, we can also set the parameters ($k = 48$, $t = 2^{18}$, $h = 128$, $n = 384$, SPHINCS-FORS v.3 in Figure 1.4) to achieve the same (but provable) security level.

See details in Figure 1.4 (SPHINCS-FORS-256s is written as v.1 in this figure).

Remark 14 *Note that the EU-qCMA security of SPHINCS-FORS is reduced to EU-qCMA security of rFORS. However, our security bound for rFORS seems non-tight since the reduction even does not require any security notion for the pseudorandom function inside of rFORS. It is an open question to find a tighter reduction for rFORS. We conjecture that the security of rFORS can be bounded by $O(q^2(\frac{r^2}{t})^k + q^2/2^n)$ conditioned that the hash functions are modeled as quantum random oracles, and the pseudorandom function is quantum-access secure [106]. If it is true, then the security level of SPHINCS-FORS will be much higher than what we depicted (it then provides 96-bit EU-qCMA security when $q_s = 2^{64}$, security level 4 in NIST standardization).*

5.6 An Attack on SPHINCS-FORS

This subsection shows a (classical) chosen message attack on SPHINCS-FORS focusing on breaking eTSR. It is similar to the attack on SPHINCS+ focusing on ITSR, but there are some slight differences. This implies the difference in security levels between SPHINCS-FORS and SPHINCS+ in the EU-CMA model and that the EU-CMA security bound of SPHINCS-FORS is tight.

First, the adversary queries arbitrary q_s messages to the signing oracle and obtains the corresponding sigantures. Let S_j be the set of labels of preimages contained in the signature of m_j, that is, $S_j = \{(idx_j, i, h_i(z_j||m_j))\}_{i\in[k]}$.

After that, the adversary uses Grover's algorithm to find (z^*, m^*) *such that* $\exists idx^* \in \{0,1\}^h : \{(idx^*, i, h_i(z^*||m^*))\}_{i\in[k]} \subset \bigcup_{j=1}^{q_s} S_j$ *and that* m^* *has not been queried. Then,* $(m^*, (idx^*, z^*, \sigma^*_{HT}, \sigma^*_{FORS}))$ *is a forgery where* σ^*_{HT} *and* σ^*_{FORS} *can be calculated by the signatures received from the signing oracle.*

Next, we evaluate the complexity of hash queries needed in the above attack. For $idx \in \{0,1\}^h$*, let* Y_{idx} *be the set of* $y = y_1||...||y_k$ *such that* $\{(idx, i, y_i)\}_{i\in[k]} \subset S = \bigcup_{j=1}^{q_s} S_j$*. Let* $\boldsymbol{E}_{idx,\gamma}$ *be the event that* idx *is picked* γ *times. We have* $\Pr[\boldsymbol{E}_{idx,\gamma}] = \binom{q_s}{\gamma}(1-\frac{1}{2^h})^{q_s-\gamma}\frac{1}{2^{h\gamma}}$ *and*

$$\mathbb{E}[|Y_{idx}|] = \left(1-\left(1-\frac{1}{t}\right)^{\gamma}\right)^{k}\binom{q_s}{\gamma}\left(1-\frac{1}{2^h}\right)^{q_s-\gamma}\frac{t^k}{2^{h\gamma}}. \tag{5.87}$$

Thus,

$$\mathbb{E}\left[\sum_{idx}|Y_{idx}|\right] = \left(1-\left(1-\frac{1}{t}\right)^{\gamma}\right)^{k}\binom{q_s}{\gamma}\left(1-\frac{1}{2^h}\right)^{q_s-\gamma}\frac{t^k}{2^{\gamma}}. \tag{5.88}$$

Let $\Gamma = \big(1-\big(1-\frac{1}{t}\big)^{\gamma}\big)^{k}\binom{q_s}{\gamma}\big(1-\frac{1}{2^h}\big)^{q_s-\gamma}\frac{1}{2^{h\gamma}}$*. We have* $\mathbb{E}\big[\sum_{idx}|Y_{idx}|\big] = t^k 2^h \Gamma$*. In the average case, the probability of the above Grover's search is* $\mathbb{E}\big[\sum_{idx}|Y_{idx}|\big]/t^k = 2^h\Gamma$*. The number of quantum hash queries is approximately* $O(2^{-\frac{h}{2}}\Gamma^{-\frac{1}{2}})$*.*

