Improved Quantum Multicollision-Finding Algorithm

Akinori Hosoyamada[1], Yu Sasaki[1], Seiichiro Tani[2], and Keita Xagawa[1]

[1] NTT Secure Platform Laboratories,
3-9-11, Midori-cho Musashino-shi, Tokyo 180-8585, Japan.
[2] NTT Communication Science Laboratories,
3-1, Morinosato Wakamiya Atsugi-shi, Kanagawa 243-0198, Japan.
{hosoyamada.akinori,sasaki.yu,tani.seiichiro,xagawa.keita}@lab.ntt.co.jp

Abstract. The current paper improves the number of queries of the previous quantum multi-collision finding algorithms presented by Hosoyamada et al. at Asiacrypt 2017. Let an l-collision be a tuple of l distinct inputs that result in the same output of a target function. The previous algorithm finds l-collisions by recursively calling the algorithm for finding $(l-1)$-collisions, and it achieves the query complexity of $O(N^{(3^{l-1}-1)/(2\cdot 3^{l-1})})$. The new algorithm removes the redundancy of the previous recursive algorithm so that different recursive calls can share a part of computations. The new algorithm achieves the query complexity of $\tilde{O}(N^{(2^{l-1}-1)/(2^{l}-1)})$. Moreover, it finds multiclaws for random functions, which are harder to find than multicollisions.

Keywords post-quantum cryptography, quantum algorithm, multiclaw, multicollision

1 Introduction

Post-quantum cryptography has recently been discussed very actively in the cryptographic community. Quantum computers would completely break many classical public-key cryptosystems. In response, NIST is now conducting a standardization to select new public-key cryptosystems that resist attacks with quantum computers. Given this background, it is now important to investigate how quantum computers can impact on other cryptographic schemes including cryptographic hash functions.

A multicollision for a function f denotes multiple inputs to f such that they are mapped to the same output value. In particular, an l-collision denotes a tuple of l distinct inputs $x_1, x_2, \cdots, x_l$ such that $f(x_1) = f(x_2) = \cdots = f(x_l)$.

A multicollision is an important object in cryptography. Lower bounds on the complexity of finding a multicollision are sometimes used to give security bounds in provable security (e.g., security bounds for the schemes based on the sponge construction [JLM14]). In a similar context, the complexity of finding a multicollision directly impacts on the best cryptanalysis against some constructions. Furthermore, multicollisions can be used as a proof-of-work for blockchains. In digital payment schemes, a coin must be a bit-string the validity of which can be easily checked but which is hard to produce. A micro-payment scheme, MicroMint [RS96], defines coins as 4-collisions for a function. If 4-collisions can be produced quickly, a malicious user can counterfeit coins. Some recent works prove security of schemes and protocols based on the assumption that there exist functions for which it is hard to find multicollisions [BKP18,BDRV18,KNY18].

Hosoyamada et al. provided a survey of multicollision finding algorithms with quantum computers [HSX17]. They first showed that an l-collision can be produced with at most $O(N^{1/2})$ queries to the target function by iteratively applying the Grover search [Gro96,BBHT98] l times. They also reported that a combination of Zhandry's algorithm with $l = 3$ [Zha15] and Belovs' algorithm [Bel12] achieves $O(N^{10/21})$ for $l = 3$, which is faster than the simple application of Grover's algorithm. Finally, Hosoyamada *et al.* presented their own algorithm that recursively applies the collision finding algorithm by Brassard, Høyer, and Tapp [BHT98]. Their algorithm achieves the query complexity of $O(N^{(3^{l-1}-1)/(2\cdot 3^{l-1})})$. For $l = 3$ and $l = 4$, the complexities are $O(N^{4/9})$ and $O(N^{13/27})$, respectively, and the algorithm works as follows.

- To search for 3-collisions, it first iterates the $O(N^{1/3})$-query quantum algorithm for finding a 2-collision $O(N^{1/9})$ times. Then, it searches for the preimage of any one of the $O(N^{1/9})$ 2-collisions by using Grover's algorithm, which runs with $O(N^{4/9})$ queries.
- To search for 4-collisions, it iterates the $O(N^{4/9})$-query quantum algorithm for finding a 3-collision $O(N^{1/27})$ times. Then, it searches for the preimage of any one of the $O(N^{1/27})$ 3-collisions with $O(N^{13/27})$ queries.

As demonstrated above, the recursive algorithm by Hosoyamada *et al.* [HSX17] runs $(l-1)$-collision algorithm multiple times, but in each invocation, the algorithm starts from scratch. This fact motivates us to consider reusing the computations when we search for multiple $(l-1)$-collisions.

Our Contributions. In this paper, we improve the query complexity of the previous multicollision finding algorithm by removing the redundancy of the algorithm. The new algorithm achieves the query complexity of $\tilde{O}\left(N^{\frac{2^{l-1}-1}{2^l-1}}\right)$. The complexities for small l's are listed in Table 1. A comparison between complexities can be found in Figure 1. Our algorithm finds a 2-collision, 3-collision, 4-collision, and 5-collision of SHA3-512 with $2^{170.7}$, $2^{219.4}$, $2^{238.9}$, and $2^{247.7}$ quantum queries, respectively, up to a constant factor (Table 2).

Moreover, our new algorithm finds multiclaws for random functions, which are harder to find than multicollisions : An l-claw for functions $f_i\colon X_i \to Y$ for $1 \le i \le l$ is defined as a tuple $(x_1, \dots, x_l) \in X_1 \times \cdots \times X_l$ such that $f_i(x_i) = f_j(x_j)$ for all (i, j). Our quantum algorithm finds an l-claw with $\tilde{O}\left(N^{\frac{2^{l-1}-1}{2^l-1}}\right)$ quantum queries, if $|X_1|, \dots, |X_l|, |Y|$ are all in $O(N)$.

