Quantum Cryptanalysis of Hash and Claw-Free Functions

(Invited Paper)

Gilles Brassard[1⋆], Peter Høyer[2⋆⋆], and Alain Tapp[1⋆⋆⋆]

[1] Université de Montréal, Département IRO
C.P. 6128, succursale centre-ville, Montréal (Québec), Canada H3C 3J7
{brassard,tappa}@iro.umontreal.ca
[2] Odense University, Department of Mathematics and Computer Science
Campusvej 55, DK–5230 Odense M, Denmark
u2pi@imada.ou.dk

Abstract. We give a quantum algorithm that finds collisions in arbitrary r-to-one functions after only $O(\sqrt[3]{N/r})$ expected evaluations of the function, where N is the cardinality of the domain. Assuming the function is given by a black box, this is more efficient than the best possible classical algorithm, even allowing probabilism. We also give a similar algorithm for finding claws in pairs of functions. Further, we exhibit a space-time tradeoff for our technique. Our approach uses Grover's quantum searching algorithm in a novel way.

1 Introduction

A *collision* for function $F : X \to Y$ consists of two distinct elements $x_0, x_1 \in X$ such that $F(x_0) = F(x_1)$. The *collision problem* is to find a collision in F under the promise that there is one.

This problem is of particular interest for cryptology because some functions known as *hash functions* are used in various cryptographic protocols. The security of these protocols depends crucially on the presumed difficulty of finding collisions in such functions. A related question is to find so-called *claws* in pairs of functions; our quantum algorithm extends to this task. In particular, this has consequences for the security of classical signature and bit commitment schemes.

A function F is said to be *r-to-one* if every element in its image has exactly r distinct preimages. We assume throughout this note that function F is given as a black box, so that it is not possible to obtain knowledge about it by any other means than evaluating it on points in its domain. When F is two-to-one,

⋆ Supported in part by Canada's NSERC, Québec's FCAR, and the Canada Council.
⋆⋆ Supported in part by the ESPRIT Long Term Research Programme of the EU under project number 20244 (ALCOM-IT). Research carried out while this author was at the Université de Montréal.
⋆⋆⋆ Supported in part by postgraduate fellowships from NSERC and FCAR

the most efficient classical algorithm possible for the collision problem requires an expected $\Theta(\sqrt{N})$ evaluations of F, where $N = |X|$ denotes the cardinality of the domain. This classical algorithm, which uses a principle reminiscent of the birthday paradox, is reviewed in the next section.

Recently, at a talk held at AT&T, Eric Rains [8] asked if it is possible to do better on a quantum computer. In this note, we give a positive answer to this question by providing a quantum algorithm that finds a collision in an arbitrary two-to-one function F after only $\Theta(\sqrt[3]{N})$ expected evaluations.

Earlier, Simon [9] addressed the XOR-*mask problem* defined as follows. Consider a positive integer n. We are given a function $F : \{0,1\}^n \to \{0,1\}^n$ and promised that either F is one-to-one or it is two-to-one and there exists an $s \in \{0,1\}^n$ such that $F(x_0) = F(x_1)$ if and only if $x_0 \oplus x_1 = s$, for all distinct $x_0, x_1 \in \{0,1\}^n$, where $\oplus$ denotes the bitwise exclusive-or. Simon's problem is to decide which of these two conditions holds, and to find s in the latter case. Note that finding s is equivalent to finding a collision in the case that F is two-to-one. Simon gave a quantum algorithm to solve his problem in expected time polynomial in n and in the time required to compute F. The running time required for this task on a quantum computer was recently improved to being polynomial in the worst case (rather than in the expected case), thanks to a more sophisticated algorithm [3]. Simon's algorithm is interesting from a theoretical point of view because any classical algorithm that uses only sub-exponentially (in n) many evaluations of F cannot hope to distinguish between the two types of functions significantly better than simply by tossing a coin, assuming equal *a priori* probabilities [9, 3]. Unfortunately, the XOR-mask constraint when F is two-to-one is so restrictive that Simon's algorithm has not yet found a practical application.

More recently, Grover [6, 7] discovered a quantum algorithm for a different searching problem. We are given a function $F : X \to \{0,1\}$ with the promise that there exists a unique $x_0 \in X$ so that $F(x_0) = 1$, and we are asked to find x_0. Provided the domain of the function is of cardinality a power of two ($N = 2^n$), Grover gave a quantum algorithm that finds the unknown x_0 with probability at least $1/2$ after only $\Theta(\sqrt{N})$ evaluations of F.