Note that in SPHINCS+, the quantum hash query complexity needed in breaking ITSR is $O(\Gamma^{-\frac{1}{2}})$*. Thus, with the same parameters, the quantum bit security of SPHINCS-FORS is* $h/2$ *lower than SPHINCS+.*

5.7 Conclusion and Open Questions

This work analyzes the security of stateless HBS schemes, especially in terms of quantum-access security. First, for few-time signature schemes, we prove the concrete security levels of the schemes and the quantum-access security of FORS. Then, we turn to quantum-access security of many-time HBS schemes. We show quantum-access attacks (or superposition attacks) on HBS schemes, such as SPHINCS and SPHINCS+. The time complexity of the quantum-access attacks is lower than the optimal attacks in the classical setting. Next, we propose a variant of SPHINCS+ called SPHINCS-FORS. It is a provably secure HBS scheme against quantum chosen message attacks. As far as we know, it is the first practical HBS scheme with provable security against quantum-access attacks.

Note that our quantum-access attacks do not work on the probabilistic version of SPHINCS+. Since there is no security proof, it is an open question whether probabilistic SPHINCS+ is secure in the quantum-access setting. In addition, our security proof of SPHINCS-FORS against quantum chosen message attacks is possibly non-tight. It shows a lower bound of the security, but we are unaware of any concrete attacks that reach this bound. It is also an open question whether we can get a tighter reduction or if there exists a better quantum-access attack.

Chapter 6

Conclusion

Digital signature schemes are widely used for authentications in various situations. In the post-quantum setting, hash-based signature (HBS) schemes are attractive due to the minor assumptions and short key sizes. It is significant to carefully analyze the security of HBS schemes before we use them. Our security analysis is two-fold. First, since the security is based on security notions for hash functions, it is necessary to have a comprehensive understanding of these notions. Second, considering various situations and attacks, it is necessary to guarantee security in various security models.

In this dissertation, we finish the security analysis from the fundamental assumptions to upper constructions.

In Chapter 3, we analyze the security of subset-resilient hash function families (SRH), a building block of (stateless) HBS schemes. First, we give a generic quantum attack on SRH, implying an upper bound of the security level. Second, we prove the existence of decisional-collision-resistant hash function families by assuming the existence of SRH. Third, we prove the fully black-box separation from one-way permutations. The above two statements confirm the existential "position" in the cryptographic worlds of hash functions.

In Chapter 4, we give fine-grained security notions for stateful signature schemes. It is inspired by the fact that many HBS schemes are stateful and the state may be more vulnerable than the seret keys in practice. We consider adversaries with different control abilities over the state. Then, we show some separations and general conversions among these security notions. Some of the models can be considered bridges between stateful and stateless signature schemes. Finally, we give some general constructions of stateful signature schemes from a one-time signature scheme, a fundamental primitive in cryptography.

In Chapter 5, we analyze the quantum-access security of stateless HBS schemes. Quantum-access security is a strictly stronger notion than classical ones, where the adversary can send messages with superpositions to the signing oracle. Our analysis is in two directions. First, we show quantum-access attacks on SPHINCS and SPHINCS+, two practical stateless HBS schemes. The time complexity is lower than the optimal attacks in the classical setting. Second, we propose a new HBS scheme with provable security against quantum-access attacks. As far as we know, this is the first practical HBS scheme whose quantum-access security is provable.

In summary, one can have a deeper understanding of HBS schemes from this dissertation, especially in terms of security levels. On the one hand, we need to have a comprehensive and accurate knowledge of the security of the underlying hash functions before we use them as the building blocks. On the other hand, when analyzing the security of a cryptographic scheme, we need to consider different adversaries in various situations in practice. The classical security models may be not enough to analyze the security against these attacks.