Table 1. Query complexities of l-collision finding quantum algorithms. Each fraction denotes the logarithm of the number of queries to the base N. The query complexity asymptotically approaches $1/2$ as l increases.

l	2	3	4	5	6	7	8
[HSX17] : $\frac{3^{l-1}-1}{2\cdot 3^{l-1}}$	$\frac{1}{3}$	$\frac{4}{9}$	$\frac{13}{27}$	$\frac{40}{81}$	$\frac{121}{243}$	$\frac{364}{729}$	$\frac{1093}{2187}$
Ours : $\frac{2^{l-1}-1}{2^{l}-1}$	$\frac{1}{3}$	$\frac{3}{7}$	$\frac{7}{15}$	$\frac{15}{31}$	$\frac{31}{63}$	$\frac{63}{127}$	$\frac{127}{255}$

l	2	3	4	5	6	7	8
[HSX17] : $\frac{3^{l-1}-1}{2\cdot 3^{l-1}}$	0.3333..	0.4444..	0.4814..	0.4938..	0.4979..	0.4993..	0.4997..
Ours : $\frac{2^{l-1}-1}{2^{l}-1}$	0.3333..	0.4285..	0.4666..	0.4838..	0.4920..	0.4960..	0.4980..

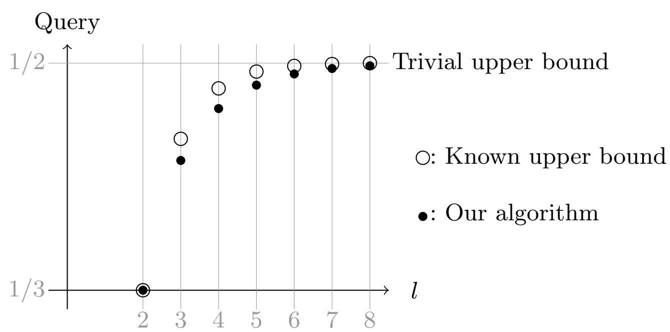

Fig. 1. Quantum query complexity for finding an l-collision. "Query" denotes the logarithm of the number of queries to the base N.

Table 2. The number of queries required to find an l-collision of SHA3-512.

l	2	3	4	5
[HSX17]	$2^{170.7}$	$2^{227.6}$	$2^{246.5}$	$2^{252.8}$
Ours	$2^{170.7}$	$2^{219.4}$	$2^{238.9}$	$2^{247.7}$

Paper Outline. The remaining of this paper is organized as follows. In section 2, we describe notations, definitions and settings. In section 3, we review previous works related to the multicollision-finding problem. In section 4, we give our new quantum algorithm and its complexity analysis. In section 5, we conclude this paper.

Concurrent Work. Very recently, Liu and Zhandry [LZ18] showed that for any constant l, $\Theta\left(N^{\frac{1}{2}(1-\frac{1}{2^l-1})}\right)$ quantum queries are both necessary and sufficient to find a l-collision with constant probability, for a random function. That is, they gave an improved upperbound and a new lowerbound on the average case. The comparisons are summarized as follows:

– Liu and Zhandry consider the l-collision case that $|X| \geq l|Y|$, where X is the domain and Y is the range. We treat the case that $|X| \geq \frac{l}{c}|Y|$ for any integer constant c. We also consider the *multiclaw* case.
– Their exponent $\frac{1}{2}(1-\frac{1}{2^l-1})$ is the same as ours $\frac{2^{l-1}-1}{2^l-1}$.
– They give the upperbound $O\left(N^{\frac{1}{2}(1-\frac{1}{2^l-1})}\right)$, while we give $\tilde{O}\left(N^{\frac{1}{2}(1-\frac{1}{2^l-1})}\right)$. We pay $(\ln N)^2$ factor to allow smaller domains.
– They give a lowerbound, which matches with their upperbound.

We finally note that our result on an improved l-collision finding algorithm for the case $|X| \geq l|Y|$ with query complexity $O\left(N^{\frac{1}{2}(1-\frac{1}{2^l-1})}\right)$ is reported in the Rump Session of Asiacrypt 2017.

2 Preliminaries

For a positive integer M, let $[M]$ denote the set $\{1,\ldots,M\}$. In this paper, N denotes a positive integer. We assume that l is a small constant. We focus on reducing quantum *query* complexities for finding multicollisions and multiclaws. We do not consider other complexity notions such as time complexity and the size of quantum circuits. Unless otherwise noted, all sets are non-empty and finite. For sets X and Y, $\mathsf{Func}(X,Y)$ denotes the set of functions from X to Y. For each $f \in \mathsf{Func}(X,Y)$, we denote the set $\{f(x) \mid x \in X\}$ by $\mathrm{Im}(f)$. For a set X, let $U(X)$ denote the uniform distribution over X. For a distribution $\mathcal{D}$ on a set X, let $x \sim \mathcal{D}$ denote that x is a random variable that takes a value drawn from X according to $\mathcal{D}$.

An *l-collision* for a function $f\colon X \to Y$ is a tuple of elements $(x_1,\ldots,x_l,y)$ in $X^\ell \times Y$ such that $f(x_i) = f(x_j) = y$ and $x_i \neq x_j$ for all $1 \leq i \neq j \leq l$. An l-collision is simply called a *collision* for $l = 2$, and called a *multicollision* for $l \geq 3$. Moreover, an *l-claw* for functions $f_i : X_i \to Y$ for $1 \leq i \leq l$ is a tuple $(x_1,\ldots,x_l,y) \in X_1 \times \cdots \times X_l \times Y$ such that $f_1(x_1) = \cdots = f_l(x_l) = y$. An l-claw is simply called a *claw* for $l = 2$, and called a *multiclaw* for $l \geq 3$.

The problems of finding multicollisions or multiclaws are often studied in the contexts of both cryptography and quantum computation, but the problem settings of interest change depending on the contexts. In the context of quantum computation, most problems are studied in the *worst case*, and an algorithm is said to (efficiently) solve a problem only when it does (efficiently) for all functions. On the other hand, most problems in cryptography are studied in the *average case*, since randomness is one of the most crucial notion in cryptography. In particular, we say that an algorithm (efficiently) solves a problem if it (efficiently) solves the problem with a high probability on average over randomly chosen functions.

This paper focuses on the settings of interest in the context of cryptography. Formally, our goal is to solve the following two problems.

Problem 1 (Multicollision-finding problem, average case). Let l be a positive integer constant, and X, Y denote non-emtpy finite sets. Suppose that a function $F\colon X \to Y$ is chosen uniformly at random and given as quantum oracles. Then, find an l-collision for F.