A natural generalization of this searching problem occurs when $F : X \to Y$ is an arbitrary function. Given some $y_0 \in Y$, we are asked to find an $x \in X$ such that $F(x) = y_0$, provided such an x exists. If $t = |\{x \in X \mid F(x) = y_0\}|$ denotes the number of different solutions, Grover's algorithm can be generalized [1] to find a solution whenever it exists ($t \geq 1$) after an expected number of $\Theta(\sqrt{N/t})$ evaluations of F. Although the algorithm does not need to know the value of t ahead of time, it is more efficient (in terms of the hidden constant in the O notation) when t is known, which will be the case for most algorithms given here. From now on, we refer to this generalization of Grover's algorithm as $\mathbf{Grover}(F, y_0)$. Note that the number of evaluations of F is not polynomially bounded in $\log N$ when $t \ll N$; nevertheless Grover's algorithm is considerably more efficient than classical brute-force searching.

In the next section, we give our new quantum algorithm for solving the collision problem for two-to-one functions. We then discuss a straightforward gen-

eralization to r-to-one functions and even to arbitrary functions whose image is sufficiently smaller than their domain. A natural space-time tradeoff emerges for our technique. Finally, we give applications to finding claws in pairs of functions.

2 Algorithms for the Collision Problem

We first state two simple algorithms for the collision problem, one classical and one quantum. Both of these algorithms use an expected number of $\Theta(\sqrt{N})$ evaluations of the given function, but the quantum algorithm is more space efficient. We derive our improved algorithm from these two simple solutions.

The first solution is a well-known classical probabilistic algorithm, here stated in slightly different terms than traditionally. The algorithm consists of three steps. First, it selects a random subset $K \subseteq X$ of cardinality $k = c\sqrt{N}$ for an appropriate constant c. Then, it computes the pair $(x, F(x))$ for each $x \in K$ and sorts these pairs according to the second entry. Finally, it outputs a collision in K if there is one, and otherwise reports that none has been found. Based on the birthday paradox, it is not difficult to show that if F is two-to-one then this algorithm returns a collision with probability at least $1/2$ provided c is sufficiently large ($c \approx 1.18$ will do). If we take a pair $(x, F(x))$ as unit of space then the algorithm can be implemented in space $\Theta(\sqrt{N})$, and $\Theta(\sqrt{N})$ evaluations of F suffice to succeed with probability $1/2$. If we care about running time rather than simply the number of evaluations of F, it may be preferable to resort to universal hashing [4] rather than sorting to find a collision in K. This would avoid spending $\Theta(\sqrt{N} \log N)$ time sorting the table, making possible a $\Theta(\sqrt{N})$ overall expected running time if we assume that each evaluation of F takes constant time. We stick to the sorting paradigm for simplicity and because it is not clear if the benefits of universal hashing carry over to quantum parallelism situations such as ours. We come back to this issue in Section 3.

The simple quantum algorithm for two-to-one functions also consists of three steps. First, it picks an arbitrary element $x_0 \in X$. Then, the algorithm computes $x_1 = \textbf{Grover}(H, 1)$ where $H : X \to \{0, 1\}$ denotes the function defined by $H(x) = 1$ if and only if $F(x) = F(x_0)$ but $x \neq x_0$. Finally, it outputs the collision $\{x_0, x_1\}$. There is exactly one $x \in X$ that satisfies $H(x) = 1$, so $t = 1$, and thus the expected number of evaluations of F is also $\Theta(\sqrt{N})$, still to succeed with probability $1/2$, but constant space suffices.

Our new algorithm, denoted **Collision** and given below, can be thought of as a logical union of the two algorithms above. The main idea is to select a subset K of X and then use **Grover** to find a collision $\{x_0, x_1\}$ with $x_0 \in K$ and $x_1 \in X \setminus K$. The expected number of evaluations of F and the space used by the algorithm are determined by the parameter $k = |K|$, the cardinality of K.

Collision(F, k)

1. Pick an arbitrary subset $K \subseteq X$ of cardinality k. Construct a table L of size k where each item in L holds a distinct pair $(x, F(x))$ with $x \in K$.
2. Sort L according to the second entry in each item of L.

3. Check if L contains a collision, that is, check if there exist distinct elements $(x_0, F(x_0)), (x_1, F(x_1)) \in L$ for which $F(x_0) = F(x_1)$. If so, proceed to step 6.
4. Compute $x_1 = \mathbf{Grover}(H, 1)$ where $H : X \to \{0, 1\}$ denotes the function defined by $H(x) = 1$ if and only if there exists $x_0 \in K$ so that $(x_0, F(x)) \in L$ but $x \neq x_0$.
5. Find $(x_0, F(x_1)) \in L$.
6. Output the collision $\{x_0, x_1\}$.