Publication List

Journal Articles

- Quan Yuan, Mehdi Tibouchi, Masayuki Abe, ”On subset-resilient hash function families”. Designs, Codes and Cryptography. 90, pages 719–758, 2022
- Quan Yuan, Mehdi Tibouchi, Masayuki Abe, ”Security Notions for Stateful Signature Schemes”. IET Information Security. Volume 16, Issue 1. pp. 1-17. Jan 2022

Talks

- Quan Yuan, Mehdi Tibouchi, Masayuki Abe. ”Quantum-Accessible Security of Stateless Hash-based Signature Schemes”, SCIS 2022.
- Quan Yuan, Mehdi Tibouchi, Masayuki Abe. ”Security Notions of Stateful Signature Schemes”, SCIS 2021.

Bibliography

[1] Hash-based signatures revisited: A dynamic FORS with adaptive chosen message security. In *Progress in Cryptology - AFRICACRYPT 2020* (2020), vol. 12174 of *LNCS*, Springer, pp. 239–257.

[2] Aaronson, S., and Shi, Y. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM (JACM) 51*, 4 (2004), 595–605.

[3] Alagic, G., Majenz, C., Russell, A., and Song, F. Quantum-access-secure message authentication via blind-unforgeability. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2020), Springer, pp. 788–817.

[4] Ambainis, A. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing 37*, 1 (2007), 210–239.

[5] Ambainis, A., Hamburg, M., and Unruh, D. Quantum security proofs using semi-classical oracles. In *Annual International Cryptology Conference* (2019), Springer, pp. 269–295.

[6] Asharov, G., and Segev, G. Limits on the power of indistinguishability obfuscation and functional encryption. *SIAM Journal on Computing 45*, 6 (2016), 2117–2176.

[7] Aumasson, J.-P., Bernstein, D. J., Beullens, W., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.-L., Hülsing, A., Kampanakis, P.,

Kölbl, S., Lange, T., Lauridsen, M. M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., and Westerbaan, B. SPHINCS+ - submission to the nist post-quantum project v.3.

[8] Aumasson, J.-P., and Endignoux, G. Clarifying the subset-resilience problem. Cryptology ePrint Archive, Report 2017/909.

[9] Aumasson, J.-P., and Endignoux, G. Improving stateless hash-based signatures. In *Topics in Cryptology - CT-RSA 2018* (2018), vol. 10808 of *LNCS*, Springer, pp. 219–242.

[10] Barić, N., and Pfitzmann, B. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology - EUROCRYPT' 97* (1997), vol. 1233 of *LNCS*, Springer, pp. 480–494.

[11] Bellare, M., and Rogaway, P. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security* (1993), pp. 62–73.

[12] Bellare, M., and Rogaway, P. The exact security of digital signatures-how to sign with rsa and rabin. In *International conference on the theory and applications of cryptographic techniques* (1996), Springer, pp. 399–416.

[13] Belovs, A. Learning-graph-based quantum algorithm for k-distinctness. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science* (2012), IEEE, pp. 207–216.

[14] Berman, I., Degwekar, A., Rothblum, R. D., and Vasudevan, P. N. Multi-collision resistant hash functions and their applications. In *Advances in Cryptology –EUROCRYPT 2018* (2018), vol. 10821 of *LNCS*, Springer, pp. 133–161.

[15] Bernstein, D. J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schwabe, M. S., and Wilcox-O'Hearn, Z. SPHINCS: Practical stateless hash-based signatures. In *Advances in*

Cryptology - EUROCRYPT 2015 (2015), vol. 9056 of *LNCS*, Springer, pp. 368–397.

[16] Bernstein, D. J., and Hülsing, A. Decisional second-preimage resistance: when does spr imply pre? In *International Conference on the Theory and Application of Cryptology and Information Security* (2019), Springer, pp. 33–62.

[17] Bernstein, D. J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., and Schwabe, P. The SPHINCS+ signature framework. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019), Association for Computing Machinery, pp. 2129–2146.

[18] Bitansky, N., Haitner, I., Komargodski, I., and Yogev, E. Distributional collision resistance beyond one-way functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2019), Springer, pp. 667–695.

[19] Bitansky, N., Kalai, Y. T., and Paneth, O. Multi-collision resistance: a paradigm for keyless hash functions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (2018), pp. 671–684.