Problem 2 (Multiclaw-finding problem, average case). Let l be a positive integer constant, and $X_1, \ldots, X_l, Y$ denote non-emtpy finite sets. Suppose that functions $f_i\colon X_i \to Y (1 \leq i \leq l)$ are chosen independently and uniformly at random, and given as quantum oracles. Then, find an l-claw for $f_1, \ldots, f_l$.

Roughly speaking, Problem 1 is easier to solve than Problem 2. Suppose that $F\colon X \to Y$ is a function, and we want to find an l-collision for F. Let $X_1, \ldots, X_l$ be subsets of X such that $X_i \cap X_j = \emptyset$ for $i \neq j$ and $\bigcup_i X_i = X$. If $(x_1, \ldots, x_l, y)$ is an l-claw for $F|_{X_1}, \ldots, F|_{X_l}$, then it is obviously an l-collision for F. In general, an algorithm for finding an l-claw can be converted into one for finding l-collisions. To be precise, the following lemma holds.

Lemma 1. *Let X, Y be non-empty finite sets, and $X_1, \ldots, X_l$ be subsets of X such that $X_i \cap X_j = \emptyset$ for $i \neq j$ and $\bigcup_i X_i = X$. If there exists a quantum algorithm $\mathcal{A}$ that solves Problem 2 for the sets $X_1, \ldots, X_l, Y$ by making at most q quantum queries with probability at least p, then there exists a quantum algorithm $\mathcal{B}$ that solves Problem 1 for the sets X, Y by making at most q quantum queries with probability at least p.*

How to measure the size of a problem also changes depending on which context we are in. In the context of cryptography, the problem size is often regarded as the size of the range of functions in the problem rather than the size of the domains, since the domains of cryptographic functions such as hash functions are much larger than their ranges. Hence, we regard the range size $|Y|$ as the size of Problem 1 (and Problem 2) when we analyze the complexity of quantum algorithms.

In the context of quantum computation, there exist previous works on problems related to ours [Bel12,Amb04,Tan09,BDH$^+$01] (element distinctness problem, for example), but those works usually focus on the worst case complexity and regard the domain sizes of functions as the problem size. In particular, there does not exist any previous work that studies multiclaw-finding problem for general l in the average case, to the best of authors' knowledge.

3 Previous Works

3.1 The Grover Search and Its Generalization

As a main tool for developing quantum algorithms, we use the quantum database search algorithm that was originally developed by Grover [Gro96] and later generalized by Boyer, Brassard, Høyer, and Tapp [BBHT98]. Below we introduce the generalized version.

Theorem 1. *Let X be a non-empty finite set and $f\colon X \to \{0,1\}$ be a function such that $t/|X| < 17/81$, where $t = |f^{-1}(1)|$. Then, there exists a quantum algorithm* BBHT *that finds x such that $f(x) = 1$ with an expected number of quantum queries to f at most*

$$\frac{4|X|}{\sqrt{(|X|-t)t}} \leq \frac{9}{2} \cdot \sqrt{\frac{|X|}{t}}.$$

If $f^{-1}(1) = \emptyset$, then BBHT *runs forever.*

Theorem 1 implies that we can find l-collisions and l-claws for random functions with $O(\sqrt{N})$ quantum queries, if the sizes of range(s) and domain(s) of function(s) are $\Theta(N)$: Suppose that we are given functions $f_i\colon X_i \to Y$ for $1 \leq i \leq l$, where $|X_1|, \ldots, |X_l|$, and $|Y|$ are all in $\Theta(N)$, and we want to find an l-claw for those functions. Take an element $y \in Y$ randomly, and define $F_i\colon X_i \to \{0,1\}$ for each i by $F_i(x) = 1$ if and only if $f_i(x) = y$. Then, effectively, by applying BBHT to each F_i, we can find $x_i \in X_i$ such that $f_i(x_i) = y$ for each i with $O(\sqrt{N})$ quantum queries. Similarly we can find an l-collision for a random function $F\colon [N] \to [N]$ with $O(\sqrt{N})$ quantum queries. In particular, $O(\sqrt{N})$ is a trivial upper bound of Problem 1 and Problem 2.

3.2 The BHT Algorithm

Brassard, Høyer, and Tapp [BHT98] developed a quantum algorithm that finds 2-claws (below we call it BHT).[3] BHT finds a claw for two one-to-one functions $f_1\colon X_1 \to Y$ and $f_2\colon X_2 \to Y$ as sketched in the following. For simplicity, here we assume $|X_1| = |X_2| = |Y| = N$. Under this setting, BHT finds a 2-claw with $O(N^{1/3})$ quantum queries.

[3] As in our case, the BHT algorithm also focus on only quantum query complexity. Although it runs in time $\tilde{O}(N^{1/3})$ on an idealized quantum computer, it requires $\tilde{O}(N^{1/3})$ qubits to store data in quantum memories. Recently Chailloux et al. [CNS17] has developed a quantum 2-collision finding algorithm that runs in time $\tilde{O}(N^{2/5})$, which is polynomially slower than the BHT algorithm but requires only $O(\log N)$ quantum memories.

Rough Sketch of BHT :

1. **Construction of a list L.** Take a subset $S \subset X_1$ of size $N^{1/3}$ arbitrarily. For each $x \in S$, compute the value $f_1(x)$ by making a query and store the pair $(x, f(x))$ in a list L.
2. **Extension to a claw.** Define a function $F_L\colon X_2 \to \{0,1\}$ by $F_L(x') = 1$ if and only if the value $f_2(x') \in Y$ appears in the list L (i.e., there exists $x_1 \in S$ such that $f_2(x') = f_1(x_1)$). Apply BBHT to F_L and find $x_2 \in X_2$ such that $f_2(x_2)$ appears in L.
3. **Finalization.** Find $(x_1, f_1(x_1)) \in L$ such that $f_1(x_1) = f_2(x_2)$, and then output (x_1, x_2).

Quantum query complexity. BHT finds a claw with $O(N^{1/3})$ quantum queries. In the first step, the list L is constructed by making $O(N^{1/3})$ quantum queries to f_1. In the second step, since $|F_L^{-1}(1)| = |f_2^{-1}(f_1(S))|$ is equal to $N^{1/3}$, BBHT finds x_2 with $O(\sqrt{N/N^{1/3}}) = O(N^{1/3})$ quantum queries to f_2 (note that we can evaluate F_L by making one query to f_2). The third step does not require queries. Therefore BHT finds a collision by making $O(N^{1/3})$ quantum queries in total.

