Quantum Security of Subset Cover Problems

Samuel Bouaziz-Ermann
LIP6, Paris, France
Sorbonne Université, Paris, France
CNRS, Paris, France

Alex B. Grilo
LIP6, Paris, France
Sorbonne Université, Paris, France
CNRS, paris, France

Damien Vergnaud
LIP6, Paris, France
Sorbonne Université, Paris, France
CNRS, Paris, France

Abstract

The subset cover problem for $k \geq 1$ hash functions, which can be seen as an extension of the collision problem, was introduced in 2002 by Reyzin and Reyzin to analyse the security of their hash-function based signature scheme HORS. The security of many hash-based signature schemes relies on this problem or a variant of this problem (e.g. HORS, SPHINCS, SPHINCS+, ...).

Recently, Yuan, Tibouchi and Abe (2022) introduced a variant to the subset cover problem, called restricted subset cover, and proposed a quantum algorithm for this problem. In this work, we prove that any quantum algorithm needs to make $\Omega\left((k+1)^{-\frac{2^k}{2^{k+1}-1}} \cdot N^{\frac{2^k-1}{2^{k+1}-1}}\right)$ queries to the underlying hash functions with codomain size N to solve the restricted subset cover problem, which essentially matches the query complexity of the algorithm proposed by Yuan, Tibouchi and Abe.

We also analyze the security of the general (r,k)–subset cover problem, which is the underlying problem that implies the unforgeability of HORS under a r-chosen message attack (for $r \geq 1$). We prove that a generic quantum algorithm needs to make $\Omega\left(N^{k/5}\right)$ queries to the underlying hash functions to find a $(1,k)$-subset cover. We also propose a quantum algorithm that finds a (r,k)-subset cover making $O\left(N^{k/(2+2r)}\right)$ queries to the k hash functions.

2012 ACM Subject Classification Security and privacy → Cryptography

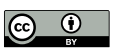
Keywords and phrases Cryptography, Random oracle model, Quantum information

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.9

Related Version *Full Version*: `https://eprint.iacr.org/2022/1474`

Funding This work was partially funded by PEPR integrated project EPiQ ANR-22-PETQ-0007 part of Plan France 2030. This work is part of HQI initiative (www.hqi.fr) and is supported by France 2030 under the French National Research Agency award number "ANR-22-PNCQ-0002".
Alex B. Grilo: ABG is supported by ANR JCJC TCS-NISQ ANR-22-CE47-0004, and by the PEPR integrated project EPiQ ANR-22-PETQ-0007 part of Plan France 2030.

Acknowledgements We thanks the anonymous reviewers for their valuable comments that helped improving the quality of this paper.



1 Introduction

Cryptographic hash functions are functions mapping arbitrary-length inputs to fixed-length outputs and are one of the central primitives in cryptography. They serve as building blocks for numerous cryptographic primitives such as key-establishment, authentication, encryption, or digital signatures. In particular, *one-time signatures* – i.e. in which the signing key can be used only once – based only on hash functions were proposed by Lamport as soon as in 1979 [9]. The basic idea is to evaluate a cryptographic hash function on secret values to generate the public verification key and to authenticate a single message by revealing a subset of those secret pre-images.

With the development of quantum technologies, which may bring drastic attacks against widely deployed cryptographic schemes based on the hardness of integer factorization or the discrete logarithm [12], hash-based signatures have regained interest within the realm of "post-quantum" cryptography and the recent NIST standardization process. In particular, the SPHINCS+ candidate [3] has been selected in 2022 for standardization by NIST and other constructions are standardized by IETF/IRTF. The SPHINCS+ signature scheme and its predecessor SPHINCS [2] make use of a Merkle-hash tree and of HORST, a variant of a hash-based scheme called HORS [11]. HORS (for "Hash to Obtain Random Subset") uses a hash function to select the subset of secret pre-images to reveal in a signature and the knowledge of these secrets for several subsets may not be enough to produce a forgery, a property that makes HORS a *few-time signature* scheme.

More concretely, the security of HORS (and HORST) relies on the hardness of finding a subset cover (SC) for the underlying hash function. More formally, to define the (r, k)*–SC problem*, we consider the hash function as the concatenation of $k \geq 1$ hash functions $h_1, \ldots, h_k$ (with smaller outputs) and the problem is to find, for some integer $r \geq 1$, $r + 1$ elements $x_0, x_1 \ldots, x_r$ in the hash function domain such that $x_0 \notin \{x_1, \ldots, x_r\}$, and

$$\{h_i(x_0) | 1 \leq i \leq k\} \subseteq \bigcup_{j=1}^{r} \{h_i(x_j) | 1 \leq i \leq k\}.$$

The hardness of this problem for concrete popular hash functions has not been studied in depth but Aumasson and Endignoux [1] proved in 2017 a lower bound on the number of queries to hash functions for the SC problem in the Random Oracle Model (ROM). However, the exact security of HORS (and more generally HORST, SPHINCS and SPHINCS+) with respect to quantum attacks is still not clear. Since quantum computing provides speedups for many problems (e.g. Grover's search algorithm [8] and Brassard, Høyer, and Tapp [6] collision search algorithm), it is important to provide lower bounds in a quantum world.

1.1 Our results

In this paper, we explore the difficulty of finding subset cover for idealized hash functions for quantum algorithms. We also consider a variant called the k-restricted subset cover (k–RSC) problem where, given k functions $h_1, \ldots, h_k : \mathcal{X} \to \mathcal{Y}$ such that $N = |\mathcal{Y}|$, one has to find $k + 1$ elements $x_0, x_1 \ldots, x_k$ such that:

$$\forall 1 \leq i \leq k, h_i(x_0) = h_i(x_i)$$

and $x_0 \notin \{x_1, \ldots, x_k\}$. This variant was defined recently by Yuan, Tibouchi and Abe [14], who showed a quantum algorithm to solve it. The main contributions of this work are:

1. **Lower bound on k–RSC**: we prove that $\Omega\left((k+1)^{-\frac{2^k}{2^{k+1}-1}} \cdot N^{\frac{2^k-1}{2^{k+1}-1}}\right)$ quantum queries to the idealized hash functions are needed to find a k–RSC with constant probability. (Theorem 14)
2. **Lower bound on $(1,k)$–SC**: we prove that $\Omega\left((k!)^{-1/5} \cdot N^{k/5}\right)$ quantum queries to the idealized hash functions are needed to find a $(1,k)$–SC with constant probability. (Theorem 21)
3. **Upper bound on (r,k)–SC:** we present a quantum algorithm that finds a (r,k)–SC with constant probability with $O\left(N^{k/(2+2r)}\right)$ queries to the hash functions when k is divisible by $r+1$, and $O\left(N^{k/(2+2r)+1/2}\right)$ otherwise. (Theorem 29)

1.2 Technical Overview

To prove our lower bounds on the query complexity, we use the technique called *compressed random oracle model* introduced by Zhandry in [15]. Its goal is to record information about the queries of an adversary A in the quantum random oracle model and it permits "on-the-fly" simulation of random oracles (or *lazy sampling*) by considering the uniform superposition of all possible random oracles instead of picking a single random oracle at the beginning of the computation. The technique uses a register to keep a record of a so-called *database* of the random oracle and this register is updated whenever A makes a query to the random oracle. At the end of A's computation, the reduction can measure the register of the database, and the distribution of the outputs is uniformly random, as if we had chosen a random oracle at the beginning of its computation. This new register that contains the database is at the gist of our lower bounds.