[20] Bleichenbacher, D., and Maurer, U. M. Directed acyclic graphs, one-way functions and digital signatures. In *Advances in Cryptology - CRYPTO '94* (1994), vol. 839 of *LNCS*, Springer, pp. 75–82.

[21] Böhl, F., Hofheinz, D., Jager, T., Koch, J., Seo, J. H., and Striecks, C. Practical signatures from standard assumptions. In *Advances in Cryptology - EUROCRYPT 2013* (2013), vol. 7881 of *LNCS*, Springer, pp. 461–485.

[22] Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., and Zhandry, M. Random oracles in a quantum world. In *International conference on the theory and application of cryptology and information security* (2011), Springer, pp. 41–69.

[23] Boneh, D., and Zhandry, M. Quantum-secure message authentication codes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2013), Springer, pp. 592–608.

[24] Boneh, D., and Zhandry, M. Secure signatures and chosen ciphertext security in a quantum computing world. In *Advances in Cryptology - CRYPTO 2013* (2013), vol. 8043 of *LNCS*, Springer, pp. 361–379.

[25] Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., and Schrottenloher, A. Quantum attacks without superposition queries: the offline Simon's algorithm. In *International Conference on the Theory and Application of Cryptology and Information Security* (2019), Springer, pp. 552–583.

[26] Bos, J. W., Hülsing, A., Renes, J., and van Vredendaal, C. Rapidly verifiable XMSS signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), 137–168.

[27] Brassard, G., Høyer, P., and Tapp, A. Quantum cryptanalysis of hash and claw-free functions. In *Latin American Symposium on Theoretical Informatics* (1998), Springer, pp. 163–169.

[28] Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., and Rückert, M. On the security of the winternitz one-time signature scheme. In *International conference on cryptology in Africa* (2011), Springer, pp. 363–378.

[29] Buchmann, J., Dahmen, E., and Hülsing, A. XMSS - a practical forward secure signature scheme based on minimal security assumptions. In *PQCrypto 2011: Post-Quantum Cryptography* (2011), vol. 7071 of *LNCS*, Springer, pp. 117–129.

[30] Buchmann, J., García, L. C. C., Dahmen, E., Döring, M., and Klintsevich, E. CMSS - an improved merkle signature scheme. In *Progress in Cryptology - INDOCRYPT06* (2007), vol. 4329 of *LNCS*, Springer, pp. 349–363.

[31] Buchmann, J., Klintsevich, E. D., Okeya, K., and Vuillaume, C. Merkle signatures with virtually unlimited signature capacity. In *ACNS 2007: Applied Cryptography and Network Security* (2011), vol. 4521 of *LNCS*, Springer, pp. 31–45.

[32] Castelnovi, L., Martinelli, A., and Prest, T. Grafting trees: a fault attack against the SPHINCS framework. In *International Conference on Post-Quantum Cryptography* (2018), Springer, pp. 165–184.

[33] Chailloux, A., Naya-Plasencia, M., and Schrottenloher, A. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *Advances in Cryptology – ASIACRYPT 2017* (2017), vol. 10625 of *LNCS*, Springer, pp. 211–240.

[34] Chatterjee, R., Chung, K.-M., Liang, X., and Malavolta, G. A note on the post-quantum security of (ring) signatures. In *IACR International Conference on Public-Key Cryptography* (2022), Springer, pp. 407–436.

[35] Coron, J.-S. On the exact security of full domain hash. In *Annual International Cryptology Conference* (2000), Springer, pp. 229–235.

[36] Cramer, R., Damgård, I., and Pedersen, T. Efficient and provable security amplifications. In *International Workshop on Security Protocols* (1996), Springer, pp. 101–109.

[37] Cramer, R., and Damgård, I. New generation of secure and practical rsa-based signatures. In *Advances in Cryptology - CRYPTO' 96* (1996), vol. 1109 of *LNCS*, Springer, pp. 173–185.

[38] Cremers, C., Düzlü, S., Fiedler, R., Fischlin, M., and Janson, C. Buffing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy (SP)* (2021), IEEE, pp. 1696–1714.