Extension to a collision-finding algorithm. It is not difficult to show that BHT works for random functions. Thus, BHT can be extended to the quantum collision-finding algorithm as mentioned in section 2. Suppose we want to find a (2-)collision for a random function $F\colon X \to Y$. Here we assume $|X| = 2N$ and $|Y| = N$ for simplicity.

Now, choose a subset $X_1 \subset X$ of size N arbitrarily and let $X_2 := X \setminus X_1$. Then we can find a collision for F by applying the BHT algorithm introduced above to the functions $F|_{X_1}$ and $F|_{X_2}$, since a claw for them becomes a collision for F.

3.3 The HSX Algorithm

Next, we introduce a quantum algorithm to find multicollisions that was developed by Hosoyamada, Sasaki, and Xagawa [HSX17] (the algorithm is designed to find only multicollisions, and cannot find multiclaws). Below we call their algorithm HSX.

The main idea of HSX is to apply the strategy of BHT recursively: To find an l-collision, HSX calls itself recursively to find many $(l-1)$-collisions, and then extend any one of those $(l-1)$-collisions to an l-collision by applying BBHT.

Rough Sketch of HSX : In what follows, N denotes $|Y|$. Let us denote HSX(l) by the HSX algorithm for finding l-collisions. HSX(l) finds an l-collision for a random function $f\colon X \to Y$ with $|X| \geq l \cdot |Y|$ as follows.

Recursive call to construct a list L_{l-1}. Apply HSX($l-1$) to f $N^{1/3^{l-1}}$ times to obtain $N^{1/3^{l-1}}$ many $(l-1)$-collisions. Store those $(l-1)$-collisions in a list L_{l-1}.

Extension to an l-collision. Define $F_{l-1}\colon X \to \{0,1\}$ by $F_{l-1}(x') = 1$ if and only if there exists an $(l-1)$-collision $(x_1, \ldots, x_{l-1}, y) \in L_{l-1}$ such that $(x_1, \ldots, x_{l-1}, x', y)$ forms an l-collision for f, i.e., $f(x') = y$ and $x' \neq x_i$ for $1 \leq i \leq l-1$. Apply BBHT to F_{l-1} to find $x_l \in X$ such that $F_{l-1}(x_l) = 1$.

Finalization. Find $(x_1, \ldots, x_{l-1}, y) \in L_{l-1}$ such that $F_{l-1}(x_l) = y$. Output $(x_1, \ldots, x_{l-1}, x_l, y)$.

Quantum query complexity. HSX finds a l-collision with $O(N^{(3^{l-1}-1)/2\cdot 3^{l-1}})$ quantum queries on average, which can be shown by induction as follows. For 2-collisions, HSX(2) matches the BHT algorithm. For general $l \geq 3$, suppose that HSX$(l-1)$ finds an $(l-1)$-collision with $O(N^{(3^{l-2}-1)/2\cdot 3^{l-2}})$ quantum queries on average. In its first step, HSX(l) makes $N^{1/3^{l-1}} \cdot O(N^{(3^{l-2}-1)/2\cdot 3^{l-2}}) = O(N^{(3^{l-1}-1)/2\cdot 3^{l-1}})$ quantum queries. Moreover, in its second step, HSX(l) makes $O(\sqrt{N/N^{(3^{l-2}-1)/2\cdot 3^{l-2}}}) = O(N^{(3^{l-1}-1)/2\cdot 3^{l-1}})$ quantum queries by using BBHT. The third step does not make quantum queries. Therefore it follows that HSX(l) makes $O(N^{(3^{l-1}-1)/2\cdot 3^{l-1}})$ quantum queries in total.

4 New Quantum Algorithm Mclaw

This section gives our new quantum algorithm Mclaw that finds an l-claw for random functions $f_i\colon X_i \to Y$ for $1 \leq i \leq l$, where $|Y| = N$ and there exists a constant $c \geq 1$ such that $\frac{N}{c} \leq |X_i|$ for all i, with $\tilde{O}(N^{(2^{l-1}-1)/(2^l-1)})$ quantum queries. Roughly speaking, this means that, an l-collision for a random function $f\colon X \to Y$, where $|X|$ and $|Y|$ are in $O(N)$, can be found with $\tilde{O}(N^{(2^{l-1}-1)/(2^l-1)})$ quantum queries, which improves the previous result [HSX17] (see section 2).

Our algorithm assumes that $|X_1|, \ldots, |X_l|$, and $|Y|$ are all in $O(N)$. However, it can also be applied to the functions of interest in the context of cryptography, i.e., the functions of which domains are much larger than ranges, by restricting the domains of them to suitable subsets.

The main idea of our new algorithm is to improve HSX by getting rid of its redundancy: To find an l-collision, HSX recursively calls itself to find many $(l-1)$-collisions. Once HSX finds an $(l-1)$-collision $\gamma = (x_1, \ldots, x_{l-1}, y)$, it stores γ in a list L_{l-1}, *discards all the data that was used to find* γ, and then start to search for another $(l-1)$-collision γ'. It is inefficient to discard data every time an $(l-1)$-collision is found, and our new algorithm Mclaw reduces the number of quantum queries by reusing those data. Moreover, HSX cannot solve multiclaw-finding problem while the BHT algorithm can also solve claw-finding problem. Our algorithm Mclaw can solve both of two problems.

We begin with describing intuition of our algorithm, and then give its formal description.

4.1 Intuitive Description and Complexity Analysis

We explain the idea of how to develop the BHT algorithm, how to develop a quantum algorithm to find 3-claws from BHT, and how to extend it further to the case of finding an l-claw for any l.

How to develop the BHT algorithm. Here we review how the BHT algorithm is developed. Let $f_1\colon X_1 \to Y$ and $f_2\colon X_2 \to Y$ be one-to-one functions. The goal of the BHT algorithm is to find a (2-)claw for f_1 and f_2 with $O(N^{1/3})$ quantum queries. For simplicity, below we assume that $|X_1| = |X_2| = |Y| = N$ holds. Let t_1 be a parameter that defines the size of a list of 1-claws for f_1. It will be set as $t_1 = N^{1/3}$.