In Section 3, we prove the lower bound on the query complexity to solve the RSC problem. We consider an algorithm A after i quantum queries to the random oracle and call its state at this moment $|\psi_i\rangle$. Our goal is to compute an upper bound for the value $|P_k^{RSC}|\psi_i\rangle|^2$, where P_k^{RSC} is the projection onto the databases that contain a k–RSC. Computing such a bound leads to a lower bound on the number of queries needed for solving k–RSC with constant probability. To prove our bound, we proceed by induction: assuming we proved a bound for the k'–RSC problem for all $k' < k$, we prove a bound for the k–RSC problem. The analysis is naturally divided into two parts: whenever A finds a k–RSC after i quantum queries, it means that either:

1. A finds it after $i-1$ quantum queries;
2. or A finds it with the i^{th} quantum query.

The first case is recursive and it remains to bound $|P_k^{RSC}|\psi_i\rangle|$ in the second case. Here, the database (after $i-1$ quantum queries) must contain a certain number of k'–RSC (for some $k' < k$), in order for A to find k–RSC with the i^{th} query. Using this strategy, we obtain a recursive formula from which we can deduce the bound on $|P_k^{RSC}|\psi_i\rangle|$.

In Section 4.1, we prove a lower bound for the $(1,k)$–SC problem. The idea of the proof is similar to the proof for the lower bound of the k–RSC problem but we have to compute a bound for another problem that we define: the *j–repetition* problem.

Finally in Sections 4.2 and 4.3, we design a family of quantum algorithms for finding a (r,k)–SC. These algorithms are inspired by the algorithm from [14] to solve the k–RSC problem and [10]'s algorithm for finding multi-collisions. These algorithms are recursive and take as input two parameters $t, k' \in \mathbb{N}$ and perform the following:

1. Find t distinct $(r-1,k')$–SC;
2. Find the (r,k)–SC.

The parameters t and k' are chosen in order to optimize the complexity of the algorithm. The first step is done by applying $r-1$ times the algorithm for the value k', and the second step uses Grover's algorithm [8][5].

Full version of the paper

In this paper, most of the proofs are omitted in the interest of space. The proofs of all the lemmas and theorems stated in this paper can be found in the full version of the paper, available on eprint.

1.3 Related works, discussion and open problems

Collision-finding

The link between finding a multi-collision and finding a subset cover was first discussed in [14], since their algorithm is inspired from the one for finding multi-collisions in [10]. In the latter, they also show a lower bound for finding multi-collisions, and our proof of lower bounds uses the same technique they used. We make use of the compressed oracle technique, first introduced by Zhandry in [15], and generalize the proof of the lower bound on multi-collisions to the RSC and SC problems.

Restricted Subset Cover

There is currently only one quantum algorithm for finding RSC [14]. Our lower bound for finding a RSC matches their upper bound when k, the number of functions, is constant. However when k is not a constant, their algorithm makes $O\left(k \cdot N^{\frac{2^k-1}{2^{k+1}-1}}\right)$ queries to $h_1, \ldots, h_k$, which roughly leaves a $k^{3/2}$ gap between the best known attack and our lower bound. To the best of our knowledge, this is the first lower bound on the RSC problem for a quantum algorithm, and there are no such result for classical algorithms. It would be interesting to see if we can close this gap further.

Tighter bounds for $(1,k)$–SC

When k is constant, the lower bound for $(1,k)$–SC is $\Omega\left(N^{k/5}\right)$, while our algorithm for this problem makes $O\left(N^{k/4}\right)$ queries to the oracle (when k is even). It would be interesting to tighten this gap, especially since the results for $(1,k)$–SC are probably necessary to prove the lower bounds (r,k)–SC for $r \geq 2$.

For non-constant k, our lower bound for $(1,k)$–SC is $\Omega\left(C_k^{-1/5} \cdot N^{k/5}\right)$, where $C_k = \sum_{j=2}^{k} \frac{k!}{(j-1)!} \leq k! \cdot e$. Notice that this term cannot be neglected for large values of k. For example with $k = \log(N)$, we have $C_k \geq N$. In comparison, our best algorithm for $(1,k)$–RSC, the factor in k is $\binom{k}{(k+1)/2}^{-1/2} \leq \frac{2^{(k+1)/2}}{\left(\frac{k+1}{2} \cdot \pi\right)^{1/4}}$, which is very far from our bound on C_k. It would also be interesting to see if we can tighten this gap.

Bounds for (r,k)–SC

Unfortunately, expanding our result for the (r,k)–SC problem is much more complicated than the case $r = 1$ and actually even proving the case $r = 2$ is not simple. To prove such a result, one would need a bound for the problem of finding j distinct $(1,k)$–SC problem. While proving such a bound is challenging, it is also unclear what the problem of finding

j distinct $(1,k)$–SC is. Indeed, an important property for our technique in the first lower bound proofs is that by making one query to the oracle, the adversary cannot find two or more k–RSC. The same property must hold for the problem of finding j distinct $(1,k)$–SC, and this definition and subsequent analysis remain open.

Security of SPHINCS and SPHINCS+

The signature scheme SPHINCS relies on the HORST scheme (for "HORS with trees") which adds a Merkle tree to the HORS scheme to compress the public key. The security of HORST also relies on the (r,k)–SC problem but the security of SPHINCS rely on different security notions of the underlying hash functions. In particular, it depends on a variation of the SC problem classed the *target subset cover* (TSC) problem [11]. The main difference comes from the fact that the message signed using HORST is an unpredictable function of the actual message and this prevents an attacker to construct a subset cover beforehand.

Nevertheless, the authors of [2] stated an existential unforgeability result for SPHINCS [2, Theorem 1] under q_s-adaptive chosen message attacks. The success probability in such attacks is roughly upper-bounded by:

$$\sum_{r=1}^{\infty} \min\left(2^{r(\log q_s - h)+h}, 1\right) \cdot Succ_A((r,k) - SC),$$

where h is the height of the tree used in SPHINCS, and $Succ_A((r,k) - SC)$ denotes the success probability of an adversary A to find a (r,k)–SC. The authors made the assumption that this term is negligible for any probabilistic adversary A and our quantum lower bound on the query number to find a $(1,k)$–SC can be seen as a first step towards proving this assumption (for idealized hash functions). To assess the security of SPHINCS from [2, Theorem 1] for concrete parameters such as those proposed in [2] (namely $h = 60, q_s = 2^{30}$), it would also be necessary to upper-bound the success probabilities $Succ_A((2,k) - SC)$ and $Succ_A((3,k) - SC)$, which we leave for future work. For example, one could try to apply [13, Theorem 4.12] to get a lower bound for (r,k)–SC more easily, but the obtained bound will most likely not be tight.

