Designs, Codes and Cryptography (2022) 90:719–758
https://doi.org/10.1007/s10623-022-01008-4

On subset-resilient hash function families

Quan Yuan[1] · **Mehdi Tibouchi**[1,2] · **Masayuki Abe**[1,2]

Received: 18 August 2021 / Revised: 3 January 2022 / Accepted: 7 January 2022 /
Published online: 6 February 2022

Abstract
In this paper, we analyze the security of subset-resilient hash function families, which is first proposed as a requirement of a hash-based signature scheme called HORS. Let $\mathcal{H}$ be a family of functions mapping an element to a subset of size at most k. (r, k)-subset resilience guarantees that given a random function H from $\mathcal{H}$, it is hard to find an $(r+1)$-tuple $(x, x_1, \ldots, x_r)$ such that (1) $H(x)$ is covered by the union of $H(x_i)$ and (2) x is not equal to any x_i. Subset resilience and its variants are related to nearly all existing stateless hash-based signature schemes, but the power of this security notion is lacking in research. We present three results on subset resilience. First, we show a generic quantum attack against subset resilience, whose time complexity is smaller than simply implementing Grover's search. Second, we show that subset-resilient hash function families imply the existence of distributional collision-resistant hash function families. Informally, distributional collision resistance is a relaxation of collision resistance, which guarantees that it is hard to find a *uniform* collision for a hash function. This result implies a comparison among the power of subset resilience, collision resistance, and distributional collision resistance. Third, we prove the fully black-box separation from one-way permutations.

Keywords Subset resilience · Hash functions · Quantum attacks · Distributional collision resistance · Black-box separation · Hash-based signature schemes

Mathematics Subject Classification 94A60

Communicated by D. Stebila.

✉ Quan Yuan
yuan.quan.87x@st.kyoto-u.ac.jp

Mehdi Tibouchi
mehdi.tibouchi.br@hco.ntt.co.jp

Masayuki Abe
masayuki.abe.cp@hco.ntt.co.jp

[1] Kyoto University, Kyoto, Japan

[2] NTT Laboratories, Tokyo, Japan

1 Introduction

The digital signature scheme is an essential primitive in cryptography. Usually, a provably secure signature scheme needs to be based on the hardness of mathematical problems. However, due to the researches of quantum computations, the hardness of these problems becomes vulnerable. For example, RSA and CDH can be solved by Shor's algorithm [37]. Even though there are still several hard problems that are not broken by quantum computers until now, no one knows whether they can keep secure in the following decades. A hash-based signature scheme (HBS) becomes an attractive choice in this setting. HBS is based on assumptions of hash functions rather than the hardness of mathematical problems.

When we analyze the security of HBS, it is necessary to confirm the security notions for hash functions. Note that different HBS requires different security notions for hash functions. For example, the security of Lamport's scheme [30] requires one-wayness, and Merkle's signature scheme [34] requires one-wayness and collision resistance. These signature schemes are stateful, meaning that the signer must maintain a dynamic state during multiple signing executions. When it comes to practical stateless HBS, it usually requires a particular security notion for hash function families, called subset resilience.

Subset-resilient hashing (SRH) is a kind of hash function families, which is first proposed in [36] as a building block of HORS (Hash to Obtain Random Set), a stateless few-time HBS. 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 one element of the domain to a *subset* of size at most k. We call $(x, x_1, x_2, \ldots, x_r)$ be 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 $i = 1, \ldots, r$. 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 "short" hash functions $(h_1, \ldots, h_k)$. When sampling H, it samples $h_1, \ldots, h_k$ from the hash function family $\mathcal{H} = \{h : \{0,1\}^{m(n)} \to \{0,1\}^{l(n)}\}$, and then define H as $H(x) = \{h_1(x), \ldots, h_k(x)\}$.

Little knowledge about SRH is known. Aumasson and Endignoux [4] propose an upper bound of generic security level for SRH, where the hash functions are treated as random oracles. 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 SRH and other assumptions (such as CRH)? Can we construct a provable SRH by other fundamental primitives, such as one-way permutations?

1.1 Our results

In this paper, we attempt to answer the above questions on SRH.

- *A generic quantum attack against SRH* First, we give a quantum algorithm finding subset covers, giving an upper bound of the quantum security of subset resilience. The algorithm is more efficient than simply implementing Grover's search algorithm [20] 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 requires that it is hard to find a

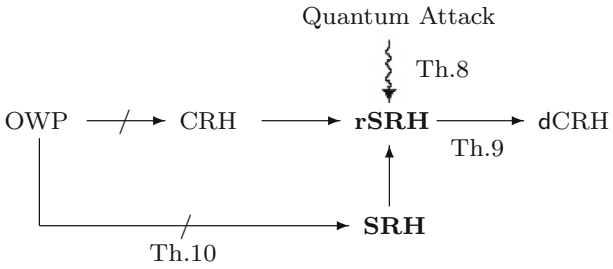

Fig. 1 The relation among subset resilience and other assumptions. A→B means that the existence of A implies the existence of B. A↛B means the impossibility of constructing B from A in the fully black-box manner

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 (OWP) in a fully black-box manner. Our proof uses Simon's separating oracle [38], which proves the fully black-box separation of dCRH from OWP. Although we show that the existence of SRH implies the existence of dCRH in the second result, the third result is still not trivial. It is because the second result does not yield a black-box construction.

To sum up, the relation of SRH and other assumptions about hash functions is depicted in Fig. 1 (where rSRH will be introduced in the following).

1.2 Our techniques

1.2.1 Observations on subset cover finding problem

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

Problem 1 iven $(h_1, \ldots, h_k)$ 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 $H(x) = \{h_1(x), \ldots, h_k(x)\}$.

Problem 2 iven $(h_1, \ldots, h_k)$ and $h_i : X \to Y$ for each i, find $(x, x_1, ...x_r)$ such that $h_i(x) \in \{h_i(x_1), \ldots, h_i(x_r)\}$ 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 as an (r, k)-*restricted subset cover* with regard to $(h_1, \ldots, h_k)$. 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 $(h_1, \ldots, h_k)$ is not harder than finding an (r, k')-subset cover of $(h_1, \ldots, h_{k'})$.*

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 $(x, x_1, \ldots, x_r)$ be an (r, k)-restricted subset cover of $(h_1, \ldots, h_k)$. It implies that for each $i \in [k]$, there exists $x_{a_i} \in \{x_1, \ldots, x_r\}$ such that $h_i(x) = h_i(x_{a_i})$. Thus, $(x, x_{a_1}, \ldots, x_{a_k})$ 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 $(h'_1, \ldots, h'_r)$, it is also an (r, k)-restricted subset cover with regard to $(h_1, \ldots, h_k)$.*

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 $(x, x_{a_1}, \ldots, x_{a_i})$ such that $h_i(x) = h_i(x_i)$ for each i. It is equivalent to the following problem:

Problem 3 iven $(h_1, \ldots, h_k)$ 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 $(h_1, \ldots, h_k)$. We simply call it as 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.

Next, we focus on Problem 3 and k-rSRH and give our results in the three directions. The results can be simply extended to general (r, k)-rSRH and (r, k)-SRH.

1.2.2 Technical overview

Quantum algorithms To find a k-restricted subset cover for multiple functions, we modify Liu's algorithm [32] for finding multi-collision to our multiple function settings (and also modify Hosoyamada's algorithm [23] in Appendix B) . We give an overview of the first algorithm in particular, and the second algorithm is modified in a similar method.

Liu's algorithm, Hosoyamada's algorithm, and ours rely on Grover's algorithm. Given a function $F : X \to \{0, 1\}$ where $|F^{-1}(1)| = t$, Grover's algorithm can find $x \in X$ such that $F(x) = 1$ with probability $O(tq^2/|X|)$ after q number of quantum evaluations of F. Thus, finding a solution requires approximately $O((|X|/t)^{1/2})$ number of quantum computations of F.

We first introduce Liu's algorithm finding multi-collisions w.r.t. k-to-1 functions. Given a k-to-1 function $H : X \to Y$ where $|X| = k|Y| = kN$, it is required to output distinct

$x_1, \ldots, x_k$ such that $H(x_1) = ... = H(x_k)$. In the beginning, it picks t_0 number of random elements from X, and list them in X_1. Let Y_1 be the list of images of X_1. Next, define $F(x)$ such that $F_1(x) = 1$ if and only if $H(x) \in Y_1$ and $x \notin X_1$, and run Grover's algorithm on F_1. Grover's algorithm outputs a solution after $O(\sqrt{|X|/t_0}) = O(\sqrt{N/t_0})$ number of quantum queries to H, which implies a collision of H.

Then, the algorithm repeats Grover's algorithm t_2 times, obtaining t_2 number of collisions. After that, it lists t_2 collisions in list X_2, and lets Y_2 be the list of images of X_2. Similarly, define $F_2(x)$ such that $F_2(x) = 1$ if and only if $F(x) \in Y_2$ and $x \notin X_2$. By running Grover's algorithm, it obtains a 3-collision of H with $O(\sqrt{(N/t_2)})$ quantum queries to H. Again, it repeats Grover's algorithm t_3 times, and then lists t_3 3-collisions in X_3. Similarly, in loop s, it lists t_s number of s-collision of H in X_s. Let $t_k = 1$. After iterations, it eventually obtains a k-collision of H. With well-defined $t_1, \ldots, t_k$, the total number of quantum computations of F in this algorithm is $O(N^{\frac{1}{2}(1-\frac{1}{2^k-1})})$.