First, collect t_1 many 1-claws for f_1 and store them in a list L_1. This first step makes t_1 queries. Second, extend 1-claws in L_1 to a 2-claw for f_1 and f_2, by using BBHT, and output the obtained 2-claw. Since BBHT makes $O(\sqrt{N/t_1})$ queries to make a 2-claw from L_1, this second step makes $O(\sqrt{N/t_1})$ queries (see Theorem 1). Overall, the above algorithm makes $q_2(t_1) = t_1 + \sqrt{N/t_1}$ quantum queries up to a constant factor.

The function $q_2(t_1)$ takes its minimum value $2 \cdot N^{1/3}$ when $t_1 = N^{1/3}$. The BHT algorithm is developed in this way, by setting $t_1 = N^{1/3}$.

From BHT to a 3-claw-finding algorithm. Next, we show how the above strategy to develop the BHT algorithm can be extended to develop a 3-claw-finding algorithm. Let $f_i\colon X_i \to Y$ be one-to-one functions for $1 \leq i \leq 3$. Our goal here is to find a 3-claw for f_1, f_2, and f_3 with $O(N^{3/7})$ quantum queries. For simplicity, below we assume $|X_1| = |X_2| = |X_3| = |Y| = N$. Let t_1, t_2 be parameters that define the number of 1-claws for f_1 and that of 2-claws for f_1 and f_2, respectively. (They will be fixed later.)

First, collect t_1 many 1-claws for f_1 and store them in a list L_1. This first step makes t_1 queries. Second, extend 1-claws in L_1 to t_2 many 2-claws for f_1 and f_2, by using BBHT, and store them in a list L_2. Here we do not discard the list L_1 until we construct the list L_2 of size t_2, while the HSX algorithm does. Since BBHT makes $O(\sqrt{N/t_1})$ queries to make a 2-claw from L_1, this second step makes $t_2 \cdot O(\sqrt{N/t_1})$ queries (see Theorem 1). Finally, extend 2-claws in L_2 to a 3-claw for f_1, f_2, and f_3, by using BBHT, and output the obtained 3-claw. This final step makes $O(\sqrt{N/t_2})$ queries. Overall, the above algorithm makes $q_3(t_1, t_2) = t_1 + t_2 \cdot \sqrt{N/t_1} + \sqrt{N/t_2}$ quantum queries up to a constant factor.

The function $q_3(t_1, t_2)$ takes its minimum value $3 \cdot N^{3/7}$ when $t_1 = t_2 \cdot \sqrt{N/t_1} = \sqrt{N/t_2}$, which is equivalent to $t_1 = N^{3/7}$ and $t_2 = N^{1/7}$. We can develop a 3-claw finding algorithm with $O(N^{3/7})$ quantum queries in this way, by setting $t_1 = N^{3/7}$ and $t_2 = N^{1/7}$.

***l*-claw-finding algorithm for general *l*.** Generalizing the above idea to find 3-claws, we can find l-claws for general l as follows. Let $f_i : X_i \to Y$ be one-to-one functions for $1 \leq i \leq l$. Our goal here is to find an l-claw for $f_1, \ldots, f_l$.

For simplicity, below we assume that $|X_1| = \cdots = |X_l| = |Y| = N$ holds. Let $t_1, \ldots, t_{l-1}$ be parameters.

First, collect t_1 many 1-claws for f_1 and store them in a list L_1. This first step makes t_1 queries. In the i-th step for $2 \leq i \leq l-1$, extend $(i-1)$-claws in L_{i-1} to t_i many i-claws for $f_1, \ldots, f_i$, by using BBHT, and store them in a list L_i. Here we do not discard the list L_{i-1} until we construct the list L_i of size t_i. Since BBHT makes $O(\sqrt{N/t_{i-1}})$ queries to make an i-claw from L_{i-1}, the i-th step makes $t_i \cdot O(\sqrt{N/t_{i-1}})$ queries. Finally, extend $(l-1)$-claws in L_{l-1} to an l-claw for $f_1, \ldots, f_l$, by using BBHT, and output the obtained l-claw. This final step makes $O(\sqrt{N/t_{l-1}})$ queries. Overall, this algorithm makes $q_l(t_1, \ldots, t_{l-1}) = t_1 + t_2 \cdot \sqrt{N/t_1} + \cdots + t_{l-1} \cdot \sqrt{N/t_{l-2}} + \sqrt{N/t_{l-1}}$ quantum queries up to a constant factor. The function $q_l(t_1, \ldots, t_{l-1})$ takes its minimum value $l \cdot N^{(2^{l-1}-1)/(2^l-1)}$ when $t_1 = t_2 \cdot \sqrt{N/t_1} = \cdots = t_{l-1} \cdot \sqrt{N/t_{l-2}} = \sqrt{N/t_{l-1}}$, which is equivalent to $t_i = N^{(2^{l-i}-1)/(2^l-1)}$. Hence we can find an l-claw with $O(N^{(2^{l-1}-1)/(2^l-1)})$ quantum queries, by setting $t_i = N^{(2^{l-i}-1)/(2^l-1)}$. Our new quantum algorithm Mclaw is developed based on the above strategy for random functions.

4.2 Formal Description

Next, we formally describe our quantum multiclaw-finding algorithm Mclaw. A formal complexity analysis of Mclaw is given in the next subsection, and this subsection only describes how the algorithm works.

Let N be a sufficiently large integer and suppose that $|Y| = N$ holds. Below we assume that $|X_i| \leq |Y|$ holds for all i. This is a reasonable assumption since, if there is an algorithm that solves Problem 2 in the case that $|X_i| \leq |Y|$ holds for all i, then we can also solve the problem in other cases: If $|X_i| > |Y|$ holds for some i, take a subset $S_i \subset X_i$ such that $|S_i| = |Y|$ and find an l-claw for $f_1, \ldots, f_{i-1}, f_i|_{S_i}, f_{i+1}, \ldots, f_l$. Then the l-claw is also an l-claw for $f_1, \ldots, f_l$.