SPHINCS+ is an enhancement of SPHINCS, which makes the scheme more efficient and its security relies on another variant of the SC problem, namely the interleaved target subset cover (ITSC) problem. It would also be interesting to see if our methods can be used to prove similar bounds for the TSC and ITSC problems. At last, one could also try to design algorithms for these two problems, as no quantum algorithms for them exist yet to the best of our knowledge.

2 Preliminaries

We assume the reader is familiar with the theory of quantum information. We denote the concatenation by $||$.

2.1 Compressed oracle technique

We now present the key ingredients of Zhandry's compressed oracle technique, first defined in [15] and refined in [7]. As mentioned in the introduction, the technique uses a register to keep a record of a so-called *database* of the random oracle and this register is updated whenever an adversary A makes a query to the random oracle. This new register that contains the database is at the gist of our lower bounds.

We consider the *Quantum Random Oracle Model*, first defined in [4]. In this model, we are given black-box access to a *random* function $H : \mathcal{X} \to \mathcal{Y}$. For our model, the adversary will work on three different registers $|x, y, z\rangle$. The first register is the query register, the second register is the answer register and the third register is the work register. The first two registers are used for queries and answers to the oracle, while the last register is for the adversary's other computations. We first define the unitary StO that represents the *Standard Oracle* and that computes as follows:

$$\mathsf{StO} \sum_{x,y,z} \alpha_{x,y,z} |x, y, z\rangle \to \sum_{x,y,z} \alpha_{x,y,z} |x, y + H(x), z\rangle$$

This unitary corresponds to a query to H.

Now, we define Zhandry's compressed oracle. In this model, instead of starting with a random function H, we start with the uniform superposition of all random functions $|H\rangle$, where $|H\rangle$ encodes the truth table of the function H. In this model, there is a register for each $x \in \mathcal{X}$, and the value of this register in the state $|H\rangle$ corresponds to $H(x)$. That is, we have that $|H\rangle = \bigotimes_{x \in \mathcal{X}} |H(x)\rangle_x$ Let $\mathcal{H} = \{H : \mathcal{X} \to \mathcal{Y}\}$ be the set of all possible functions H. We define a new register, the database register $|H\rangle$, that starts in the uniform superposition $\frac{1}{|\mathcal{H}|} \sum_{H \in \mathcal{H}} |H\rangle$. This register starts in product state with the other registers, and Zhandry's idea is that instead of modifying the adversary's register when querying the oracle, we will modify the database register instead. To do so, we simply consider the *Fourier basis* for the y and the H register before querying the Standard Oracle.

We write this unitary O and it works as follows:

$$\mathsf{O} \sum_{x,\hat{y},z} \alpha_{x,\hat{y},z} |x, \hat{y}, z\rangle \otimes \sum_{\hat{H} \in \mathcal{H}} \alpha_{\hat{H}} \left|\hat{H}\right\rangle \to \sum_{x,\hat{y},z} \alpha_{x,\hat{y},z} |x, \hat{y}, z\rangle \otimes \sum_{\hat{H} \in \mathcal{H}} \alpha_{\hat{H}} \left|\hat{H} \ominus (x, \hat{y})\right\rangle,$$

where, for any fixed $x \in \mathcal{X}$ and $z \in \mathcal{Y}$, $H \ominus (x, z) : \mathcal{X} \to \mathcal{Y}$ is defined as:

$$H \ominus (x, z)(x') = \begin{cases} H(x') & \text{if } x' \neq x \\ H(x) - z & \text{if } x' = x. \end{cases}$$

In other words, $H \ominus (x, z)$ is obtained by replacing the value of $H(x)$ by $H(x) - z$ in H.

This unitary can be implemented by applying the QFT to the registers $|y\rangle$ and $|H\rangle$, applying the Standard Oracle, then applying the $QFT^\dagger$ again on the $|y\rangle$ and $|H\rangle$ registers.

Finally, we define the compression part. The idea behind the compression is that for every x in the database mapped to $\left|\hat{0}\right\rangle$, we remap it to $|\perp\rangle$, where $\perp$ is a new value outside of $\mathcal{Y}$. More formally, the compression part is done by applying:

$$\mathsf{Comp} = \bigotimes_x \left(|\perp\rangle \left\langle\hat{0}\right| + \sum_{\hat{y}:\hat{y} \neq \hat{0}} |\hat{y}\rangle \langle\hat{y}| \right)$$

in the Fourier basis.

Since at the start of the computation, the database will be initiated with the uniform superposition over all $\mathcal{H}$ possible, then after q queries the state of the database can be described with q vectors. In order to apply the compression as a unitary, we declare that $\mathsf{Comp} |\perp\rangle = \left|\hat{0}\right\rangle$.

Now, we can define the *Compressed Oracle*:

$$\mathsf{cO} = \mathsf{Comp} \circ \mathsf{O} \circ \mathsf{Comp}^\dagger.$$

Of course the compression part inevitably creates some losses, compared to only using the Standard Oracle. The precise characterization of these losses is given in one of Zhandry's lemma, and can be stated as follows:

▶ **Lemma 1** (Lemma 5 from [15]). *Let A be an algorithm that makes queries to a random oracle $H : \mathcal{X} \to \mathcal{Y}$, and output $(x_1, \ldots, x_k, y_1, \ldots, y_k) \in \mathcal{X}^k \times \mathcal{Y}^k$. Let p be the probability that $\forall 1 \leq i \leq k$, $H(x_i) = y_i$. Similarly, consider the algorithm A running with the Compressed Oracle cO, and output $(x'_1, \ldots, x'_k, y'_1, \ldots, y'_k) \in \mathcal{X}^k \times \mathcal{Y}^k$. Let p' be the probability that $\forall 1 \leq i \leq k$, $H'(x'_i) = y'_i$, where H' is obtained by measuring the H register at the end of the execution of the algorithm A. Then:*

$$\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{k}{|\mathcal{Y}|}}.$$

In the rest of the paper, we will have that $\sqrt{\frac{k}{|\mathcal{Y}|}}$ is negligible, and thus we will neglect this term.

We also have the following lemma from [7] that describes the operator $\mathsf{cO}_{(x,\hat{y})} : \mathcal{H} \to \mathcal{H}$, which is defined as the operator applied on $|H\rangle$ when applying cO to $|x\rangle |\hat{y}\rangle \otimes |H\rangle$. More formally, we have that:

$$\mathsf{cO} |x\rangle |\hat{y}\rangle \otimes |H\rangle = |x\rangle |\hat{y}\rangle \otimes \mathsf{cO}_{(x,\hat{y})} |H\rangle .$$

▶ **Lemma 2** (Lemma 4.3 from [7]). *For any $\hat{y} \neq \hat{0}$, the operator $\mathsf{cO}_{(x,\hat{y})}$ is represented by the following matrix:*

	$\perp$	r
$\perp$	0	$\frac{\omega_N^{-ry}}{\sqrt{\lvert\mathcal{Y}\rvert}}$
y'	$\frac{\omega_N^{yy'}}{\sqrt{\lvert\mathcal{Y}\rvert}}$	$\begin{cases} \left(1 - \frac{2}{\lvert\mathcal{Y}\rvert}\right)\omega_N^{yy'} + \frac{1}{\lvert\mathcal{Y}\rvert} & \text{if } y' = r \\ \frac{1-\omega_N^{yy'}-\omega_N^{ry}}{\lvert\mathcal{Y}\rvert} & \text{if } y' \neq r \end{cases}$

For $\hat{y} = \hat{0}$, we have that $\mathsf{cO}_{(x,\hat{0})}$ is the identity.