This algorithm can be modified to find k-restricted subset covers. Suppose the target functions $h_1, \ldots, h_k$ are all 2-to-1 functions. Given $(h_1, \ldots, h_k)$, we first picks t_0 random elements of X, and list them in X_0. Let Y_1 be the image of X_0 with regard to h_1. Define F_1 such that $F_1(x) = 1$ if and only if $h_1(x) \in Y_1$ and $x \notin X_1$. Run Grover's algorithm on F_1 t_1 times. Until now, our algorithm is the same as the original one.

Then, we list the solution of Grover's algorithm on F_1 in X_1'. Note that each element in X_1' implies a collision with an element of X_0 with regard to h_1. List these elements of X_0 in X_1.

Next, we inject h_2 to our algorithm. Let Y_2 be the list of images with regard to h_2. Denote F_2 such that $F_2(x) = 1$ if and only if $h_2(x) \in Y_2$ and $x \notin X_1$, and then run Gorver's algorithm on X_2 t_3 times. Similarly, we list the solutions of Grover's algorithm in X_2' and list the corresponding elements of X_1 in X_2.

Observe that X_2 and X_2' contain collisions w.r.t. h_2. In addition, since $X_2 \subset X_1$, and the collisions of X_2 are all recorded in X_1'. As a result, we obtain a list of t_3 number of 2-restricted subset covers w.r.t. (h_1, h_2).

Similarly, we can inject $h_3, \ldots, h_k$ into our algorithm. Eventually, we obtain a k-restricted subset cover with regard to $(h_1, \ldots, h_k)$. By well-defined $t_0, \ldots, t_k$, the number of quantum queries to h_i is optimized to approximate $O(N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})})$, which is the same as that of Liu's algorithm for finding $(k+1)$-collisions.

To eliminate the requirement of 2-to-1 functions, we require $h_1, \ldots, h_k$ to be general functions with high compression. Formally, we require that the size of the domain is more than $(k+1)$ times larger than the range (which is not a strong condition for hash functions). This condition guarantees that an element of the domain has a second preimage w.r.t. each h_i with high probability. In this case, Grover's algorithms go through with high probability.

Relate to dCRH Our proof is inspired by [28], which proves the statement that the existence of multi-collision-resistant hashing (MCRH) implies the existence of (infinitely often) secure dCRH. In this paper, we turn to prove a similar statement on k-rSRH (compressing $2n$ bits to n bits) instead of MCRH.

We start with the case that $k = 2$. To prove that the existence of 2-rSRH implies the existence of dCRH, we assume towards the contradiction that there does not exist a dCRH in the world. It implies that there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ that samples a uniform collision of $h \leftarrow \mathcal{H}$ with overwhelming probability over the choice of h. Here, a uniform collision (x_1, x_2) of h implies that x_1 is randomly distributed from the domain

and x_2 is uniformly distributed from the set $h^{-1}(h(x_1))$. Next, to show the contradiction, we need to construct a polynomial-time algorithm breaking 2-rSRH using $\mathcal{A}$.

Now we can obtain collisions with regard to h_1 and h_2 using $\mathcal{A}$, but how to obtain two collision pairs where ones of them are identical? A naive idea is that running $\mathcal{A}(h_1)$ and $\mathcal{A}(h_2)$ respectively. However, since $\mathcal{A}$ outputs uniform collisions, a 2-restricted subset cover only appears with negligible probability.

Recall that $\mathcal{A}$ is probablistic. We write $\mathcal{A}(h; r)$ where r is the randonmess. If we run $\mathcal{A}(h_1)$ twice using randomness r_1 and r_2, we can obtain (x_1, x_2) and (x_3, x_4), which are uniform collisions of h_1 with overwhelming probability. Can we choose particular r_1 and r_2 such that we can build a "bridge" between x_1 and x_3 with regard to h_2?

We fix the input function of $\mathcal{A}$. Now $\mathcal{A}(h_1; \cdot)$ can be considered a deterministic function of r. Since we focus on x_1 and x_3 rather than x_2 and x_4, we denote $f(\cdot)$ by the first element of the $\mathcal{A}(h_1; \cdot)$. If we want (x_1, x_3) to be a collision of h_2, the problem becomes finding r_1 and r_2 such that $h_2(f(r_1)) = h_2(f(r_2))$. Surprisingly, it becomes another collision-finding problem w.r.t. $h_2(f(\cdot))$.

Therefore, our algorithm breaking 2-rSRH runs as follows. Given (h_1, h_2) and a dCRH-breaker $\mathcal{A}$, define $f(r)$ as the first element of $\mathcal{A}(h_1; r)$ and $h'(r) \triangleq h_2(f(r))$. Next, run $(r_1, r_2) \leftarrow \mathcal{A}(h')$. After that, run $(x_1, x_2) \leftarrow \mathcal{A}(h_1; r_1)$ and $(x_3, x_4) \leftarrow \mathcal{A}(h_1; r_2)$. We observe that $h_1(x_1) = h_1(x_2)$ and $h_2(x_1) = h_2(x_3)$ hold.

Next, we need to show that (x_1, x_2) and (x_1, x_3) are both distinct. The proof of the first statement is not complicated. Since r_1 is the first element of output of $\mathcal{A}(h')$, r_1 is uniformly distributed from the domain. Thus, $(x_1, x_2) \leftarrow \mathcal{A}(h_1; r_1)$ flips a random coin, and (x_1, x_2) is distributed as a uniform collision of h_1. The main problem is the second statement. We prove this statement by in two steps. Let $f(r)$ be the first element of $\mathcal{A}(h_1; r)$, which implies that $x_1 = f(r_1)$ and $x_3 = f(r_2)$. First, we observe that f is regular due to the definition of $\mathcal{A}$. Thus, for every x, the corresponding sets $f^{-1}(x)$ are of the same size and distinct. Since (r_1, r_2) is a uniform collision of h', we prove that r_1 and r_2 drop into the same set $f^{-1}(x)$ for some x only with negligible probability. Therefore, $f(r_1) = f(r_2)$ (which implies $x_1 = x_3$) only holds with negligible probability.

From the above statements, we can construct a machine breaking 2-rSRH with $\mathcal{A}$. Then we extend the result to k-rSRH for general constant k by inductions. That is, we constrct a machine breaking $(k+1)$-rSRH with $\mathcal{A}$ together with a machine breaking k-rSRH, say Break-k-SR. By running Break-k-SR twice with different random coins, we can obtain two k-restricted subset covers of $h_1, \ldots, h_k$. Then, we need to build a bridge between the first elements of two subset covers. Similarly, let $f(r)$ be the first elements of Break-k-SR$(h_1, \ldots, h_k; r)$ and $h'(r) \triangleq h_{k+1}(f(r))$. By running $\mathcal{A}(h')$, we can obtain the two random coins that we want. Essentially, this implies general relation between dCRH and k-rSRH for constant k.

Separating from OWP Suppose there exists a fully black-box construction of k-rSRH from OWP f, say $C^f = (C_1^f, \ldots, C_k^f)$. It implies that there exists a reduction $\mathcal{R}$ such that: if there exists an adversary $\mathcal{A}$ that can break k-restricted subset resilience of C^f, then there exists a polynomial-time reduction $\mathcal{R}$ inverting f with access to f and $\mathcal{A}$ with non-negligible probability. Our result rules out the existence of such a polynomial-time $\mathcal{R}$ for any polynomial-time C^f.

Our proof uses Simon's separating oracle [38] consisting of two oracles $\Gamma = (f, \mathsf{CoverFinder})$. f is a random permutation, and $\mathsf{CoverFinder}^f(C_1^f, \ldots, C_k^f)$ is an oracle that can output a k-restricted subset cover of $(C_1^f, \ldots, C_k^f)$ with high probability. (Note that CoverFinder may be exponential-time.) Thus, CoverFinder behaves as the adversary $\mathcal{A}$

breaking k-rSRH. Now we prove that $\mathcal{R}^{\Gamma}$ cannot invert f. More formally, we prove that for random y, $\mathcal{R}^{\Gamma}(y)$ cannot output x such that $f(x) = y$ with non-negligible probability.

If $\mathcal{R}$ is only given the access to f rather than $(f, \mathsf{CoverFinder})$, this statement is true since a random permutation f_n is one-way with overwhelming probability [19]. However, CoverFinder may be helpful for inversing f. In our proof, we define a special event called y**-hit**. When $\mathcal{R}$ queries CoverFinder, if it obtains some the preimage of y with regard to f from the output, we say that y**-hit** occurs. Intuitively, $\mathcal{R}$ can gain some advantage of inversing y from CoverFinder only if it triggers y**-hit** in the queries to CoverFinder . Nevertheless, since CoverFinder outputs a random subset cover for each query, y**-hit** only occurs with negligible probability in each query. This implies that $\mathcal{R}$ can hardly gain advantage from the queries to CoverFinder, and thus it cannot invert f.