For each fixed $f_i \colon X_i \to Y$ and a list $L \subset Y$, define $F_i^L \colon X_i \to \{0,1\}$ by $F_i^L(x) = 1$ if and only if $f_i(x) \in L$. Our algorithm is parametrized by a positive integer k, and we denote the algorithm for the parameter k by Mclaw_k. We impose an upper limit on the number of queries that Mclaw_k is allowed to make : We design Mclaw_k in such a way that it immediately stops and aborts if the number of queries reaches the limit specified by the parameter $\mathsf{Qlimit}_k := k \cdot 30l\sqrt{c} \cdot N^{\frac{2^{l-1}-1}{2^l-1}} (\ln N)^2$. The upper limit Qlimit_k is necessary to prevent the algorithm from running forever, and to make the expected value of the number of queries converge. We also define the parameters controlling the sizes of the lists:

$$N_i := \begin{cases} \frac{N}{9(\ln N)^2} & (i = 0), \\ N^{\frac{2^{l-i}-1}{2^l-1}} & (i \geq 1). \end{cases} \tag{1}$$

For ease of notation, we define L_0 and L'_0. We let $L_0 = L'_0$ be an arbitrary subset of Y of cardinality $2N_0 \cdot \ln N$ $(= 2N/(9 \ln N))$. Then, Mclaw_k is described as in Algorithm 1.

Algorithm 1 Mclaw_k

Input: Randomly chosen functions $f_1, \ldots, f_l$ ($f_i \colon X_i \to Y$ and $|X_i| \leq |Y|$)).
Output: An l-multiclaw for $f_1, \ldots, f_l$ or $\perp$.
Stop condition: If the number of queries reaches Qlimit_k, stop and output $\perp$.

  $L_1, \ldots, L_l \leftarrow \emptyset$, $L'_1, \ldots, L'_l \leftarrow \emptyset$.
  for $i = 1$ to l **do**
    for $j = 1$ to $2N_i \cdot \ln N$ **do**
      if $i = 1$ **then**
        Take $x_j \in X_1$ that does not appear in L_1, $y \leftarrow f_1(x_j)$. //1 query is made
      else
        Find $x_j \in X_i$ whose image $y := f_i(x_j)$ is in L'_{i-1} by running BBHT on the boolean function $F_i^{L'_{i-1}}$.   //multiple queries are made
      end if
      $L_i \leftarrow L_i \cup \{(x^{(1)}, \ldots, x^{(i-1)}, x_j, y)\}$, $L'_i \leftarrow L'_i \cup \{y\}$.
      $L_{i-1} \leftarrow L_{i-1} \setminus \{(x^{(1)}, \ldots, x^{(i-1)}, y)\}$, $L'_i \leftarrow L'_{i-1} \setminus \{y\}$.
    end for
  end for
  Return an element $(x^{(1)}, \ldots, x^{(l)}; y) \in L_l$ as an output.

4.3 Formal Complexity Analysis

This section gives a formal complexity analysis of Mclaw. The goal of this section is to show the following theorem.

Theorem 2. *Assume that there exists a positive integer constant c such that $|X_i| \geq \frac{1}{c}|Y|$ holds for each i. If $|Y| = N$ is sufficiently large,* Mclaw_k *finds an l-claw with a probability at least*

$$1 - \left(\frac{1}{k} + l \cdot \exp\left(-\frac{N^{\frac{1}{2^l-1}}}{16c^2} \right) + \frac{l}{N} \right), \tag{2}$$

by making at most

$$\mathsf{Qlimit}_k = k \cdot 30l\sqrt{c} \cdot N^{\frac{2^{l-1}-1}{2^l-1}} (\ln N)^2 \tag{3}$$

queries.

This theorem shows that, for each small integer $k \geq 2$, Mclaw_k finds an l-claw with an overwhelming probability by making $O\left(N^{\frac{2^{l-1}-1}{2^l-1}} (\ln N)^2 \right)$ queries.

Proof (of Theorem 2). We show that Equation 2 holds. Let us define $\mathsf{good}^{(i)}$ to be the event that

$$|\mathrm{Im}(f_i) \cap L'_{i-1}| \geq \frac{N_{i-1}}{c} \wedge \frac{17|X_i|}{81} \geq |f_i^{-1}(L'_{i-1})| \tag{4}$$

holds just before Mclaw_k starts to construct i-multiclaws. (Intuitively, under the condition that $\mathsf{good}^{(i)}$ occurs, the number of queries does not become too large.) We show the following claim.

Claim. For sufficiently large N,

$$\Pr\left[\mathsf{good}^{(i)}\right] \geq 1 - \exp\left(-\frac{1}{16c^2} N_{i-1} \ln N\right) - \frac{1}{N}. \tag{5}$$

holds.

Proof. In this proof we consider the situation that Mclaw_k has finished to make L_{i-1} and before starting to make i-claws. In particular, we assume that $|L_{i-1}| = |L'_{i-1}| = 2N_{i-1} \ln N$.

Let us define a random variable

$$W := \{x \mid f_i(x) \in L'_{i-1}\}. \tag{6}$$

In addition, let $\mathsf{pregood}^{(i)}$ be the event that

$$|W| > \frac{3}{2c} N_{i-1} \ln N \tag{7}$$

holds. Now we use the following lemma as a fact.

Lemma 2 (Chernoff's bound [Sho09, Theorem 8.24]). *Let $0 \leq p \leq 1$ be a constant. Let $Z_1, \ldots, Z_s$ be random variables that take values in $\{0,1\}$ such that $\Pr[Z_i = 1] = p$ for any i, and $\bar{Z}$ be the random variable defined by $\bar{Z} := \sum_i Z_i$. Then, $\Pr[\bar{Z} \leq sp - \delta] \leq \exp(-\delta^2/2sp)$ holds for $0 \leq \delta \leq sp$.*

Apply Chernoff's bound above with $s = |X_i|$, $p = |L'_{i-1}|/N$, $\delta = \frac{1}{2c} \cdot N_{i-1} \ln N$, and $\{Z_x\}_{x \in X_i}$. Here, Z_x is the random variable such that $Z_x = 1$ if $f_i(x) \in L'_{i-1}$ and $Z_x = 0$ otherwise, for each x. Then $\bar{Z} := \sum_x Z_x = |W|$ holds and we have

$$\Pr_{f_i \sim U(\mathsf{Func}(X_i, Y))}\left[|W| \leq sp - \delta\right] = \Pr\left[\bar{Z} \leq sp - \delta\right] \leq \exp(-\delta^2/2sp). \tag{8}$$