We also define, for any compressed $H : \mathcal{X} \to \mathcal{Y} \cup \{\perp\}$, for any fixed $x \in \mathcal{X}$ and $z \in \mathcal{Y}$, $H \cup (x, z) : \mathcal{X} \to \mathcal{Y}$ as:

$$H \cup (x, z)(x') = \begin{cases} H(x') & \text{if } x' \neq x \\ z & \text{if } x' = x. \end{cases}$$

In other words, $H \cup (x, z)$ is obtained by replacing the value of $H(x)$ by z in H.

In the following, we will model the adversary (A) as a series of computation alternating between unitaries and oracle calls. The adversary's quantum state will first be initialized to $|0\rangle^{\otimes N}$. Then, his computation will be decomposed as:

$$A = U_k \mathsf{cO} U_{k-1} \mathsf{cO} \ldots \mathsf{cO} U_2 \mathsf{cO} U_1 \tag{1}$$

So that, if $|\psi_i\rangle = \sum_{x,y,z,D} \alpha_{x,y,z,D} |x, y, z, D\rangle$ is the state of the adversary after i quantum queries to cO, then U_{i+1} operates on the registers x, y and z only. We also define *database properties*:

▶ **Definition 3** (Database property). *A database property is a subset of $\mathcal{H}$. Any database property D can be seen as a projector on $\mathcal{H}$, as follows:*

$$\sum_{d\in D} |d\rangle\langle d|$$

We write $\mathcal{D} = \{I | I \subseteq \mathcal{H}\}$ the set of all subspaces of $\mathcal{H}$, that also corresponds to the set of all database properties.

We now state and prove two lemmas adapted from [10] that we will use thoroughly in this paper. The first lemma will allow us to ignore the unitaries that the adversary A applies on the first registers of the state.

▶ **Lemma 4** (adapted from Lemma 8 from [10]). *For any unitary U, any projector P, and any state $|\phi\rangle$,*

$$|(I\otimes P)\cdot(U\otimes I)|\phi\rangle| = |(I\otimes P)|\phi\rangle|$$

The second lemma bounds the amplitude of measuring a database that satisfies a property P at the i^{th} step of the algorithm, i.e. just after the i^{th} query to the oracle. In this bound, the first term captures the case where we succeed to find a database that satisfies P before the i^{th} query. The second term captures the case where we did not have it before the i^{th} query, but found it with the i^{th} one.

▶ **Lemma 5** (adapted from Lemma 9 from [10]). *Let $|\phi_i\rangle$ be the state of an algorithm A just before the i^{th} quantum query to cO, and $|\psi_i\rangle$ the state of the same algorithm right after the i^{th} quantum query to cO. Let P be any projector on D. We have that:*

$$|P|\psi_i\rangle| \leq |P|\phi_i\rangle| + |P\mathsf{cO}(I-P)|\phi_i\rangle|$$

Proof.

$$\begin{aligned}|P|\psi_i\rangle| = |P\mathsf{cO}|\phi_i\rangle| &= |P\mathsf{cO}(P|\phi_i\rangle + (I-P)|\phi_i\rangle)| \\ &\leq |P|\phi_i\rangle| + |P\mathsf{cO}(I-P)|\phi_i\rangle)|\,,\end{aligned}$$

where the inequality comes from the triangle inequality and the fact that $P\mathsf{cO}P \leq P$. ◀

▶ Remark. In the next section and in the rest of the paper, we will consider multiple functions $h_1,\ldots,h_k : \mathcal{X}\to\mathcal{Y}$ for some fixed k. Note that this is equivalent to considering one function $H : \mathcal{X}\to\mathcal{Y}^k$, such that we interpret, for any $x\in\mathcal{X}$, the output $H(x)$ as the concatenation of values of the functions applied to x, i.e. $H(x) = h_1(x)||h_2(x)||\cdots||h_k(x)$. Hence, in this setting, the compressed oracle is used on the function H, and a query to any of the h_i is a query to all of the h_i's. Thus, in our results, we count the number of queries to the function H and thus the number of queries to all of the h_i's. It may seem that we lose some accuracy in this setting, however this is with the same method that multiple random functions are implemented in the literature.

2.2 The problem of subset cover and its variants

We define the problem of subset cover.

▶ **Definition 6** ((r,k)–SC). *Let $k, r \in \mathbb{N}^*$. Let $h_1,\cdots,h_k : \mathcal{X}\to\mathcal{Y}$. A (r,k)–SC for $(h_1,\cdots,h_k)$ is a set of $r+1$ elements $x_0,x_1,x_2,\cdots,x_r$ in $\mathcal{X}$ such that:*

$$\{h_i(x_0) | 1\leq i\leq k\} \subseteq \bigcup_{j=1}^{r} \{h_i(x_j) | 1\leq i\leq k\}$$

In other words, for each $1 \leq i \leq k$, there exists a $1 \leq j \leq r$ and a $1 \leq \ell \leq k$ such that $h_i(x_0) = h_\ell(x_j)$.

We notice two facts regarding the parameters of (r,k)–SC. First, we have that the problem becomes easier when r increases. Secondly, we have that when $r > k$, a (r,k)–SC contains a (k,k)–SC. Thus finding a (r,k)–SC when $r > k$ is the same as when $r = k$. For simplicity, we use k–SC as a shorthand of (k,k)–SC.

We also define the database properties $P^{SC}_{(r,k)}$ of containing a (r,k)–SC, that is the set of databases that contains a (r,k)–SC. More formally, we have that:

$$P^{SC}_{(r,k)} = \left\{ D \in \mathcal{D} \middle| \exists x_0, x_1, \ldots, x_r, \forall i \neq 0, x_0 \neq x_i, H(x_0) \subseteq \bigcup_{i=1}^{r} H(x_i) \right\},$$

where for $x \in \mathcal{X}, H(x) = \{h_1(x), \ldots, h_k(x)\}$.

We follow now with the definition of a harder variation of the k-subset cover called the *k-restricted subset cover* (k–RSC).