Our proof consists of four steps. First, we construct the separating oracle Γ and show that it can break any k-rSRH. Second, we prove that it is hard to compute $f^{-1}(y)$ without triggering y**-hit** even given access to Γ. Third, we prove that if there exists a polynomial-time adversary computing $f^{-1}(y)$ with triggering y**-hit**, then there exists another polynomial-time adversary computing $f^{-1}(y)$ without triggering y**-hit** (which contradicts to the last statement). The last two statements imply the one-wayness of f given access to Γ. Finally, we use the above statements to prove the separation result.

1.3 Motivations

Different hash functions have different security levels. For example, a hash function may be more resistant to collisions than another one. However, none of them can avoid *generic* attacks, which work regardless of the structures of hash functions. For instance, a birthday attack is a generic attack against collision resistance in classical settings. Grover's algorithm [20] and BHT algorithm [12] are generic quantum attacks on one-wayness and collision resistance, respectively.

Generic attacks are essential for the security analysis of hash functions. It provides an upper bound for the security level. In many cases, if the hash function is considered a random oracle, the complexity of an optimal generic attack usually represents the exact security level of this security notion. Since subset resilience is a security notion used in hash-based signature schemes, which are required to be post-quantum, it is essential to analyze the complexity of generic attacks on this security notion. It motivates our first result, giving generic quantum attacks on subset resilience.

Usually, we prefer cryptographic schemes based on *weaker* assumptions. For instance, we prefer a scheme based on the hardness of CDH (Computational Diffie-Hellman Problem) rather than that based on the hardness of DDH (Decisional Diffie-Hellman Problem). Since finding a solution to CDH is harder than DDH, it is more convincing that CDH is hard. In hash-based cryptography, we also have this consideration. If **P=NP**, there is no one-way or collision-resistant hash function at all. However, if we believe that one-way functions exist (this world is called *Minicrypt* in [26]), does it imply that CRH also exists? Actually, it is not the truth. Simon [38] first proves the impossibility of constructing a CRH from one-way permutation in fully black-box manners. Komargodski et al. [29] define four worlds of hashing-related primitives. Roughly speaking, in one of the worlds (called *Hashomania*), CRH exists, while in other worlds (called *Unihash* and *Minihash*), one-way functions exist, but no CRH exists. We do not know which our real world is, but we prefer the schemes that remain secure in weaker worlds.

It motivates our second result analyzing the "power" of SRH. When we assume that SRH exists, we need to know the reliability of this statement. In our work, we prove that the existence of SRH implies the existence of dCRH. It implies that the assumption that SRH exists lives in the world where dCRH exists.

In addition, we compare the power of subset resilience with one-wayness in our third result. We rule out the possibility of constructing an SRH from one-way permutations in fully black-box manners. Although it is not a positive result linking SRH to one-way permutations, it proves that SRH does not obviously exist in $Minicrypt$ in a fully black-box manner. Consider the HORS signature scheme, a hash-based signature scheme whose CMA security is based on one-wayness and SRH. Our result implies that CMA security of HORS cannot be reduced to one-wayness solely in fully black-box manners.

We remark that the security of subset resilience does not directly affect the security of SPHINCS [8] and its variants [5,9], which are practical and many-time stateless HBS. In SPHINCS, when signing a message m, it generates a pseudorandomness r, and then signs the hash value of $r||m$. This enables the schemes to be based on a variant assumption called *target* subset resilience rather than subset resilience. (It is similar in other variants of SPHINCS.) However, if the computation of r is corrupted (e.g. in the case that r is computed by calculating a pseudorandom function on the message m, and the key of the pseudorandom function is leaked), the security will be degraded to be based on (restricted) subset resilience. Note that the leakage of the pseudorandom function key only affects the security while the key pair is still in use. A leakage after its life will not cause a forgery immediately.

1.4 Related work

Subset resilience and hash-based signatures Subset resilience is first proposed in [36] as one of the assumptions needed in a few-time hash-based signature scheme called HORS. The security under chosen message attack is based on one-wayness and subset resilience. Then, Pieprzyk et al. [35] propose HORS++, a variant of HORS, and remove the requirement of subset resilience. Instead, they introduce a cover-free family [17], which captures similar property to subset resilience. In other words, a cover-free family imply an information-theoretic version of SRH, meaning that the probability of finding a subset cover is exactly 0.

In 2015, Bernstein et al. [8] propose SPHINCS, a practical stateless hash-based signature scheme. This scheme implements HORST, a simple variant of HORS, as a primitive in the structure. 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.

After that, Aumasson and Endignoux [4] first analyze the generic security of subset resilience in classical settings and then propose an attack on HORS called 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 called Gravity-SPHINCS [5].

The state-of-art version of SPHINCS is SPHINCS+ [9]. In SPHINCS+, the few-time signature scheme is instantiated by FORS, another variant of HORST. SPHINCS+ is now considered an alternative candidate of NIST post-quantum cryptography standardization. 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. [40] analyze the security of FORS in SPHINCS+ and proposes a variant of FORS called DFORS. The motivation of DFORS also comes from a consideration about the non-target cases.

As far as we know, all of the existing practical stateless (few-time or many-time) hash-based signature schemes are related to subset resilience or its variants. On the other hand, all of the existing stateful hash-based signature schemes (including one-time signature schemes) are not related to subset resilience, such as MSS [34], XMSS [13], XMSS^{MT} [24], XMSS-T [25], and LMS [31,33].

Collision Resistance and its variants Collision-resistant hashing (CRH) is first proposed in [15]. A "birthday attack" can be implemented to find collisions with $O(N^{1/2})$ number of queries to the function where N denotes the size of the range. In a quantum setting, Brassard et al. [12] 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 [1,42]. In addition, Chailloux et al. [14] 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) is first proposed in [16]. Komargodski and Yogev [28] first analyze the power of dCRH and proves that a statistical zero-knowledge proof implies a dCRH. Then, Bitansky et al. [10] prove that a dCRH implies a constant-round statistically-hiding commitment scheme, and a two-message statistically-hiding commitment scheme also implies a dCRH.

Remark 1 We can follow the idea of dCRH to define a relaxation of k-rSRH that we call distributional k-rSRH. That is, given $h_1, \ldots, h_k$, it is hard to sample $(x, x_1, \ldots, x_k)$ such that x is uniformly distributed from the domain and x_i is uniformly chosen from the set $S_i(x) \triangleq \{x' : h_i(x') = h_i(x)\}$. Indeed, our second and third result on k-rSRH also works on distributional k-rSRH.

Multi-collision-resistant hashing (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. Interestingly, MCRH has very similar properties to k-rSRH. Suzuki et al. [39] first analyze the time complexity of MCRH. Hosoyamada et al. [23] and Liu and Zhandry [32] propose two quantum algorithms on finding k-collisions, which can be modified to our quantum algorithms on finding $(k-1)$-restricted subset covers. Komargodski and Yogev [28] prove that the existence of MCRH implies the existence of dCRH. In recent years, there are other results on MCRH [11,29].

Remark 2 Although MCRH and k-rSRH have very similar properties, we do not know any precise relation between them (neither implication nor separation). In our perspective, they are likely to be assumptions in "orthogonal" directions. A multi-collision implies a "series" of collisions of a single function, while a restricted subset cover implies "paralleled" collisions of several functions. More generally, we can mix up these ideas and define a rather hard problem as follows: given $h_1, \ldots, h_k$ with the same domain X, find $S = (x_{i,j}) \in X^{m \times k}$ such

that for each $i \in [k]$, it holds that $h_i(x_{i,1}) = \ldots = h_i(x_{i,m})$ and that $x_{i,1}, \ldots, x_{i,m}$ are distinct. We can also define an assumption that finding such a solution is hard for some hash function families. However, we cannot observe any applications or motivations of this assumption except it is a more general form of collision resistance that covers both multi-collisions and multiple functions.

Black-box Separation The first black-box separating result is proposed in [27], which demonstrates the impossibility of constructing a key-agreement protocol from one-way permutations in a black-box manner. Simon [38] proves the fully black-box separation of CRH from one-way permutations. Indeed, Simon's proof also rules out the possibility to construct a dCRH from one-way permutations.

In 2005, Haitner et al. [21] prove the fully black-box separation of constant-round statistically-hiding commitment schemes from one-way permutations. Furthermore, Berman et al. [7] and Komargodski et al. [29] 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.

1.5 Organizations

This paper is organized as follows: In Sect. 2, we introduce the preliminaries and give the definition of restricted subset-resilient hash function families (rSRH). In Sect. 3, we give a quantum algorithm breaking any k-rSRH. In Sect. 4, we prove the statement that if k-rSRH exists, then distributional collision-resistant hash function families exist. In Sect. 5, we prove the statement that it is infeasible to construct a provable k-rSRH by one-way permutations. In Sect. 6, we extend the three results to general (r, k)-SRH. In Sect. 7, we give the conclusions and several open questions.

2 Preliminaries

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 $[n] = \{1, \ldots, n\}$. We say that $\epsilon : \mathbb{N} \leftarrow \mathbb{R}^+$ 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.

2.1 Efficient function families

Definition 1 (*Efficient function family ensemble*) A function family ensemble $\mathcal{F} = \{\mathcal{F}_n : \mathcal{D}_n \rightarrow \mathcal{R}_n\}_{n \in \mathbb{N}}$ 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 $(f_1, \ldots, f_k) \in \mathcal{F}_n^k$.

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.