[39] Dahmen, E., Okeya, K., Takagi, T., and Vuillaume, C. Digital signatures out of second-preimage resistant hash functions. In *PQCrypto 2008:*

Post-Quantum Cryptography (2008), vol. 5299 of *LNCS*, Springer, pp. 109–123.

[40] Damgård, I., Funder, J., Nielsen, J. B., and Salvail, L. Superposition attacks on cryptographic protocols. In *International Conference on Information Theoretic Security* (2013), Springer, pp. 142–161.

[41] Damgård, I. B. Collision free hash functions and public key signature schemes. In *Advances in Cryptology - EUROCRYPT' 87* (1988), vol. 304 of *LNCS*, Springer, pp. 203–216.

[42] Ding, J., and Schmidt, D. Rainbow, a new multivariable polynomial signature scheme. In *International conference on applied cryptography and network security* (2005), Springer, pp. 164–175.

[43] Dods, C., Smart, N., and Stam, M. Hash based digital signature schemes. In *Cryptography and Coding* (2005), vol. 3796 of *LNCS*, Springer Verlag, pp. 98–115.

[44] Dubrov, B., and Ishai, Y. On the randomness complexity of efficient sampling. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing* (2006), STOC '06, Association for Computing Machinery, p. 711–720.

[45] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and Stehlé, D. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 238–268.

[46] Dwork, C., and Naor, M. An efficient existentially unforgeable signature scheme and its applications. *Journal of Cryptology 11*, 3 (1998), 187–208.

[47] ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory 31*, 4 (1985), 469–472.

[48] Erdös, P., Frankl, P., and Füredi, Z. Families of finite sets in which no set is covered by the union of r others. In *Israel Journal of Mathematics* (1985), vol. 51, pp. 79–89.

[49] Even, S., Goldreich, O., and Micali, S. On-line/off-line digital signatures. In *Conference on the Theory and Application of Cryptology* (1989), Springer, pp. 263–275.

[50] Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., and Zhang, Z. Falcon: Fast-fourier lattice-based compact signatures over ntru. *Submission to the NIST' s post-quantum cryptography standardization process 36*, 5 (2018).

[51] Gaborit, P., and Schrek, J. Efficient code-based one-time signature from automorphism groups with syndrome compatibility. In *2012 IEEE International Symposium on Information Theory Proceedings* (2012), pp. 1982–1986.

[52] Gagliardoni, T., Hülsing, A., and Schaffner, C. Semantic security and indistinguishability in the quantum world. In *Annual international cryptology conference* (2016), Springer, pp. 60–89.

[53] Garg, S., Yuen, H., and Zhandry, M. New security notions and feasibility results for authentication of quantum data. In *Annual International Cryptology Conference* (2017), Springer, pp. 342–371.

[54] Gennaro, R., Gertner, Y., Katz, J., and Trevisan, L. Bounds on the efficiency of generic cryptographic constructions. *SIAM journal on Computing 35*, 1 (2005), 217–246.

[55] Gennaro, R., and Trevisan, L. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science, FOCS2000* (2000), IEEE, pp. 305–313.

[56] Goldreich, O. *Foundations of Cryptography: Volume 2, Basic Applications*. Campbridge University Press, Cambridge, UK, 2004.

[57] Goldreich, O., Goldwasser, S., and Micali, S. How to constuct random functions (extended abstract). In *FOCS* (1984), pp. 464–479.

[58] Goldwasser, S., Micali, S., and Rivest, R. L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on computing 17*, 2 (1988), 281–308.

[59] Grilo, A. B., Hövelmanns, K., Hülsing, A., and Majenz, C. Tight adaptive reprogramming in the qrom. In *International Conference on the Theory and Application of Cryptology and Information Security* (2021), Springer, pp. 637–667.

[60] Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (1996), pp. 212–219.

[61] Haitner, I., Hoch, J. J., Reingold, O., and Segev, G. Finding collisions in interactive protocols—tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM Journal on Computing 44*, 1 (2015), 193–242.

[62] Håstad, J., Impagliazzo, R., Levin, L. A., and Luby, M. A pseudorandom generator from any one-way function. *SIAM Journal on Computing 28*, 4 (1999), 1364–1396.