▶ **Definition 7** (k–RSC). *Let $k \in \mathbb{N}^*$. Let $h_1, \ldots, h_k : \mathcal{X} \to \mathcal{Y}$. A* k-restricted subset cover *(k–RSC) for $(h_1, \ldots, h_k)$ is a set of $k+1$ elements $x_0, x_1, x_2, \ldots, x_k$ in $\mathcal{X}$ such that:*

$$\forall i \in \{1, \ldots, k\}, h_i(x_0) = h_i(x_i) \text{ and } x_0 \neq x_i.$$

We also define the database properties $P^{RSC}_{k,\ell}$ of k distinct ℓ–RSC, that is the set of databases that contains k distinct ℓ–RSC. More formally, we have that:

$$P^{RSC}_{k,\ell} = \left\{ D \in \mathcal{D} \middle| \begin{array}{l} \exists x_{0,1}, \ldots, x_{\ell,1}, \forall i \neq 0, x_{0,1} \neq x_{i,1}, \forall i, h_i(x_{0,1}) = h_i(x_{i,1}) \\ \exists x_{0,2}, \ldots, x_{\ell,2}, \forall i \neq 0, x_{0,2} \neq x_{i,2}, \forall i, h_i(x_{0,2}) = h_i(x_{i,2}) \\ \vdots \\ \exists x_{0,\ell}, \ldots, x_{\ell,k}, \forall i \neq 0, x_{0,k} \neq x_{i,k}, \forall i, h_i(x_{0,k}) = h_i(x_{i,k}) \\ \forall i \neq j, (h_1(x_{0,i}), \ldots, h_\ell(x_{0,i})) \neq (h_1(x_{0,j}), \ldots, h_\ell(x_{0,j})) \end{array} \right\} \tag{2}$$

The problem of finding a k–RSC was introduced in [14], in which the authors describe an algorithm that finds a k–RSC in $O\left(kN^{\frac{1}{2}\left(1-\frac{1}{2^{k+1}-1}\right)}\right)$ quantum queries to $h_1, \ldots, h_k$ when the h_i's are such that $|\mathcal{X}| \geq (k+1)|\mathcal{Y}|$.

We discuss now the last condition in Equation (2). We remark that while such condition was not explicitly imposed in [10] for their lower bound for finding multi-collisions, this property is implicitly and extensively used in their proof. Such a property is needed because when they count k-collisions (that is, k distinct $x_1, \ldots, x_k$ such that $H(x_1) = \cdots = H(x_k)$), they are actually interested in the number of possible *images* that would be helpful to reach a $(k+1)$-collision. In particular, this is helpful since one query can only transform *one* k-collision (with such a property) into a $(k+1)$-collision.

In our case, the last line of (2) ensures that the "supporting set" of the k–RSC (i.e. the set of images of the $x_{0,i}$ by the different random functions $h_1, \ldots, h_k$) is unique. As in the multi-collision case, this condition will be crucial to extend a k–RSC to a $(k+1)$–RSC, and for this reason we define it explicitly in $P^{RSC}_{k,\ell}$.

Finally, we state a result from [10], regarding the amplitude of finding j distinct 2-collisions:

▶ **Lemma 8** (adapted from [10], Corollary 11). *Given a random function $h : \mathcal{X} \to \mathcal{Y}$ where $|N| = \mathcal{Y}$, let $f^{col}_{i,j}$ be the amplitude of the D containing at least j distinct 2-collisions after i quantum queries. Then:*

$$f^{col}_{i,j} \leq \left(\frac{4e \cdot i^{3/2}}{j\sqrt{N}}\right)^j.$$

3 Lower bound on the k-restricted subset cover problem

In this section, we prove a lower bound for the k–RSC problem defined in Definition 7. This section follows closely [10]'s proof of their lower bound on finding multi-collisions. We first prove a lower bound for finding k distinct 2–RSC, which will be necessary in our induction step. Finally, we will prove the induction step in the last subsection and obtain a lower bound on finding s distinct k–RSC.

3.1 Finding k distinct 2-restricted subset cover

We want to bound the number of queries needed to find k distinct triplets that satisfy a 2–RSC. We have the following result:

▶ **Theorem 9.** *Given two random functions $h_1, h_2 : \mathcal{X} \to \mathcal{Y}$ where $N = |\mathcal{Y}|$, a quantum algorithm needs to make $\Omega(k^{4/7} \cdot N^{3/7})$ queries to h_1 and h_2 to find k distinct 2–RSC with constant probability, for any $k \leq N^{1/8}$.*

To prove this theorem, we first introduce some notation. We denote $P_{2,k,\ell}$ the set of databases that satisfies k distinct 2–RSC, and that contain exactly ℓ collisions on h_1. We denote $g_{i,k} = \left| P_{k,2}^{RSC} |\psi_i\rangle \right|$ and $\widehat{g}_{i,k,\ell} = |P_{2,k,\ell} |\psi_i\rangle|$, where $|\psi_i\rangle$ is the state just after the i^{th} query to $H = (h_1, h_2)$.

Our goal is to bound $g_{i,k}$, and we will first prove a recursive formula stated in the next lemma.

▶ **Lemma 10.** *For every $i \in \mathbb{N}$, and every $k \in \mathbb{N}$, we have that:*

$$g_{i,k} \leq g_{i-1,k} + \sqrt{2 \sum_{\ell \geq 0} \frac{\ell}{N} \widehat{g}_{i-1,k-1,\ell}^2} + \frac{(i-1)}{N} g_{i-1,k-1}.$$

We will split the sum in two using $\mu_3(j)$ as a threshold. We also define a new notation that will simplify expressions:

▶ **Definition 11.**

$$A_i = \sum_{\ell=0}^{i-1} \sqrt{2} \left(\sqrt{\frac{\mu_3(\ell-1)}{N}} + \sqrt{8} \frac{\ell - 1}{N} \right),$$

where

$$\mu_3(\ell) = \max \left\{ 8e \frac{\ell^{3/2}}{\sqrt{N}}, 10N^{1/8} \right\}.$$

Before bounding $g_{i,k}$, we first prove a bound on A_i.

▶ **Lemma 12.** *For every $i \in \mathbb{N}$, we have that:*

$$A_i \leq 8\sqrt{e} \frac{i^{7/4}}{N^{3/4}} + 4\frac{i^2}{N} + O\left(N^{-1/48}\right).$$

It follows that $A_i < 2eN^{1/8}$ for $i \leq N^{1/2}$.

We can now state the lemma that bounds $g_{i,k}$.

▶ **Lemma 13.** *For every $i \in \mathbb{N}$ and $k \in \mathbb{N}$, we have that:*

$$g_{i,k} < \frac{A_i^k}{k!} + \sqrt{2} \cdot 2^{-N^{1/8}}.$$

We can now prove the main theorem of this subsection.

Proof of Theorem 9. Following from Lemma 13, we have that:

$$g_{i,k} \leq \frac{A_i^k}{k!} + \sqrt{2} \cdot 2^{-N^{1/8}} \leq \left(\frac{A_i \cdot e}{k}\right)^k + \sqrt{2} \cdot 2^{-N^{1/8}}.$$

We now use the bound on A_i of Lemma 12:

$$g_{i,k} \leq \left(\frac{8e^{3/2}}{k} \cdot \frac{i^{7/4}}{N^{3/4}} + \frac{4e}{k} \cdot \frac{i^2}{N} + \frac{e}{k} \cdot O\left(N^{-1/48}\right)\right)^k + \sqrt{2} \cdot 2^{-N^{1/8}}.$$

So if $i = o(k^{4/7} \cdot N^{3/7})$, then $g_{i,k} = o(1)$. Hence if we want $g_{i,k}$ to be a constant, i.e. not $o(1)$, we must have $i = \Omega\left(k^{4/7} \cdot N^{3/7}\right)$. ◀

3.2 Finding k distinct s-restricted subset cover

In this section, we generalize the result to the problem of finding k distinct s–RSC, for any $s \geq 3$ and any $k \geq 1$. We are given s random functions $h_1, \ldots, h_s$ such that for any $i \in [1, s]$, $h_i : \mathcal{X} \to \mathcal{Y}$. We will prove the following theorem.