2.2 Security notions for hash functions

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

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

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* [15]) 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

$$\Pr_{h \leftarrow \mathcal{H}_n} \left[\begin{array}{c} x_1 \neq x_2 \\ h(x_1) = h(x_2) \end{array} \middle| (x_1, x_2) \leftarrow \mathcal{A}(1^n, h) \right] \leq \epsilon(n) \tag{2}$$

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 $\mathsf{COL}_h \subseteq \{0,1\}^m \times \{0,1\}^m$ 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* [16]) 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

$$\Pr_{h \leftarrow \mathcal{H}_n}[\Delta(\mathcal{A}(1^n, h), \mathsf{COL}_h) \leq \delta(n)] \leq 1 - \epsilon(n) \tag{3}$$

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 Grover's algorithm and its applications

In this paper, we assume the readers have sufficient knowledge of basic quantum computations and omit formal descriptions.

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 [20].

Theorem 1 [20] *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 [12,42]. 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 $L_1 = \{x_1, \ldots, x_t\}$, where x_i is chosen uniformly from the set X.
2. Evaluate $y_i = H(x_i)$ for $i = 1, \ldots, t$. List them in $L_2 = \{y_1, \ldots, y_t\}$. 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 [2,6,14,42].

In addition, Hosoyamada et al. [23] and Liu and Zhandry [32] 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 $O(N^{3n/7})$.

2.4 Subset-resilient hash funtion families

Subset resilience is first proposed in [36], which is an assumption needed for EU-CMA (Existential Unforgeability under Chosen Message Attacks) security of HORS. 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 $(x, x_1, \ldots, x_r)$ 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 6 Let $H = (h_1, \ldots, h_k)$ be a tuple of functions where $h_i : X \to Y$ for each $i \in [k]$. Let $\mathsf{ORS}_k(x) = \{h_1(x), \ldots, h_k(x)\}$. We say $(x, x_1, \ldots, x_r) \in X^{r+1}$ is an (r, k)-subset cover of H if $\mathsf{ORS}_k(x) \subseteq \bigcup_{i \in [r]} \mathsf{ORS}_k(x_i)$.

Definition 7 (*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

$$\Pr\left[\begin{array}{c} x \notin \{x_1, \ldots, x_r\} \\ \mathsf{ORS}_k(x) \subseteq \bigcup_{j \in [r]} \mathsf{ORS}_k(x_j) \end{array} \middle| \begin{array}{c} (h_1, \ldots, h_k) \leftarrow \mathsf{Samp}_k(1^n) \\ (x, x_1, \ldots, x_r) \leftarrow \mathcal{A}(1^n, h_1, \ldots, h_k) \end{array}\right] \leq \epsilon(n) \quad (4)$$

holds for large enough n.

Next, we define a weaker assumption than subset resilience, which we call as *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 $(x, x_1, \ldots, x_r)$, $h_i(x)$ 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 8 Let $H = (h_1, \ldots, h_k)$ be a tuple of functions where $h_i : X \to Y$ for each $i \in [k]$. We say that $(x, x_1, \ldots, x_r) \in X^{r+1}$ 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 9 (*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

$$\Pr\left[\begin{array}{c} \forall i \in [k], x \neq x_i \\ h_i(x) \in \bigcup\limits_{j\in[r]}\{h_i(x_j)\} \end{array}\middle| \begin{array}{c} (h_1, \ldots, h_k) \leftarrow \mathsf{Samp}_k(1^n) \\ (x, x_1, \ldots, x_r) \leftarrow \mathcal{A}(1^n, h_1, \ldots, h_k) \end{array}\right] \leq \epsilon(n), \tag{5}$$

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 $H = (h_1, \ldots, h_k)$ 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 10 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

$$\Pr\left[\begin{array}{c} \forall i \in [k], x \neq x_i \\ h_i(x) = h_i(x_i) \end{array}\middle| \begin{array}{c} (h_1, \ldots, h_k) \leftarrow \mathsf{Samp}_k(1^n) \\ (x, x_1, \ldots, x_k) \leftarrow \mathcal{A}(1^n, h_1, \ldots, h_k) \end{array}\right] \leq \epsilon(n), \tag{6}$$

holds for large enough n.

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

2.5 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 [36] is based on SRH and one-way function families.

Lemma 1 (HORS: Hash to Obtain Random Subset [36]) *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 attack in [4].

Lemma 2 *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.*

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.

We give the formal description and security proof of scheme in Lemma 2 in Appendix A.

Remark 3 Essentially, the scheme in Lemma 2 is a very simplified version of FORS [9]. 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 [9].

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 $H' = (h'_1, \ldots, h'_k)$.

Apart from hash-based signature schemes, since SRH has similar properties to cover-free families (CFFs), it can also be used in other CFF-based primitives. For example, in [41] and [22], 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 hash-based signature schemes.

3 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 [32], 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.

3.1 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 $X_1 = \{x_1^{(1)}, \ldots, x_1^{(t_1)}\}$, 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 $Y_1 = \{y_1^{(1)}, \ldots, y_1^{(t_1)}\}$. This step requires $N^{3/7}$ queries to H.
3. Define $F_1 : Dom \to \{0, 1\}$:

$$F_1(x) = \begin{cases} 1, & x \notin X_1 \text{ and } h_1(x) \in Y_1, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

   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 $X_2 = \{x_2^{(1)}, \ldots, x_2^{(t_2)}\}$.
5. Evaluate $y_2^{(i)} = h_2(x_2^{(i)})$ for $i \in [t_2]$. List them in $Y_2 = \{y_2^{(1)}, \ldots, y_2^{(t_2)}\}$. This step requires $N^{1/7}$ queries to H.
6. Define $F_2 : Dom \to \{0, 1\}$:

$$F_2(x) = \begin{cases} 1, & x \notin X_2 \text{ and } h_2(x) \in Y_2, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

   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 $X_1 = \{x_1^{(1)}, \ldots, x_1^{(t_1)}\}$, 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 $Y_1 = \{y_1^{(1)}, \ldots, y_1^{(t_1)}\}$. This step requires $N^{7/15}$ queries to H.
3. Define $F_1 : Dom \to \{0, 1\}$:
$$F_1(x) = \begin{cases} 1, & x \notin X_1 \text{ and } h_1(x) \in Y_1, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$
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 $F_2 : Dom \to \{0, 1\}$:
$$F_2(x) = \begin{cases} 1, & x \notin X_1 \text{ and } h_2(x) \in Y_2, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$
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 $F_3 : Dom \to \{0, 1\}$:
$$F_2(x) = \begin{cases} 1, & x \notin X_1 \text{ and } h_3(x) \in Y_3, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

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 $s \in \{1, \ldots, k+1\}$. Let $H = (h_1, \ldots, h_k)$ 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 $X_0 = \{x^{(1)}, \ldots, x^{(t_1)}\}$, 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 as $F_s : Dom \to \{0, 1\}$ 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 $(x^{(i)}, x_1'^{(i)}, \ldots, x_k'^{(i)})$.

Remark 4 In practice, $(h_1, \ldots, h_k)$ 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. $h_1, \ldots, h_k$ can be computed by a single hash query. Thus, if the memory is large enough, we can compute all the $Y_1, \ldots, Y_k$ 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

$$\sum_{s=1}^{k} t_{s+1}\sqrt{\frac{N}{t_s}} = \sum_{s=1}^{k} N^{\frac{2^{k-s}-1}{2^{k+1}-1}+\frac{1}{2}(1-\frac{2^{k-s+1}-1}{2^{k+1}-1})} = \sum_{s=1}^{k} N^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})} = kN^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})}. \quad (12)$$

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

$$\sum_{s=1}^{k} t_s + t_{s+1} < \sum_{s=1}^{k} 2t_1 = 2kN^{\frac{1}{2}(1-\frac{1}{2^{k+1}-1})}. \quad (13)$$

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

Theorem 2 *For constant $k \geq 2$, let $H = (h_1, \ldots, h_k)$ 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 number of "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 3 *Let $H = (h_1, \ldots, h_k)$ 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

$$\Pr_x[x \notin \bigcup_{i\in[k]} Dom_{h_i}] = \frac{|Dom| - |\bigcup_{i\in[k]} Dom_{h_i}|}{|Dom|} \geq \frac{(k+1)N - kN}{kN} = \frac{1}{k}, \tag{14}$$

This completes the proof. □

Then, we require $H = (h_1, \ldots, h_k)$ 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 $S_H = Dom \setminus \bigcup_{i\in[k]} Dom_{h_i}$). 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.

$$\Pr[|F_1^{-1}(1)| < t_1] < \Pr[|X_1 \cup S_H| < t_1] < e^{-\frac{c^2 t_1}{2}}, \tag{15}$$

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 algoirithm 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 3 *For constant $k \geq 2$, let $H = (h_1, \ldots, h_k)$ 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 5 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 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 $s \in \{2, \ldots, t+1\}$ 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 $s \in \{2, \ldots, k+1\}$ 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

$$\sum_{s=1}^{k} t_{s+1}\sqrt{\frac{N}{t_s}} = \sum_{s=1}^{k} N^{\frac{1}{2}(1-\frac{\log_N t_1}{2^k-1})} = kN^{\frac{1}{2}(1-\frac{\log_N t_1}{2^k-1})}. \tag{16}$$

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.

4 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 [28], 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.