In addition, we have that $\frac{3}{2c} N_{i-1} \ln N = \frac{N}{c} \cdot p - \delta \leq sp - \delta$ since $\frac{N}{c} \leq |X_i| = s$. Thus it follows that

$$\begin{aligned} \Pr\left[\neg\mathsf{pregood}^{(i)}\right] &= \Pr_{f_i \sim U(\mathsf{Func}(X_i, Y))}\left[|W| \leq \frac{3}{2c} N_{i-1} \ln N\right] \\ &\leq \Pr_{f_i \sim U(\mathsf{Func}(X_i, Y))}\left[|W| \leq sp - \delta\right] \leq \exp(-\delta^2/2sp) \end{aligned} \tag{9}$$

holds. Moreover, we have that

$$\begin{aligned} \exp\left(-\delta^2/2sp\right) &\leq \exp\left(-\frac{1}{2} \cdot \left(\frac{1}{2c} \cdot N_{i-1} \ln N\right)^2 \cdot \frac{1}{2N_{i-1} \ln N}\right) \\ &= \exp\left(-\frac{1}{16c^2} \cdot N_{i-1} \ln N\right), \end{aligned} \tag{10}$$

which implies that

$$\Pr\left[\neg\mathsf{pregood}^{(i)}\right] \leq \exp\left(-\frac{1}{16c^2} N_{i-1} \ln N\right) \tag{11}$$

holds.

Let $\mathsf{reg}^{(i)}$ be the event that

$$|f_i^{-1}(y)| \leq \frac{3\ln N}{\ln\ln N} \tag{12}$$

holds for all $y \in Y$. Then it follows that

$$\Pr\left[\neg\mathsf{reg}^{(i)}\right] \leq 1/N \tag{13}$$

holds for sufficiently large N, from the standard ball-into-bins arguments (see Lemma 5.1 of [MU17], for example).

Note that the event $\mathsf{good}^{(i)}$ always occurs if both of the events $\mathsf{pregood}^{(i)}$ and $\mathsf{reg}^{(i)}$ occurs: Since

$$|\mathrm{Im}(f_i) \cap L'_{i-1}| \geq \frac{|f_i^{-1}(L'_{i-1})|}{\max_y |f_i^{-1}(y)|} = \frac{|W|}{\max_y |f_i^{-1}(y)|} \tag{14}$$

holds, $|\mathrm{Im}(f_i) \cap L'_{i-1}|$ is lower bounded by

$$\frac{\frac{3}{2c} N_{i-1}\ln N}{\frac{3\ln N}{\ln\ln N}} \geq \frac{N_{i-1}}{c} \tag{15}$$

for sufficiently large N, if $\mathsf{pregood}^{(i)}$ and $\mathsf{reg}^{(i)}$ occurs. In addition, since

$$|f_i^{-1}(L'_{i-1})| \leq |L'_{i-1}| \cdot \max_y |f_i^{-1}(y)| \tag{16}$$

holds, $|f_i^{-1}(L'_{i-1})|$ is upper bounded as

$$\begin{aligned} 2N_{i-1}\ln N \cdot \frac{3\ln N}{\ln\ln N} &\leq \begin{cases} \frac{2}{3} \cdot \frac{N}{\ln\ln N} & (i=1) \\ \frac{6(\ln N)^2}{\ln\ln N} \cdot N^{\frac{2^{l-i+1}-1}{2^l-1}} & (i \geq 2) \end{cases} \\ &< \frac{17}{81}|X_i| \end{aligned} \tag{17}$$

for sufficiently large N, if $\mathsf{reg}^{(i)}$ occurs. Thus $\mathsf{good}^{(i)}$ always occurs if both of the events $\mathsf{pregood}^{(i)}$ and $\mathsf{reg}^{(i)}$ occurs.

Now we have

$$\begin{aligned} \Pr\left[\mathsf{good}^{(i)}\right] &\geq \Pr\left[\mathsf{good}^{(i)} \;\middle|\; \mathsf{pregood}^{(i)} \wedge \mathsf{reg}^{(i)}\right] \cdot \Pr\left[\mathsf{pregood}^{(i)} \wedge \mathsf{reg}^{(i)}\right] \\ &\geq 1 \cdot \left(1 - \Pr\left[\neg\mathsf{pregood}^{(i)}\right] - \Pr\left[\neg\mathsf{reg}^{(i)}\right]\right) \\ &\geq 1 - \exp\left(-\frac{1}{16c^2}N_{i-1}\ln N\right) - \frac{1}{N}, \end{aligned} \tag{18}$$

which completes the proof. □

Let good denote the event $\mathsf{good}^{(1)} \wedge \cdots \wedge \mathsf{good}^{(l)}$. Then we can show the following claim.

Claim. For sufficiently large N, it holds that

$$\mathbf{E}\left[Q \mid \mathsf{good}\right] \leq \frac{1}{k}\mathsf{Qlimit}_k, \tag{19}$$

where Q is the total number of queries made by Mclaw_k.

Proof. Let us fix i and j. Let $Q_j^{(i)}$ denote the number of queries made by Mclaw_k in the j-th search to construct i-multiclaws, and $Q^{(i)}$ denote $\sum_j Q_j^{(i)}$. In the j-th search to construct i-multiclaw, we search x from X_i, where there exist at least $N_{i-1}/c - j + 1$ answers under the condition that $\mathsf{good}^{(i)}$ occurs. If the number of answers $t = |f_i^{-1}(L'_{i-1})|$ is upper bounded by $17|X_i|/81$, the expected value of the number of queries made by BBHT in the j-th search to construct i-multiclaws is upper bounded by

$$\frac{4|X_i|}{\sqrt{(|X_i| - t)t}} \leq \frac{9}{2}\sqrt{|X_i|/t} \leq \frac{9}{2}\sqrt{N/t}. \tag{20}$$