▶ **Theorem 14.** *Given s random functions $h_1, \ldots, h_s : \mathcal{X} \to \mathcal{Y}$ where $N = |\mathcal{Y}|$, a quantum algorithm needs to make $\Omega\left((s+1)^{-\frac{2^s}{2^{s+1}-1}} \cdot k^{\frac{2^s}{2^{s+1}-1}} \cdot N^{\frac{2^s-1}{2^{s+1}-1}}\right)$ queries to $h_1, \ldots, h_s$ to find k distinct s–RSC with constant probability, for any $s \leq \log(\log(N))$ and any $k \geq N^{1/2^{s+1}}$.*

And naturally we have the following corollary for $k = 1$:

▶ **Corollary 15.** *Given s random functions $h_1, \ldots, h_s : \mathcal{X} \to \mathcal{Y}$ where $N = |\mathcal{Y}|$, a quantum algorithm needs to make $\Omega\left((s+1)^{-\frac{2^s}{2^{s+1}-1}} \cdot N^{\frac{2^s-1}{2^{s+1}-1}}\right)$ queries to $h_1, \ldots, h_s$ to find one s–RSC with constant probability, for any $s \leq \log(\log(N))$.*

In order to prove Theorem 14, we first define some notations, starting with the notations for the amplitudes. We define:

1. $f_{i,j}$ as the amplitude of the databases D containing at least j distinct $(s-1)$–RSC after i quantum queries.
2. $\widehat{g}_{i,j,k}$ as the amplitude of the databases D containing at least j distinct $(s-1)$–RSC and exactly k distinct s–RSC after i quantum queries.
3. $g_{i,k}$ as the amplitude of the databases D containing exactly k distinct s–RSC after i quantum queries.

More formally, let $|\phi_i\rangle$ (resp. $|\psi_i\rangle$) be the state of the algorithm just before (resp. after) the i^{th} query to the oracle. We have:

$$f_{i,j} = \left|P^{RSC}_{j,(s-1)} |\psi_i\rangle\right|,$$
$$\widehat{g}_{i,j,k} = \left|P^{RSC}_{j,(s-1)} P^{RSC}_{k,s} \neg P^{RSC}_{k+1,s} |\psi_i\rangle\right|,$$
$$g_{i,k} = \left|P^{RSC}_{k,s} \neg P^{RSC}_{k+1,s} |\psi_i\rangle\right|.$$

We want to bound $g_{i,k}$, and to do so, we define some convenient notation. We start by defining Π_s, a term that appears in the bound of $g_{i,k}$.

▶ **Definition 16.** *Let Π_s be defined as follows:*

$$\begin{cases} \Pi_1 = 1 \\ \Pi_2 = 1 \\ \forall s \geq 2, \quad \Pi_{s+1} = 2 \cdot \sqrt{s} \cdot \sqrt{\Pi_s} \end{cases}$$

We define $A_{i,s}$ and $\mu_s(\ell)$ as follows:

▶ **Definition 17.**

$$A_{i,s} = \sum_{\ell=0}^{i-1} B_{\ell,s-1},$$

where

$$B_{\ell,s} = \sqrt{s \cdot \frac{\mu_{s+1}(\ell)}{N}} + 4\left(\frac{\ell}{N}\right)^{s/2} + \left(\sum_{r=2}^{s} \frac{\ell}{N^r}\right)^{1/2},$$

and

$$\mu_s(\ell) = \max\left\{\Pi_{s-1} \cdot (8e)^{\frac{2^{s-2}-1}{2^{s-3}}} \frac{\ell^{(2^{s-1}-1)/2^{s-2}}}{N^{(2^{s-2}-1)/2^{s-2}}}, 40 \cdot s^2 \cdot \Pi_{s-1} \cdot N^{1/2^s}\right\}.$$

We can now state the bound on $g_{i,k}$ that we will need to prove Theorem 14:

▶ **Lemma 18.** *For every $i \in \mathbb{N}$ and every $k \in \mathbb{N}$, we have that:*

$$g_{i,k} \leq \frac{A_{i,s+1}^k}{k!} + O\left(2^{-(s+1)^2 \cdot \Pi_s \cdot N^{1/2^{s+1}}}\right).$$

In order to prove Lemma 18, we first prove a bound on $A_{i,s}$.

▶ **Lemma 19.** $A_{i,s} \leq (8e)^{\frac{2^{s-2}-1}{2^{s-2}}} \frac{i^{(2^s-1)/2^{s-1}}}{N^{(2^{s-1}-1)/2^{s-1}}} \cdot \Pi_s + O\left(s^4 \cdot \Pi_s \cdot N^{-1/(2^s(2^s-2))}\right)$

At last we bound Π_s to conclude the analysis.

▶ **Proposition 20.** *We have for any $s \in \mathbb{N}$ that:*

$$\Pi_s \leq 4s$$

Proof. The statement is true for $s = 1, 2$. Assume it is true for $s \geq 2$. Then,

$$\Pi_{s+1} = 2\sqrt{s} \cdot \sqrt{\Pi_s} \leq 2\sqrt{s} \cdot \sqrt{4s} \leq 4(s+1). \qquad \blacktriangleleft$$

Finally, we can prove Theorem 14:

Proof of Theorem 14. From Lemma 19, we have:

$$A_{i,s} \leq (8e)^{\frac{2^{s-2}-1}{2^{s-2}}} \frac{i^{(2^s-1)/2^{s-1}}}{N^{(2^{s-1}-1)/2^{s-1}}} \cdot \Pi_s + O\left(s^4 \cdot \Pi_s \cdot N^{-1/(2^s(2^s-2))}\right).$$

Hence we can bound $g_{i,k}$ for any i, k, by:

$$\begin{aligned}
g_{i,k} &\leq \frac{A_{i,s+1}^k}{k!} + O\left(2^{-(s+1)^2\cdot\Pi_s\cdot N^{1/2^{s+1}}}\right) \\
&\leq \left(\frac{e\cdot A_{i,s+1}}{k}\right)^k + O\left(2^{-(s+1)^2\cdot\Pi_s\cdot N^{1/2^{s+1}}}\right) \\
&\leq \left(\frac{e}{k}(8e)^{\frac{2^{s-1}-1}{2^{s-1}}}\frac{i^{(2^{s+1}-1)/2^s}}{N^{(2^s-1)/2^s}}\cdot\Pi_{s+1} + \frac{e}{k}\cdot O\left((s+1)^4\Pi_{s+1}\cdot N^{-1/(2^{s+1}(2^{s+1}-2))}\right)\right)^k \\
&\quad + O\left(2^{-(s+1)^2\cdot\Pi_s\cdot N^{1/2^{s+1}}}\right) \\
&\leq \left(\frac{e}{k}\cdot(8e)^{\frac{2^{s-1}-1}{2^{s-1}}}\frac{i^{(2^{s+1}-1)/2^s}}{N^{(2^s-1)/2^s}}\cdot 4(s+1) + \frac{e}{k}\cdot O\left(4(s+1)^5\cdot N^{-1/(2^{s+1}(2^{s+1}-2))}\right)\right)^k \\
&\quad + O\left(2^{-4s(s+1)^2\cdot N^{1/2^{s+1}}}\right),
\end{aligned}$$

where the first inequality comes from Lemma 18, the third inequality comes from Lemma 19 and the last inequality comes from Proposition 20.