4.1 From 2-rSRH to dCRH

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

Theorem 4 *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 $\mathsf{BreakSR}$ 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

$$\Pr_{h_1 \leftarrow \mathcal{D}_1}[\Delta(\mathcal{A}(1^n, h_1), \mathsf{COL}_{h_1}) \leq \delta(n)] > 1 - \epsilon(n), \tag{17}$$

and thus

$$\Pr_{(h_1,h_2) \leftarrow \mathsf{Samp}}[\Delta(\mathcal{A}(1^n, h_1), \mathsf{COL}_{h_1}) \leq \delta(n)] > 1 - \epsilon(n). \tag{18}$$

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:

$$h_2'(r) \triangleq h_2(\mathcal{A}^1(h_1; r)). \tag{19}$$

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

$$\Pr_{(h_1,h_2) \leftarrow \mathsf{Samp}}[\Delta(\mathcal{A}'(1^n, h_2'), \mathsf{COL}_{h_2'}) \leq \delta'(n)] > 1 - \epsilon'(n) \tag{20}$$

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

$$\Pr\left[\begin{array}{c} x_1 \neq x_2, x_1 \neq x_3 \\ h_1(x_1) = h_1(x_2), h_2(x_1) = h_2(x_3) \end{array} \middle| \begin{array}{c} (h_1, h_2) \leftarrow \mathsf{Samp}(1^n) \\ (x_1, x_2, x_3) \leftarrow \mathsf{BreakSR}(1^n, h_1, h_2) \end{array}\right] > 1 - \mu(n) \tag{21}$$

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. $\mathsf{BreakSR}$ 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 Eq. (20), 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

$$\Pr[\Delta(\textbf{Game 1}, \textbf{Game 0}) \leq \delta'(n)] > 1 - \epsilon'(n), \tag{22}$$

  and thus

$$|\Pr[\textbf{Game 1}] - \Pr[\textbf{Game 0}]| < \epsilon'(n) + \delta'(n), \tag{23}$$

  where the probability is taken over the choice of (h_1, h_2) and the randomness of $\mathsf{BreakSR}$.
- 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 Eq. (21) also occur with overwhelming probability.

□

Lemma 4

$$\Pr_{(h_1,h_2),\mathsf{COL}_{h_2'}}[h_1(x_1) \neq h_1(x_2) \vee x_1 = x_2] < \epsilon(n) + \delta(n) + 2^{-n/2+1}. \tag{24}$$

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 Eq. (24) 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 Eq. (18)). We have

$$\begin{aligned} &\Pr_{(x_1,x_2)\leftarrow\mathcal{A}(h_1;r_1)}[h_1(x_1) \neq h_1(x_2) \vee x_1 = x_2] \leq \\ &\Pr_{(x_1',x_2')\leftarrow\mathsf{COL}_{h_1}}[h_1(x_1') \neq h_1(x_2') \vee x_1' = x_2'] + \delta(n) \end{aligned} \tag{25}$$

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

$$\Pr_{(x_1',x_2')\leftarrow\mathsf{COL}_{h_1}}[x_1 = x_2] < 2^{-n/2+1}. \tag{26}$$

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

$$\Pr_{x \leftarrow \{0,1\}^{2n}}[X_{h_1(x)} \text{ is large}] \geq \frac{2^{2n} - 2^{3n/2}}{2^{2n}} = 1 - 2^{-n/2}. \tag{27}$$

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 Eqs. (25) and (26), we complete the proof of Lemma 4.

□

Lemma 5

$$\Pr_{(h_1,h_2),\mathsf{COL}_{h_2'}}[x_1 = x_3] < \epsilon(n) + 2\delta(n) + 2^{-n/2+1} \tag{28}$$

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

$$\sum_{x_1 \in \{0,1\}^{2n}} \left| \Pr_{r_1 \in \{0,1\}^{l_r(n)}}[x_1 \leftarrow \mathcal{A}^1(h_1; r_1)] - \frac{1}{2^{2n}} \right| \leq \delta(n). \tag{29}$$

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:

$$R_x \triangleq \{r | \mathcal{A}^1(h_1; r) = x\}. \tag{30}$$

Then, we have

$$\sum_{x_1 \in \{0,1\}^{2n}} \left| \frac{|R_{x_1}|}{2^{l_r(n)}} - \frac{1}{2^{2n}} \right| \leq \delta(n). \tag{31}$$

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:

$$S_{x_1} = \{r | h_2(\mathcal{A}^1(h_1; r)) = h_2(x_1)\}. \tag{32}$$

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

$$S_{x_1} = \bigcup_{x' \in X_{h_2(x_1)}^{h_2}} R_{x'}. \tag{33}$$

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

$$\left|\frac{|R_{x_1}|}{|S_{x_1}|} - \frac{1}{|X_{h_2(x)}|}\right| \leq \delta(n). \tag{34}$$

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,

$$\Pr_{r_1}[X_{h_2(x_1)} \text{ is large}] \geq \Pr_{x_1}[X_{h_2(x_1)} \text{ is large}] - \delta(n) \geq 1 - 2^{-n/2} - \delta(n), \tag{35}$$

where the second inequality is due to Eq. (27).

From Eqs. (34) and (35) 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

$$\Pr_{(h_1,h_2),\mathsf{COL}'_{h_2}}[x_3 = x_1] < \epsilon(n) + 2\delta(n) + 2^{-n/2+1}, \tag{36}$$

which completes the proof of Lemma 5. □

From Lemmas 4 and 5 we have

$$\Pr[\textbf{Game 1}] \geq 1 - \epsilon(n) - 3\delta(n) - 2^{-n/2+2}, \tag{37}$$

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 (21) holds. This completes the proof. □

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

The induction from s to $s+1$ is similar to the construction of 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 Break-$(s+1)$-SR$(1^n, h_1, h_2, \ldots, h_{s+1})$ breaking $(s+1)$-rSRH runs as follows:

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

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

Theorem 5 *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.

□

5 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 [38]. Using this oracle, Asharov and Segev [3] proves the separation result of CRH from OWP and indistinguishability obfuscators. Berman et al. [7] proves the separation result of MCRH from OWP. We follow their work and show a similar result about k-rSRH.

Definition 11 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

$$\Pr_{\mathsf{Samp},\mathcal{A}}\left[\begin{array}{c|c} \forall i, x \neq x_i & (C_1^f, \ldots, C_k^f) \leftarrow \mathsf{Samp}^f(1^n) \\ C_i(x) = C_i(x_i) & (x, x_1, \ldots, x_k) \leftarrow \mathcal{A}^f(C_1, \ldots, C_k) \end{array}\right] \geq \frac{1}{p_{\mathcal{A}}(n)} \tag{38}$$

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

$$\Pr_{y\leftarrow\{0,1\}^n,\mathcal{R}}[y = f_n(x) | x \leftarrow \mathcal{R}^{f,\mathcal{A}}(y)] \geq \frac{1}{p_{\mathcal{R}}(n)} \tag{39}$$

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

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

Theorem 6 (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 Sect. 5.1, 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 $(C_1^f, \ldots, C_k^f)$ 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 [19]. To prove the stronger statement, we define a special event called y-**hit**. In Sect. 5.1, we prove that any polynomial-time algorithm cannot invert y without triggering this event. This uses the Reconstruction Paradigm in [18]. In Sect. 5.3, 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 Sect. 5.4, 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.

5.1 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 $\mathsf{CoverFinder}^f(C_1, \ldots, C_k)$: Let $C_1^f, \ldots, C_k^f : \{0,1\}^n \to \{0,1\}^{l(n)}$ be k circuits given access to oracle f. Given the description of $C_1^f, \ldots, C_k^f$, $\mathsf{CoverFinder}^f(C_1, \ldots, C_k)$ tries to output a k-restricted subset cover for $(C_1^f, \ldots, C_k^f)$ 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 $(w, \pi_1(w_1), \ldots, \pi_k(w_k))$. Note that $\mathsf{CoverFinder}^f$ is expected to be exponential-time.

In the following, we say 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 6 *For any permutation f_n and any oracle-aided circuits $C_1^f, \ldots, C_k^f : \{0,1\}^n \to \{0,1\}^{l(n)}$ where $n > l(n) - \log k$, there exists a polynomial $p(n)$ such that*

$$\Pr_{\mathsf{CoverFinder}}\left[\begin{array}{l}\forall i \in [k], x \neq x_i \\ C_i^f(x) = C_i^f(x_i)\end{array}\middle| (x, x_1, \ldots, x_k) \leftarrow \mathsf{CoverFinder}(1^n, C_1^f, \ldots, C_k^f)\right] \geq \frac{1}{p(n)} \tag{40}$$

for infinitely many n.