[63] Hofheinz, D., Jager, T., and Kiltz, E. Short signatures from weaker assumptions. In *Advances in Cryptology - ASIACRYPT 2011* (2011), vol. 7073 of *LNCS*, Springer, pp. 647–666.

[64] Hosoyamada, A., and Iwata, T. 4-round luby-rackoff construction is a qprp. In *International Conference on the Theory and Application of Cryptology and Information Security* (2019), Springer, pp. 145–174.

[65] Hosoyamada, A., and Sasaki, Y. Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In *Cryptographers' Track at the RSA Conference* (2018), Springer, pp. 198–218.

[66] Hosoyamada, A., Sasaki, Y., and Xagawa, K. Quantum multicollision-finding algorithm. In *Advances in Cryptology - ASIACRYPT 2017* (2017), vol. 10625 of *LNCS*, Springer, pp. 179–210.

[67] Hülsing, A. W-OTS+ - shorter signatures for hash-based signature schemes. In *Progress in Cryptology - AFRICACRYPT 2013* (2013), vol. 7918 of *LNCS*, Springer, pp. 173–188.

[68] Hülsing, A., Busold, C., and Buchmann, J. Forward secure signatures on smart cards. In *International Conference on Selected Areas in Cryptography* (2012), Springer, pp. 66–80.

[69] Hülsing, A., and Kudinov, M. Recovering the tight security proof of SPHINCS+. *Cryptology ePrint Archive* (2022).

[70] Hülsing, A., Rausch, L., and Buchmann, J. Optimal parameters for XMSS^{MT}. In *CD-ARES 2013: Security Engineering and Intelligence Informatics* (2013), vol. 8128 of *LNCS*, Springer, pp. 194–208.

[71] Hülsing, A., Rijneveld, J., and Schwabe, P. Armed SPHINCS. In *Public-Key Cryptography–PKC 2016*. Springer, 2016, pp. 446–470.

[72] Hülsing, A., Rijneveld, J., and Song, F. Mitigating multi-target attacks in hash-based signatures. In *Public-Key Cyptography - PKC 2016* (2016), vol. 9614 of *LNCS*, Springer, pp. 387–416.

[73] Impagliazzo, R. A personal view of average-case complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference* (1995), IEEE, pp. 134–147.

[74] Impagliazzo, R., and Rudich, S. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing* (1989), pp. 44–61.

[75] Kannwischer, M. J., Genêt, A., Butin, D., Krämer, J., and Buchmann, J. Differential power analysis of XMSS and SPHINCS. In *International Workshop on Constructive Side-Channel Analysis and Secure Design* (2018), Springer, pp. 168–188.

[76] Kaplan, M., Leurent, G., Leverrier, A., and Naya-Plasencia, M. Breaking symmetric cryptosystems using quantum period finding. In *Annual international cryptology conference* (2016), Springer, pp. 207–237.

[77] Komargodski, I., Naor, M., and Yogev, E. Collision resistant hashing for paranoids: Dealing with multiple collisions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2018), Springer, pp. 162–194.

[78] Komargodski, I., and Yogev, E. On distributional collision hashing. In *Advances in Cryptology - CRYPTO 2018* (2018), vol. 10992 of *LNCS*, Springer, pp. 303–327.

[79] Kuwakado, H., and Morii, M. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *2010 IEEE International Symposium on Information Theory* (2010), IEEE, pp. 2682–2685.

[80] Kuwakado, H., and Morii, M. Security on the quantum-type even-mansour cipher. In *2012 International Symposium on Information Theory and its Applications* (2012), IEEE, pp. 312–316.

[81] Lamport, L. Constructing digital signatures from a one way function. Tech. rep., Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[82] Leander, G., and May, A. Grover meets Simon–quantumly attacking the fx-construction. In *International Conference on the Theory and Applica-*

tion of Cryptology and Information Security (2017), Springer, pp. 161–178.

[83] Leighton, F. T., and Micali, S. Large provably fast and secure digital signature schemes based on secure hash functions. https://www.google.com/patents/US5432852, US Patent 5,432,852.