If $i = o\left((s+1)^{-\frac{2^s}{2^{s+1}-1}}\cdot k^{\frac{2^s}{2^{s+1}-1}}\cdot N^{\frac{2^s-1}{2^{s+1}-1}}\right)$, then $g_{i,k} = o(1)$. Hence if we want $g_{i,k}$ to be constant, i.e. not $o(1)$, we must have $i = \Omega\left(s^{-\frac{2^s}{2^{s+1}-1}}\cdot k^{\frac{2^s}{2^{s+1}-1}}\cdot N^{\frac{2^s-1}{2^{s+1}-1}}\right)$. ◀

4 The (r,k)-subset cover problem

In this section, we prove some upper and lower bounds on the (r,k)–SC problem. As far as we know, there is no quantum algorithm to find a (r,k)–SC problem, except for [14]'s algorithm when $k = r$, and for the harder problem of finding a k–RSC. We first prove a lower bound on the $(1,k)$–SC problem, then design new algorithms for finding a (r,k)–SC.

4.1 Lower bound on finding a $(1,k)$-subset cover

In this subsection, we will prove a lower bound on the $(1,k)$–SC problem. We are given k random functions $h_1,\ldots,h_k$ such that for $i\in[1,k]$, $h_i : \mathcal{X}\to\mathcal{Y}$. We write $N = |\mathcal{Y}|$ and for $x\in\mathcal{X}$, we write $H(x) = \{h_i(x) | i\in[1,k]\}$. The goal of this subsection is to prove the following theorem.

▶ **Theorem 21.** *Given k random functions $h_1,\ldots,h_k : \mathcal{X}\to\mathcal{Y}$ where $N = |\mathcal{Y}|$, a quantum algorithm needs to make $\Omega\left(C_k^{-1/5}\cdot N^{k/5}\right)$ queries to $h_1,\ldots,h_k$ to find one (1,k)–SC with constant probability, where $C_k = \sum_{j=2}^k \frac{k!}{(j-1)!}$.*

To prove Theorem 21, we introduce the problem of finding a *j-repetition* on $h_{i_1},\ldots,h_{i_j}$, that consists in finding an $x\in\mathcal{X}$ such that $h_{i_1}(x) = \cdots = h_{i_j}(x)$. More formally, we define the following database property:

▶ **Definition 22.**

$$\forall\ell, j, P_{\ell,j}^{rep} = \left\{ D\in\mathcal{D} \;\middle|\; \begin{array}{l} \exists x_1, x_2,\ldots,x_\ell, \forall i, \forall 1\leq\ell\leq j, h_1(x_i) = h_\ell(x_i) \\ \forall i\neq p, x_i\neq x_p \end{array} \right\}.$$

Note that we define the property only for ℓ distinct j-repetition on $h_1, \ldots, h_j$, because by symmetry, the probability of finding a j-repetition on $h_1, \ldots, h_j$ is the same as finding a j-repetition on $h_{i_1}, \ldots, h_{i_\ell}$.

We also define:

1. $\widetilde{f}^{rep}_{i,\ell,j}$ as the amplitude of the databases D containing *at least* ℓ distinct j-repetitions on $h_1, \ldots, h_j$ after i quantum queries.
2. $f^{rep}_{i,\ell,j}$ as the amplitude of the databases D containing *exactly* ℓ distinct j-repetitions on $h_1, \ldots, h_j$ after i quantum queries.
3. $g_{i,k}$ as the amplitude of the databases D containing at least one $(1,k)$-SC after i quantum queries.

More formally, let $|\psi_i\rangle$ be the state just after the i^{th} query to the oracle, then $\widetilde{f}^{rep}_{i,\ell,j} = \left|P^{rep}_{\ell,j}|\psi_i\rangle\right|$, $f^{rep}_{i,\ell,j} = \left|P^{rep}_{\ell,j}\neg P^{rep}_{\ell+1,j}|\psi_i\rangle\right|$, and $g_{i,k} = \left|P^{SC}_{(1,k)}|\psi_i\rangle\right|$.

Our goal is to bound $g_{i,k}$ and for that we will bound $\widetilde{f}^{rep}_{i,\ell,j}$.

▶ **Lemma 23.** *For all $i, \ell, j \in \mathbb{N}$, we have that:*

$$\widetilde{f}^{rep}_{i,\ell,j} \leq \left(\frac{4e \cdot i}{\ell \cdot N^{\frac{j-1}{2}}}\right)^{\ell}.$$

We now bound the amplitude $g_{i,k}$ with an inductive formula, as for the RSC problem.

▶ **Lemma 24.** *For all $i \in \mathbb{N}$ and $k \in \mathbb{N}$, we have that:*

$$g_{i,k} \leq g_{i-1,k} + 4\left(k^k \frac{i-1}{N^k}\right)^{1/2} + \left(\sum_{j=2}^{k}\sum_{\ell\geq 0} \frac{\ell}{N^{k+1-j}} \cdot \frac{k!}{(j-1)!} {f^{rep}_{i-1,\ell,j}}^2\right)^{1/2}.$$

We now bound $g_{i,k}$ in the following lemma.

▶ **Lemma 25.** *For every $i \in \mathbb{N}$ and $k \in \mathbb{N}$, we have that:*

$$g_{i,k} \leq 4k^{k/2} \cdot \frac{i^{3/2}}{N^{k/2}} + \sqrt{\sum_{j=2}^{k} \frac{k!}{(j-1)!}} \cdot \frac{4e \cdot i^{5/2}}{N^{k/2}}.$$

We can now prove Theorem 21.

Proof of Theorem 21. From Lemma 25, we have that:

$$g_{i,k} \leq 4k^{k/2} \cdot \frac{i^{3/2}}{N^{k/2}} + \sqrt{\sum_{j=2}^{k} \frac{k!}{(j-1)!}} \cdot \frac{4e \cdot i^{5/2}}{N^{k/2}}.$$

Writing $C_k = \sum_{j=2}^{k} \frac{k!}{(j-1)!}$, this rewrites as:

$$g_{i,k} \leq 4k^{k/2} \cdot \frac{i^{3/2}}{N^{k/2}} + \sqrt{C_k} \cdot \frac{4e \cdot i^{5/2}}{N^{k/2}}.$$

If $i = o\left(C_k^{-1/5} \cdot N^{k/5}\right)$, then $g_{i,k} = o(1)$. Hence if we want $g_{i,k}$ to be constant, i.e. not $o(1)$, we must have $i = \Omega\left(C_k^{-1/5} \cdot N^{k/5}\right)$. ◀

4.2 Algorithm for finding a $(1, k)$-subset cover

We now describe an algorithm that finds a $(1, k)$–SC, assuming $|\mathcal{X}| = |\mathcal{Y}|^k = N^k$. We first notice that an algorithm that finds a collision on H also finds a $(1, k)$–SC in an expected $O(N^{k/3})$ number of queries. We show now that there is a more efficient algorithm, as stated in the following theorem:

▶ **Theorem 26.** *There exists a quantum algorithm that finds a $(1, k)$–SC in expected $O\left(N^{k/4}\right)$ quantum queries if k is even, and $O(N^{k/4+1/12})$ if k is odd.*

To prove this theorem, we describe the following algorithm (which takes as parameters j and t, whose values will be chosen later):

▶ **Algorithm 27.** *Input: $j \in \{2, \ldots, k\}$ and $t \in \mathbb{N}$.*

1. *Define $F_1 : \mathcal{X} \to \{0, 1\}$ as follows:*

$$F_1(x) = \begin{cases} 1, & \text{if } h_1(x) = h_2(x) = \cdots = h_j(x) \\ 0, & \text{otherwise.} \end{cases}$$

   *(Note that an element $x \in \mathcal{X}$ such that $F_1(x) = 1$ is a j-*repetition*.)*
2. *Execute Grover's algorithm t times on F_1 to find t distinct j-repetitions in H. Let $T = \{x_1, \ldots, x_t\}$ be the set of these j-repetitions.*
3. *Define $F_2 : \mathcal{X} \to \{0, 1\}$ as follows:*

$$F_2(x) = \begin{cases} 1, & \text{if there exists } x_0 \in T \text{ such that } h_1(x) = h_1(x_0) \\ & \text{and for } 1 \leq m \leq k - j, h_{m+1}(x) = h_{j+m}(x_0) \\ 0, & \text{otherwise.} \end{cases}$$

4. *Execute Grover's algorithm to find an x such that $F_2(x) = 1$*
5. *Find x_0 in T corresponding to x, and output (x, x_0).*

▶ **Lemma 28.** *Algorithm 27 makes an expected number of $O\left(N^{(2k-j+1)/6}\right)$ queries to the oracle when $j \leq \frac{k+2}{2}$ for $t = N^{(k-2j+2)/3}$.*

We now prove Theorem 26

Proof of Theorem 26. From Lemma 28, the complexity of Algorithm 27 is $O(N^{(2k-j+1)/6})$ when $j \leq \frac{k+2}{2}$.

- If k is even, then we pick $j = \frac{k+2}{2}$ to reach a complexity of $O(N^{k/4})$.
- If k is odd, then we pick $j = \frac{k+1}{2}$ to reach a complexity of $O(N^{k/4+1/12})$.

Note that if $j > \frac{k+1}{2}$, then the second step of the algorithm is expected to make at least $O\left(N^{\frac{k+1}{4}}\right)$ quantum queries, which is worse than $O(N^{k/4+1/12})$. ◀

▶ Remark. Note that we do not reach the lower bound of Theorem 21, and it would be interesting to see if the gap can be further reduced by either improving our lower bounds or designing a more efficient algorithm.

4.3 Algorithm for finding a (r, k)-subset cover

In this section, we describe an algorithm for solving the (r, k)–SC problem. We consider the case where $|\mathcal{X}| = |r \cdot \mathcal{Y}|^k = r^k \cdot N^k$. The result is stated as follows:

▶ **Theorem 29.** *There exists a quantum algorithm that finds a (r, k)–SC in $O\left(N^{k/(2+2r)}\right)$ quantum queries to H, if k is divisible by $r+1$, and $O\left(N^{k/(2+2r)+1/2}\right)$ otherwise.*

The idea of the algorithm is essentially the same as Algorithm 27 of Section 4.2:

1. we first find t distinct $(r-1, k')$–SC for some integers t and k';
2. we then find the (r, k)–SC.

The first step is done recursively, using the algorithm defined for lower values of k' and $r-1$. The second step uses Grover's algorithm. The algorithm can be defined for any value of k' and t, and we pick them to optimize the complexity.

More formally, we define the algorithm recursively. Assume that we have an algorithm that can output a $(r-1, k')$–SC in $O\left(N^{k'/2r}\right)$ queries, for any $k' < k$ such that k' is divisible by r. Then, we can find a (r, k)–SC as follows:

▶ **Algorithm 30.** *Input: $t \in \mathbb{N}$, $k' \in \mathbb{N}$.*

1. *Execute the $(r-1, k')$–SC algorithm t times to find t distinct $(r-1, k')$–SC in H. Let $T = \{(x_{1,0}, x_{1,1}, \ldots, x_{1,r-1}), \ldots, (x_{t,0}, x_{t,1}, \ldots, x_{t,r-1})\}$ be the set of these $(r-1, k')$–SC.*
2. *Define $F : \mathcal{X} \to \{0, 1\}$ as follows:*

$$F(x) = \begin{cases} 1, & \text{if there exists } (x_{i,0}, x_{i,1}, \ldots, x_{i,r-1}) \in T \text{ such that} \\ & \forall 1 \leq m \leq k - k', h_m(x) = h_{k'+m}(x_{i,0}), \\ 0, & \text{otherwise.} \end{cases}$$

3. *Execute Grover's algorithm to find an x such that $F(x) = 1$*
4. *Find $(x_{i,0}, x_{i,1}, \ldots, x_{i,r-1})$ in T and output $(x_{i,0}, x_{i,1}, \ldots, x_{i,r-1}, x)$.*

▶ **Lemma 31.** *Algorithm 30 makes an expected number of $O\left(N^{k/(2+2r)}\right)$ queries to the oracle, when k is divisible by r, and $O\left(N^{k/(2+2r)+1/2}\right)$ otherwise.*

The proof of Theorem 29 follow directly from Lemma 31.

References

1 Jean-Philippe Aumasson and Guillaume Endignoux. Clarifying the subset-resilience problem. Cryptology ePrint Archive, Report 2017/909, 2017. URL: `https://eprint.iacr.org/2017/909`.

2 Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 368–397. Springer, Heidelberg, April 2015. `doi:10.1007/978-3-662-46800-5_15`.

3 Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS$^+$ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2129–2146. ACM Press, November 2019. `doi:10.1145/3319535.3363229`.

4 Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011. `doi:10.1007/978-3-642-25385-0_3`.

5 Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, June 1998. doi:10.1002/(sici)1521-3978(199806)46:4/5<493::aid-prop493>3.0.co;2-p.

6 Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN '98: Theoretical Informatics, Third Latin American Symposium, Campinas, Brazil, April, 20-24, 1998, Proceedings*, volume 1380 of *Lecture Notes in Computer Science*, pages 163–169. Springer, 1998. doi:10.1007/BFb0054319.

7 Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 598–629. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_21.

8 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM STOC*, pages 212–219. ACM Press, May 1996. doi:10.1145/237814.237866.

9 L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

10 Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 189–218. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_7.

11 Leonid Reyzin and Natan Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP 02*, volume 2384 of *LNCS*, pages 144–153. Springer, Heidelberg, July 2002. doi:10.1007/3-540-45450-0_11.

12 Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. doi:10.1137/s0097539795293172.

13 Takashi Yamakawa and Mark Zhandry. Classical vs quantum random oracles. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 568–597. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_20.

14 Quan Yuan, Mehdi Tibouchi, and Masayuki Abe. On subset-resilient hash function families. *Designs, Codes and Cryptography*, 90, March 2022. doi:10.1007/s10623-022-01008-4.

15 Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_9.