Proof We first show that for $(x, x_1, \ldots, x_k) \leftarrow \mathsf{CoverFinder}^f(1^n, C_1^f, \ldots, C_k^f)$, 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

$$\Pr_{\mathsf{CoverFinder}}[x \in \bigcup_{i\in[k]} X_{C_i}] \leq \frac{k \cdot 2^{l(n)}}{2^n} = \frac{1}{2^{n-l(n)-\log k}} < \frac{1}{2}, \tag{41}$$

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

Next, we show that $(x, x_1, \ldots, x_k)$ is a k-restricted subset cover w.r.t. $(C_1, \ldots, C_k)$ 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 $\mathsf{CoverFinder}$, 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

$$\Pr_{\mathsf{CoverFinder}}\left[\begin{array}{l}\forall i \in [k], x \neq x_i \\ C_i(x) = C_i(x_i)\end{array}\middle| x \notin \bigcup_{i\in[k]} X_{C_i}\right] \geq \frac{1}{2^k}. \tag{42}$$

Note that k is constant. Due to inequality (41) and (42), the probability that $(x, x_1, \ldots, x_k)$ is a k-restricted subset cover is more than $\frac{1}{2^{k+1}}$. This completes the proof. □

5.2 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

$$\Pr_{\substack{y\leftarrow\{0,1\}^n \\ f_n, \mathsf{CoverFinder}, \mathcal{A}}}[y = f_n(x) | x \leftarrow \mathcal{A}^\Gamma(1^n, y)] \leq \mathbf{negl}(n) \tag{43}$$

for large enough n.

Let $\mathcal{A}$-**win** be the above event and we need to show that $\Pr[\mathcal{A}\text{-}\mathbf{win}] \leq \mathbf{negl}(n)$. 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 12 In the process of running $\mathcal{A}^\Gamma(y)$, when $\mathcal{A}$ makes a query $(C_1, \ldots, C_k)$ to $\mathsf{CoverFinder}$ and obtains $(x, x_1, \ldots, x_k)$, 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 $\mathsf{CoverFinder}$ triggering y-**hit**, we simply say that y-**hit** occurs.

Lemma 7 *For any polynomial-time adversary $\mathcal{A}$, it holds that*

$$\Pr_{\substack{y\leftarrow\{0,1\}^n, f_n, \mathcal{A} \\ \mathsf{CoverFinder}}}[\mathcal{A}\text{-}\boldsymbol{win} \wedge \overline{y\text{-}\boldsymbol{hit}}] \leq 2^{-n/7} \tag{44}$$

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**.

$$I_f = \{y | \mathcal{A}\text{-}\mathbf{win} \wedge \overline{y\text{-}\mathbf{hit}}\}. \tag{45}$$

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 $(C_1, \ldots, C_k)$ and obtains $(x, x_1, \ldots, x_k)$, 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 8 *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*

$$|I_f| \leq 3kq^2|Y_f|. \tag{46}$$

Proof Suppose a query to CoverFinder$(C_1, \ldots, C_k)$ is replied by $(x, x_1, \ldots, x_k)$. 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

$$q_{\mathcal{A}} \cdot 2kq_C + q_{\mathcal{A}} \leq 3kq^2 \tag{47}$$

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 $(C_1, \ldots, C_k)$ to CoverFinder, do as follows:
     i. Obtain $(w, \pi_1, \ldots, \pi_k)$ 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 $(w, w_1, w_2, \ldots, w_k)$.

(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 9 *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 $(w, w_1, \ldots, w_k)$. 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 $(w, w_1, \ldots, w_k)$, 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. □

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** $\wedge$ $\overline{y\text{-}\mathbf{hit}}$ 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 $\Pr_{f_n,y}[\mathcal{A}\text{-}\mathbf{wins} \wedge \overline{y\text{-}\mathbf{hit}}]$.

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

$$|Y_f| \geq \frac{|I_f|}{3kq^2} \geq \frac{2^{2n/3}}{3kq^2}. \tag{48}$$

  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

$$\Pr_{f_n}[f_n \in S_\epsilon] \leq \frac{\binom{2^n}{|Y_f|}^2 \cdot (2^n - |Y_f|)!}{(2^n)!} = \frac{\binom{2^n}{|Y_f|}}{|Y_f|!} \leq (\frac{e2^n}{|Y_f|})^{|Y_f|}(\frac{e}{|Y_f|})^{|Y_f|} \leq (\frac{e^2 2^n}{|Y_f|^2})^{|Y_f|}. \tag{49}$$

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

$$(\frac{2^n e^2}{|Y_f|^2})^{|Y_f|} \leq (\frac{9e^2k^2q^4 2^n}{2^{4n/3}})^{|Y_f|} = (\frac{9e^2k^2q^4}{2^{n/3}})^{|Y_f|} \leq 2^{-n}. \tag{50}$$

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

$$\Pr_{f_n,y}[\mathcal{A}\text{-}\mathbf{wins} \wedge \overline{y\text{-}\mathbf{hit}} \wedge f_n \in S_\epsilon] \leq \Pr_{f_n}[f_n \in S_\epsilon] \leq 2^{-n} \tag{51}$$

  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

$$\Pr_{f_n,y}[\mathcal{A}\text{-}\mathbf{wins} \wedge \overline{y\text{-}\mathbf{hit}} \wedge f_n \notin S_\epsilon] \leq \Pr_{y}[\mathcal{A}\text{-}\mathbf{wins} \wedge \overline{y\text{-}\mathbf{hit}} | f_n \notin S_\epsilon] \leq 2^{-n/3}. \tag{52}$$

  From inequality (51) and (52), we have

$$\Pr_{f_n,y}[\mathcal{A}\text{-}\mathbf{win} \wedge \overline{y\text{-}\mathbf{hit}}] \leq 2^{-n} + 2^{-n/3} \leq 2^{-n/2}, \tag{53}$$

for large enough n, which completes the proof. □

5.3 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 10 *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*

$$\Pr_{\mathsf{CoverFinder},\mathcal{A}}[\mathcal{A}\text{-}\boldsymbol{win} \wedge y\text{-}\boldsymbol{hit}] \geq \frac{1}{p(n)} \tag{54}$$

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

$$\Pr_{\mathsf{CoverFinder},\mathcal{B}}[\mathcal{B}\text{-}\boldsymbol{win} \wedge \overline{y\text{-}\boldsymbol{hit}}] \geq \frac{1}{2p(n)} \tag{55}$$

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 $(C_1^f, \ldots, C_k^f)$, 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 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 CoverFinder.

We add following operations to $\mathcal{A}$. Before $\mathcal{A}$ queries $(w, w_1, \ldots, w_k) \leftarrow \mathsf{CoverFinder}(C_1, \ldots, C_k)$, 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 $\mathsf{CoverFinder}(C_1, \ldots, C_k)$. 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 $\mathsf{CoverFinder}(C_1, \ldots, C_k)$ triggers y-**hit**, where the probability is taken over the choice of y and the randomness of 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 CoverFinder, $\mathcal{B}$ behaves as follows:

- When it queries x to f, $\mathcal{B}$ also queries x to f.
- When $\mathcal{A}$ queries $(C_1, \ldots, C_k)$ to 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 $(C_1, \ldots, C_k)$ to CoverFinder.

Next, we prove the inequality (55). Let q be the upper bound of the number of queries to CoverFinder made by $\mathcal{A}$, and let $C^{(1)}, \ldots, C^{(q)}$ be the queries to CoverFinder made by $\mathcal{A}(y)$ (each $C^{(i)}$ is a tuple of k circuits $(C_1^{(i)}, \ldots, C_k^{(i)})$). 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 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,

$$\Pr[\mathcal{B}\text{-}\mathbf{win} \wedge \overline{y\text{-}\mathbf{hit}}] = \sum_{i \in [q]} \Pr[\mathbf{Jump}_i \wedge \bigwedge_{j<i} \overline{\mathbf{Fail}_j}]. \tag{56}$$

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,

$$\Pr_{\mathsf{CoverFinder},\mathcal{B}}[\mathbf{Jump}_i \mid \bigwedge_{j<i} \overline{\mathbf{Fail}_j}] = \frac{1}{2} \Pr[\mathbf{Fail}_i \mid \bigwedge_{j<i} \overline{\mathbf{Fail}_j}]. \tag{57}$$

From equality (56) and (57), we have

$$\Pr_{\mathsf{CoverFinder},\mathcal{B}}[\mathcal{B}\text{-}\mathbf{wins} \wedge \overline{y\text{-}\mathbf{hit}}] = \frac{1}{2} \sum_{i \in [q]} \Pr[\mathbf{Fail}_i \wedge \bigwedge_{j<i} \overline{\mathbf{Fail}_j}] = \frac{1}{2} \Pr[\bigvee_{i \in [q]} \mathbf{Fail}_i]. \tag{58}$$

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.