Since

$$N_{i-1}/c - j + 1 \leq t = |f_i^{-1}(L'_{i-1})| \leq \frac{17}{81}|X_i| \tag{21}$$

holds in the j-th search to construct i-multiclaws under the condition that $\mathsf{good}^{(i)}$ occurs, it follows that

$$\begin{aligned}
\mathbf{E}\left[Q^{(i)} \;\middle|\; \mathsf{good}^{(i)}\right] &= \mathbf{E}\left[\sum_j Q_j^{(i)} \;\middle|\; \mathsf{good}^{(i)}\right] = \sum_j \mathbf{E}\left[Q_j^{(i)} \;\middle|\; \mathsf{good}^{(i)}\right] \\
&\leq \sum_j \frac{9}{2}\sqrt{\frac{N}{N_{i-1}/c - j + 1}} \\
&\leq (2N_i \ln N) \cdot 5\sqrt{c}\sqrt{N/N_{i-1}} \\
&= \begin{cases} 30\sqrt{c}N^{\frac{2^{l-1}-1}{2^l-1}}(\ln N)^2 & (i = 1) \\ 10\sqrt{c}N^{\frac{2^{l-1}-1}{2^l-1}}\ln N & (i \geq 2) \end{cases}
\end{aligned}$$

for sufficiently large N. Hence we have

$$\begin{aligned}
\mathbf{E}[Q \mid \mathsf{good}] &= \mathbf{E}\left[\sum_i Q^{(i)} \;\middle|\; \mathsf{good}\right] = \sum_i \mathbf{E}\left[Q^{(i)} \;\middle|\; \mathsf{good}^{(i)}\right] \\
&\leq 30\sqrt{c}N^{\frac{2^{l-1}-1}{2^l-1}}(\ln N)^2 + \sum_{i=2}^{l} 10\sqrt{c}N^{\frac{2^{l-1}-1}{2^l-1}}\ln N \\
&\leq 30l\sqrt{c} \cdot N^{\frac{2^{l-1}-1}{2^l-1}}(\ln N)^2 = \frac{1}{k}\mathsf{Qlimit}_k,
\end{aligned}$$

which completes the proof. $\square$

From the above claims it follows that

$$\begin{aligned}\mathbf{E}\left[Q\right] &\leq \mathbf{E}[Q \mid \mathsf{good}] + \mathbf{E}[Q \mid \neg\mathsf{good}] \Pr[\neg\mathsf{good}] \\ &\leq \left(\frac{1}{k} + \Pr[\neg\mathsf{good}]\right) \cdot \mathsf{Qlimit}_k, \end{aligned} \tag{22}$$

and

$$\begin{aligned}\Pr\left[\neg\mathsf{good}\right] &\leq \sum_i \Pr\left[\neg\mathsf{good}^{(i)}\right] \leq \sum_i \left(\exp\left(-\frac{1}{16c^2} N_{i-1} \ln N\right) + \frac{1}{N}\right) \\ &\leq l \exp\left(-\frac{N^{\frac{1}{2^l - 1}}}{16c^2}\right) + \frac{l}{N}. \end{aligned} \tag{23}$$

From Markov's inequality and Equation 22, the probability that Q reaches Qlimit_k is at most

$$\Pr\left[Q \geq \mathsf{Qlimit}_k\right] \leq \frac{\mathbf{E}[Q]}{\mathsf{Qlimit}_k} \leq \frac{1}{k} + \Pr[\neg\mathsf{good}]. \tag{24}$$

If Mclaw_k finds an l-claw, then Q does not reach Qlimit_k. Thus, from Equation 23 and Equation 24, the probability that Mclaw_k finds an l-claw is lower bounded by

$$1 - \left(\frac{1}{k} + l \cdot \exp\left(-\frac{N^{\frac{1}{2^l - 1}}}{16c}\right) + \frac{l}{N}\right), \tag{25}$$

which completes the proof. □

5 Conclusion

This paper has developed a new quantum algorithm to find multicollisions of random functions. Our new algorithm finds an l-collision of a random function $F\colon [N] \to [N]$ with $\tilde{O}\left(N^{(2^{l-1}-1)/(2^l-1)}\right)$ quantum queries, which improves the previous upper bound $O(N^{(3^{l-1})/(2\cdot 3^{l-1})})$ by Hosoyamada et al. [HSX17].[4] In fact, our algorithm can find an l-claw of random functions $f_1\colon [N] \to [N], \ldots, f_l\colon [N] \to [N]$ with the same complexity $\tilde{O}\left(N^{(2^{l-1}-1)/(2^l-1)}\right)$.

References

Amb04. Andris Ambainis. Quantum walk algorithm for element distinctness. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 22–31, 2004.

[4] As we mentioned in section 1, Liu and Zhandry [LZ18] showed that this bound is essentially tight.

BBHT98. Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505, 1998.

BDH$^+$01. Harry Buhrman, Christoph Dürr, Mark Heiligman, Peter Høyer, Frédéric Magniez, Miklos Santha, and Ronald de Wolf. Quantum algorithms for element distinctness. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 131–137, 2001.

BDRV18. Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Multi-collision resistant hash functions and their applications. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 133–161, 2018.

Bel12. Aleksandrs Belovs. Learning-graph-based quantum algorithm for k-distinctness. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 207–216, 2012.

BHT98. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN '98: Theoretical Informatics, Third Latin American Symposium, Campinas, Brazil, April, 20-24, 1998, Proceedings*, pages 163–169, 1998.

BKP18. Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 671–684, 2018.

CNS17. André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 211–240, 2017.

Gro96. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219, 1996.

HSX17. Akinori Hosoyamada, Yu Sasaki, and Keita Xagawa. Quantum multicollision-finding algorithm. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 179–210, 2017.

JLM14. Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 85–104, 2014.

KNY18. Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and *Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 162–194, 2018.

LZ18. Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. *IACR Cryptology ePrint Archive*, 2018:1096, 2018.

MU17. Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

RS96. Ronald L. Rivest and Adi Shamir. Payword and micromint: Two simple micropayment schemes. In *Security Protocols, International Workshop, Cambridge, United Kingdom, April 10-12, 1996, Proceedings*, pages 69–87, 1996.

Sho09. Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

Tan09. Seiichiro Tani. Claw finding algorithms using quantum walk. *Theor. Comput. Sci.*, 410(50):5285–5297, 2009.

Zha15. Mark Zhandry. A note on the quantum collision and set equality problems. *Quantum Information & Computation*, 15(7&8):557–567, 2015.