[84] Liu, Q., and Zhandry, M. On finding quantum multi-collisions. In *Advances in Cryptology - EUROCRYPT 2019* (2019), vol. 11478 of *LNCS*, Springer, pp. 189–218.

[85] Lyubashevsky, V., and Micciancio, D. Asymptotically efficient lattice-based digital signatures. In *TCC 2008: Theory of Cryptography* (2008), vol. 4948 of *LNCS*, Springer, pp. 37–54.

[86] Majenz, C., Manfouo, C. M., and Ozols, M. Quantum-access security of the winternitz one-time signature scheme. *arXiv preprint arXiv:2103.12448* (2021).

[87] Merkle, R. C. A certified digital signature. In *Advances in Cryptology - CRYPTO '89* (1989), G. Brassard, Ed., vol. 435 of *LNCS*, Springer, pp. 218–238.

[88] Naor, M. Bit commitment using pseudorandomness. *Journal of cryptology 4*, 2 (1991), 151–158.

[89] Nielsen, M. A., and Chuang, I. Quantum computation and quantum information, 2000.

[90] Perrig, A. The biba one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM conference on Computer and Communications Security* (2001), Springer, pp. 28–37.

[91] Persichetti, E. Efficient one-time signatures from quasi-cyclic codes: A full treatment. *Cryptography 2* (2018), 30.

[92] Pieprzyk, J., Wang, H., and Xing, C. Multiple-time signature schemes against adaptive chosen message attacks. In *SAC 2003: Selected Areas in Cryptography* (2003), vol. 3006 of *LNCS*, Springer, pp. 88–100.

[93] Poettering, B., and Stebila, D. Double-authentication-preventing signatures. In *Computer Security - ESORICS 2014* (2014), vol. 8712 of *LNCS*, Springer, pp. 436–453.

[94] Reyzin, L., and Reyzin, N. Better than biba: Short one-time signatures with fast signing and verifying. In *ACISP 2002: Information Security and Privacy* (2002), vol. 2384 of *LNCS*, Springer, pp. 144–153.

[95] Rompel, J. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing* (1990), pp. 387–394.

[96] Schnorr, C.-P. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology* (1989), Springer, pp. 239–252.

[97] Seo, J. H. Short signatures from diffie–hellman: Realizing almost compact public key. *Journal of Cryptology 30* (2017), 735–759.

[98] Seo, J. H. Efficient digital signatures from rsa without random oracles. *Information Sciences 512* (2020), 471–480.

[99] Shor, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science* (1994), Ieee, pp. 124–134.

[100] Simon, D. R. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology –EUROCRYPT'98* (1998), vol. 1403 of *LNCS*, Springer, pp. 334–345.

[101] Suzuki, K., Tonien, D., Kurosawa, K., and Toyota, K. Birthday paradox for multi-collisions. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 91*, 1 (2008), 39–45.

[102] Unruh, D. Revocable quantum timed-release encryption. *Journal of the ACM (JACM) 62*, 6 (2015), 1–76.

[103] Yamakawa, T., and Zhandry, M. Classical vs quantum random oracles. In *Advances in Cryptology - EUROCRYPT 2021* (2021), vol. 12697 of *LNCS*, Springer, pp. 568–597.

[104] Yehia, M., AlTawy, R., and Gulliver, T. A. Verifiable obtained random subsets for improving SPHINCS+. In *Australasian Conference on Information Security and Privacy* (2021), Springer, pp. 694–714.

[105] Zaverucha, G. M., and Stinson, D. R. Short one-time signatures. *Advances in Mathematics of Communications 5*, 3 (2011), 473.

[106] Zhandry, M. How to construct quantum random functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science* (2012), pp. 679–687.

[107] Zhandry, M. A note on the quantum collision and set equality problems. *Quantum Information and Computation 15*, 7-8 (2015), 557–567.

[108] Zhandry, M. How to record quantum queries, and applications to quantum indifferentiability. In *Annual International Cryptology Conference* (2019), Springer, pp. 239–268.

[109] Zhang, K., Cui, H., and Yu, Y. SPHINCS-α: A compact stateless hash-based signature scheme. *Cryptology ePrint Archive* (2022).