□

5.4 Main result

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

Theorem 7 *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 Sect. 5.1. Due to Lemma 6, for any permutation f_n, there exists a polynomial $p(n)$ such that

$$\Pr_{\mathsf{Samp},\mathsf{CoverFinder}}\left[\begin{array}{c} \forall i, x \neq x_i \\ C_i(x) = C_i(x_i) \end{array} \middle| \begin{array}{l} (C_1^f, \ldots, C_k^f) \leftarrow \mathsf{Samp}^f(1^n) \\ (x, x_1, \ldots, x_k) \leftarrow \mathsf{CoverFinder}^f(1^n, C_1, \ldots, C_k) \end{array}\right] \geq \frac{1}{p(n)}, \tag{59}$$

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

$$\Pr_{y,\mathcal{R}}[y = f_n(x) | x \leftarrow \mathcal{R}^{\Gamma}(y)] \geq \frac{1}{p_{\mathcal{R}}(n)} \tag{60}$$

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

$$\Pr_{\substack{f_n,y,\mathcal{R}\\ \mathsf{CoverFinder}}}[y = f_n(x) | x \leftarrow \mathcal{R}^{\Gamma}(y)] \geq \frac{1}{p_{\mathcal{R}}(n)}. \tag{61}$$

That is,

$$\Pr[\mathcal{R}\textbf{-win} \wedge y\textbf{-hit}] + \Pr[\mathcal{R}\textbf{-win} \wedge \overline{y\textbf{-hit}}] \geq \frac{1}{p_{\mathcal{R}}(n)}. \tag{62}$$

Due to Lemma 7, $\Pr[\mathcal{R}\textbf{-win} \wedge \overline{y\textbf{-hit}}] \leq 2^{-n/7}$ 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

$$\Pr_{f_n,y}[\Pr_{\mathsf{CoverFinder},\mathcal{R}}[\mathcal{R}\textbf{-win} \wedge y\textbf{-hit}] \geq \frac{1}{2p'_{\mathcal{R}}(n)}] \geq \frac{1}{2p'_{\mathcal{R}}(n)}. \tag{63}$$

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 10, for any $(f_n, y) \in T$, there exists a polynomial-time machine $\widetilde{\mathcal{R}}$ such that

$$\Pr_{\mathsf{CoverFinder},\widetilde{\mathcal{R}}}[\widetilde{\mathcal{R}}\textbf{-win} \wedge \overline{y\textbf{-hit}}] \geq \frac{1}{4p'_{\mathcal{R}}(n)} \tag{64}$$

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 7 showing that this probability is negligible. □

6 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 Sect. 3, we propose a quantum algorithm finding k-restricted subset covers that are also (k, k)-subset covers.
- In Sect. 4, 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 Sect. 5, 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 $H=(h_1,\ldots,h_4)$ 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 Sect. 3 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. $(h_1,\ldots,h_4)$. 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 8 (Extended Theorem 3) *For constant $k\geq 2$ and r, denote $\omega=\lceil k/r\rceil$. Let $H=(h_1,\ldots,h_k)$ 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 $h_{k+1},\ldots,h_{k'}$ 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 $(h_1,\ldots,h_k)$.

Next we assume that $k=\omega r$. For each $i\in[r]$, denote $h_i^*(x)\triangleq h^*_{(i-1)\omega+1}(x)||\ldots||h^*_{i\omega}(x)$. Thus, $H^*=(h_1^*,\ldots,h_r^*)$ 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 3 on H^*, and obtain $(x,x_1,\ldots,x_r)$ 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 9 (Extended Theorem 5) *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 $(h_1,\ldots,h_k)\leftarrow\mathsf{Samp}_k$, and let $h_i^*\triangleq h_{(i-1)\omega+1}||\ldots||h_{i\omega}$ for each $i\in[r]$. Then, it outputs $(h_1^*,\ldots,h_r^*)$. We observe that an r-restricted subset cover of $(h_1^*,\ldots,h_r^*)\leftarrow\mathsf{Samp}_k^*$ implies an (r,k)-subset cover of $(h_1,\ldots,h_k)\leftarrow\mathsf{Samp}_k$. 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 5. This completes the proof. □

Theorem 10 (Extended Theorem 7) *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 $l(n) < (n - \log r)/\omega$.*

Proof Again, we consider the case that $k = \omega r$. In the proof of Theorem 9, 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 7, there is no fully black-box contruction of such a k-rSRH from OWP. It implies that there is no fully black-box contruction of (r, k)-rSRH compressing n bits to $l(n)$ bits from OWP. □

Remark 6 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 [17], 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 Theorems 9 and 10 only work on their constraint conditions.

7 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 fully black-box manners.

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 whether we can improve the results presented in this paper.

Target subset resilience is a weaker variant of subset resilience. It is first proposed as a security notion needed in RMA security of HORS [36]. Although the CMA security of SPHINCS [8] and Gravity-SPHINCS [5] is reduced to subset resilience, the reductions are non-tight since finding a subset cover does not immediately cause a forgery. SPHINCS+ [9] 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 to be 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 around SRH.

Acknowledgements We thank anonymous reviewers for simplifying the proof of Theorem 4 and other valuable comments.

A: Signature schemes

A.1: Definition and Security models

Definition 13 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 $\mathsf{Ver}(pk, m, \sigma) = 1$.

The standard security for a signature scheme is existential unforgeability under chosen message attack (EU-CMA). In the definition, we introduce an experiment for adversary $\mathcal{A}$. In this experiment, the adversary is given the public key pk and access to the signing oracle $\mathsf{Sign}(sk, \cdot)$. $\mathcal{A}$ is allowed to query the signing oracle at most q times, and all the queries are recorded in $\{M_i\}_1^q$. Finally, $\mathcal{A}$ is required to output (m^*, σ^*) such that σ^* is a valid signature for m^* and m^* has not been queried.

Experiment $\mathsf{Exp}_{\Gamma}^{\mathsf{EU\text{-}CMA}}(1^n, \mathcal{A})$
$(pk, sk) \leftarrow \mathsf{KeyGen}(1^n)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(pk)$
if $\mathsf{Ver}(pk, m^*, \sigma^*) = 1 \wedge m^* \notin \{M_i\}_1^q$ **return** 1, otherwise **return** 0.

Definition 14 Let Γ be a signature scheme. We say Γ is existentially unforgeable under q-time chosen message attack if for all probabilistic polynomial-time adversary $\mathcal{A}$, it holds that $\Pr[\mathsf{Exp}_{\Gamma}^{\mathsf{EU\text{-}CMA}}(1^n, \mathcal{A})] \leq \mathbf{negl}(n)$, where $\mathbf{negl}$ is a negligible function.

A.2: Construction in Lemma 2

In the following we show the signature scheme in Lemma 2 and prove the EU-CMA security based on restricted subset resilience and one-wayness. 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:

- $\mathsf{KeyGen}(1^n)$: $(h_1, \ldots, h_k) \leftarrow \mathsf{Samp}_k(\mathcal{H})$. 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 $(h_1, \ldots, h_k)$ and all the $y_{i,j}$'s. The secret key sk contains $(h_1, \ldots, h_k)$ 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 $\sigma = (s_{1,t_1}, \ldots, s_{k,t_k})$.
- $\mathsf{Ver}(pk, m, \sigma)$: Parse $\sigma = (x_1, \ldots, x_k)$. 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.

Theorem 11 *Let $\mathcal{H}$ be an (r,k)-rSRH and f is a one-way function. $r \le c2^{m'}$ for some constant $c < 1$. Then, Γ is existentially unforgeable under r-time chosen message attacks.*

Proof 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 $(h_1, \ldots, h_k)$ 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 $M = (m_1, \ldots, m_r)$ be the queries of $\mathcal{A}$. If $\mathcal{A}$ succeeds, it holds that $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$ and $m^* \notin M$. Let $\sigma^* = (x_1^*, \ldots, x_k^*)$. 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 $(m, m_1, \ldots, m_r)$ is an (r,k)-resilient subset cover of $(h_1, \ldots, h_k)$. In this case, $\mathcal{R}$ returns $(m, m_1, \ldots, m_r)$ 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 $\mathsf{Adv}_{\mathcal{H}}^{(r,k)\text{-rSRH}}(n)$ the upper bound of the probability of breaking subset resilience of $\mathcal{H}$. We have

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

which is negligible due to the assumptions. □

B: A recursive quantum algorithm attacking k-rSRH

In this section, we provide a recursive algorithm attacking k-rSRH, which is inspired by the recursive multi-collision finding algorithm in [23].

Just like Sect. 3.1, we first suppose the functions are 2-to-1 functions. It can also be generalized to functions whose size of the domain is $k+1$ times larger than that of the range in a similar way. We omit the general version in this subsection.

Let $H = (h_1, \ldots, h_k)$, where $h_i : Dom \to Rng$ is 2-to-1 function for each i. For $s \in \{1, \ldots, k\}$, let $t_s = N^{1/3^s}$. We denote by $\mathsf{FindingCover}_s(h_1, \ldots, h_s)$ the quantum algorithm finding s-restricted subset covers. When $s = 1$, it is the same as the collision-finding algorithm in Sect. 2.3. When $2 \le s \le k$, $\mathsf{FindingCover}_s(h_1, \ldots, h_s)$ runs as follows:

1. Run $\mathsf{FindingCover}_{s-1}(1^n, h_1, \ldots, h_{s-1})$ t_s times, and obtain t_s number of $(s-1)$-restricted subset covers of $(h_1, \ldots, h_{s-1})$. List the results as $((x^{(1)}, x_1^{(1)}, \ldots, x_{s-1}^{(1)}), \ldots, (x^{(t_s)}, x_1^{(t_s)}, \ldots, x_{(s-1)}^{(t_s)}))$. Let $X = \{x^{(1)}, \ldots, x^{(t_s)}\}$.
2. Evaluate $y_s^{(i)} = h_s(x^{(i)})$ for all $x^{(i)} \in X$, and list them in Y_s. If there exist $i \neq j$ such that $y_s^{(i)} = y_k^{(j)}$, return $(x^{(i)}, x_1^{(i)}, \ldots, x_{s-1}^{(i)}, x^{(j)})$.
3. Define as $F_s : Dom \to \{0, 1\}$ a function such that $F_s = 1$ if and only if $x \notin X \wedge h_s(x) \in Y_s$. Run Grover's algorithm on F_s, and obtains a solution x_s.
4. Find $y_s^{(i)} \in Y_s$ such that $y_s^{(i)} = h_s(x_s)$. Output $(x^{(i)}, x_1^{(i)}, \ldots, x_{s-1}^{(i)}, x_s)$.

Next, we show that the number of queries to H required in $\mathsf{FindingCover}_s$ is at most $O(N^{\frac{1}{2}(1-\frac{1}{3^s})})$ by induction. When $s = 1$, it is obviously true since the a quantum algorithm finding collisions requires $O(N^{1/3})$ queries to the target function. Assume $\mathsf{FindingCover}_{s-1}$ requires $O(N^{\frac{1}{2}(1-\frac{1}{3^{s-1}})})$ queries to H for some $s \ge 2$, we show $\mathsf{FindingCover}_s$ requires at most $O(N^{\frac{1}{2}(1-\frac{1}{3^s})})$ queries to H.

In step 1, the number of queries to H is at most

$$t_s \cdot O(N^{\frac{1}{2}(1-\frac{1}{3^{s-1}})}) = O(N^{\frac{1}{2}(1-\frac{1}{3^{s-1}})+\frac{1}{3^s}}) = O(N^{\frac{1}{2}(1-\frac{1}{3^s})}). \tag{65}$$

In step 2, the number of queries to H is $t_s = N^{\frac{1}{3^s}}$.

Note that $|F_s^{-1}(1)| = |Y_s| = t_s$. The number of quantum queries required in step 3 is at most

$$O(\sqrt{\frac{2N}{t_s}}) = O(N^{\frac{1}{2}(1-\frac{1}{3^s})}). \tag{66}$$

To sum up, the total number of queries to H is at most $O(N^{\frac{1}{2}(1-\frac{1}{3^s})})$. By induction, it holds for any $1 \le s \le k$. Thus, the final algorithm $\mathsf{FindingCover}_k$ requires $O(N^{\frac{1}{2}(1-\frac{1}{3^k-1})})$ number of queries to the target function.

References

1. Aaronson S., Shi Y.: Quantum lower bounds for the collision and the element distinctness problems. J. ACM (JACM) **51**(4), 595–605 (2004).
2. Ambainis A.: Quantum walk algorithm for element distinctness. SIAM J. Comput. **37**(1), 210–239 (2007).

3. Asharov G., Segev G.: Limits on the power of indistinguishability obfuscation and functional encryption. SIAM J. Comput. **45**(6), 2117–2176 (2016).
4. Aumasson J.-P., Endignoux G.: Clarifying the subset-resilience problem. Cryptology ePrint Archive, Report 2017/909 (2017).
5. Aumasson J.-P., Endignoux G.: Improving stateless hash-based signatures. In: Topics in Cryptology: CT-RSA 2018, Vol. 10808 of LNCS. Springer, Berlin, pp. 219–242 (2018).
6. Belovs A.: Learning-graph-based quantum algorithm for k-distinctness. In: 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science, IEEE, pp. 207–216 (2012).
7. Berman I., Degwekar A., Rothblum R.D., Vasudevan P.N.: Multi-collision resistant hash functions and their applications. In: Advances in Cryptology: EUROCRYPT 2018, Vol. 10821 of LNCS. Springer, pp. 133–161 (2018).
8. Bernstein D.J., Hopwood, D., Hülsing, A., Lange T., Niederhagen R., Papachristodoulou L., Schwabe M.S., Wilcox-O'Hearn Z.: SPHINCS: Practical stateless hash-based signatures. In: Advances in Cryptology: EUROCRYPT 2015, Vol. 9056 of LNCS. Springer, Berlin, pp. 368–397 (2015).
9. Bernstein D.J., Hülsing A., Kölbl S., Niederhagen R., Rijneveld J., Schwabe P.: The SPHINCS+ signature framework. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery, pp. 2129–2146 (2019).
10. Bitansky N., Haitner I., Komargodski I., Yogev E.: Distributional collision resistance beyond one-way functions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, pp. 667–695 (2019).
11. Bitansky N., Kalai Y.T., Paneth O.: Multi-collision resistance: a paradigm for keyless hash functions. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 671–684 (2018).
12. Brassard G., Høyer P., Tapp A.: Quantum cryptanalysis of hash and claw-free functions. In: Latin American Symposium on Theoretical Informatics. Springer, pp. 163–169 (1998).
13. Buchmann J., Dahmen E., Hülsing A.: XMSS—a practical forward secure signature scheme based on minimal security assumptions. In: PQCrypto 2011: Post-Quantum Cryptography, Vol. 7071 of LNCS. Springer, pp. 117–129(2011).
14. Chailloux A., Naya-Plasencia M., Schrottenloher A.: An efficient quantum collision search algorithm and implications on symmetric cryptography. In: Advances in Cryptology: ASIACRYPT 2017, Vol. 10625 of LNCS. Springer, Berlin, pp. 211–240 (2017).
15. Damgård I.B.: Collision free hash functions and public key signature schemes. In: Advances in Cryptology: EUROCRYPT' 87, Vol. 304 of LNCS. Springer, Berlin, pp. 203–216 (1988).
16. Dubrov B., Ishai Y.: On the randomness complexity of efficient sampling. In: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, STOC '06. Association for Computing Machinery, pp. 711-720 (2006).
17. Erdös P., Frankl P., Füredi Z.: Families of finite sets in which no set is covered by the union of r others. Isr. J. Math. **51**, 79–89 (1985).
18. Gennaro R, Trevisan L.: Lower bounds on the efficiency of generic cryptographic constructions. In: 41st Annual Symposium on Foundations of Computer Science, FOCS2000. IEEE, pp. 305–313 (2000).
19. Gennaro R., Gertner Y., Katz J., Trevisan L.: Bounds on the efficiency of generic cryptographic constructions. SIAM J. Comput. **35**(1), 217–246 (2005).
20. Grover L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 212–219 (1996).
21. Haitner I., Hoch J.J., Reingold O., Segev G.: Finding collisions in interactive protocols–tight lower bounds on the round and communication complexities of statistically hiding commitments. SIAM J. Comput. **44**(1), 193–242 (2015).
22. Hofheinz D., Jager T., Kiltz E.: Short signatures from weaker assumptions. Adv. Cryptol **7073**, 647–666 (2011).
23. Hosoyamada A., Sasaki Y., Xagawa K.: Quantum multicollision-finding algorithm. Adv. Cryptol. **10625**, 179–210 (2017).
24. Hülsing A., Rausch L., Buchmann J.: Optimal parameters for XMSS^{MT}. In: CD-ARES 2013: Security Engineering and Intelligence Informatics **8128**, 194–208 (2013).
25. Hülsing A., Rijneveld J., Song F.: Mitigating multi-target attacks in hash-based signatures. In: Public-Key Cyptography—PKC 2016.
26. Impagliazzo R.: A personal view of average-case complexity. In: Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference, IEEE, pp. 134–147 (1995).
27. Impagliazzo R, Rudich S.: Limits on the provable consequences of one-way permutations. In: Proceedings of the twenty-first annual ACM symposium on Theory of computing, pp. 44–61 (1989).

28. Komargodski I., Yogev E.: On distributional collision hashing. In: Advances in Cryptology—CRYPTO 2018 **10992**, 303–327 (2018).
29. Komargodski I., Naor M., Yogev E.: Collision resistant hashing for paranoids: Dealing with multiple collisions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, pp. 162–194 (2018).
30. Lamport L.: Constructing digital signatures from a one way function. Technical report, Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (1979).
31. Leighton F.T., Micali S.: Large provably fast and secure digital signature schemes based on secure hash functions. US Patent 5,432,852.https://www.google.com/patents/US5432852 (1995).
32. Liu Q., Zhandry M.: On finding quantum multi-collisions. In: Advances in Cryptology—EUROCRYPT 2019 **11478**, 189–218 (2019).
33. McGrew D., Curcio M., Fluhrer S.: Leighton-micali hash-based signature. https://www.rfc-editor.org/rfc/rfc8554.html (2019).
34. Merkle R.C.: In: Brassard, G. (ed) Advances in Cryptology—CRYPTO '89 **435**, 218–238 (1989).
35. Pieprzyk J., Wang H., Xing C.: Multiple-time signature schemes against adaptive chosen message attacks. In: SAC 2003: Selected Areas in Cryptography **3006**, 88–100 (2003).
36. Reyzin L.., Reyzin N.: Better than biba: Short one-time signatures with fast signing and verifying. In: ACISP 2002: Information Security and Privacy **2384**, 144–153 (2002).
37. Shor P.W.: Algorithms for quantum computation: discret logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, Ieee, pp. 124–134 (1994).
38. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In: Advances in Cryptology—EUROCRYPT'98 **1403**, 334–345 (1998).
39. Suzuki K., Tonien D., Kurosawa K., Toyota K.: Birthday paradox for multi-collisions. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **91**(1), 39–45 (2008).
40. Yehia M., AlTawy R., Gulliver T.A.: Hash-based signatures revisited. A dynamic FORS with adaptive chosen message security. In: Progress in Cryptology—AFRICACRYPT 2020 **12174**, 239–257 (2020).
41. Zaverucha G.M., Stinson D.R.: Short one-time signatures. Adv. Math. Commun. **5**(3), 473 (2011).
42. Zhandry M.: A note on the quantum collision and set equality problems. Quantum Inf. Comput. **15**(7–8), 557–567 (2015).