Theorem 1. *Given a two-to-one function $F : X \to Y$ with $N = |X|$ and an integer $1 \leq k \leq N$, algorithm* **Collision**(F, k) *returns a collision after an expected number of $\Theta(k + \sqrt{N/k})$ evaluations of F and uses space $\Theta(k)$. In particular, when $k = \sqrt[3]{N}$ then* **Collision**(F, k) *evaluates F an expected number of $\Theta(\sqrt[3]{N})$ times and uses space $\Theta(\sqrt[3]{N})$.*

Proof. The correctness of the algorithm follows easily from the definition of H and the construction of **Grover**$(H, 1)$.

We now count the number of evaluations of F. In the first step, the algorithm uses k such evaluations. Let p be the probability that a collision is found at step 3. If it is not found, set $t = |\{x \in X \mid H(x) = 1\}|$. By the previous section, subroutine **Grover** in step 4 uses an expected number of $\Theta(\sqrt{N/t})$ evaluations of the function H to find one of the t solutions. Note that each evaluation of H requires a single evaluation of F, and $t = k$ because F is two-to-one. Finally, our algorithm evaluates F once in step 5, giving a total expected number of $k + (1 - p)(\Theta(\sqrt{N/k}) + 1)$ evaluations of F. Provided N is sufficiently large, p is negligible when $k \leq \sqrt[3]{N}$ whereas $\sqrt{N/k} < k$ otherwise. In either case, the expected number of evaluations of F is $\Theta(k + \sqrt{N/k})$ as claimed. The second part of the theorem is immediate. □

In a nutshell, the improvement of our algorithm over the simple quantum algorithm is achieved by trading time for space. Suppose the cardinality of set K is large. Then the expected number of evaluations of H used by subroutine **Grover**$(H, 1)$ is small, but on the other hand more space is needed to store table L. Analogously, the space requirements are less but also **Grover**$(H, 1)$ runs slower if K is small.

Suppose now that we apply algorithm **Collision**, not necessarily on a two-to-one function, but on an arbitrary r-to-one function where $r \geq 2$. Then we have the following theorem, whose proof is essentially the same as that of Theorem 1.

Theorem 2. *Given an r-to-one function $F : X \to Y$ with $r \geq 2$ and an integer $1 \leq k \leq N = |X|$, algorithm* **Collision**(F, k) *returns a collision after an expected number of $\Theta(k + \sqrt{N/rk})$ evaluations of F and uses space $\Theta(k)$. In particular, when $k = \sqrt[3]{N/r}$ then* **Collision**(F, k) *uses an expected number of $\Theta(\sqrt[3]{N/r})$ evaluations of F and space $\Theta(\sqrt[3]{N/r})$.* □

Note that **Collision**(F, k) can also be applied on an arbitrary function $F : X \to Y$ for which $|X| \geq r|Y|$ for some $r > 1$, even if F is not r-to-one. However, the algorithm must be modified in two ways for the general case. First of all, the subset $K \subseteq X$ of cardinality k must be picked at random, rather than arbitrarily, at step 1. Furthermore, because the number of solutions for **Grover**$(H, 1)$ is no longer known in advance to be exactly $t = (r-1)k$, the fully generalized version of Grover's algorithm given in [1] must be used at step 4.

By varying k in Theorem 2, the following space-time tradeoff emerges.

Corollary 3. *There exists a quantum algorithm that can find a collision in an arbitrary r-to-one function $F : X \to Y$, for any $r \geq 2$, using space S and an expected number of $\Theta(T)$ evaluations of F for every $1 \leq S \leq T$ subject to*

$$ST^2 \geq |F(X)|$$

where $F(X)$ denotes the image of F. □

Consider now two functions $F : X \to Z$ and $G : Y \to Z$ that have the same codomain. By definition, a *claw* is a pair $x \in X$, $y \in Y$ such that $F(x) = G(y)$. Many cryptographic protocols are based on the assumption that there are efficiently-computable functions F and G for which claws cannot be found efficiently even though they exist in large number.

The simplest case arises when both F and G are bijections, which is the usual situation when such functions are used to create classical unconditionally-concealing bit commitment schemes [2] and strong digital signature schemes [5]. If $N = |X| = |Y| = |Z|$, algorithm **Collision** is easily modified as follows.

Claw(F, G, k)

1. Pick an arbitrary subset $K \subseteq X$ of cardinality k. Construct a table L of size k where each item in L holds a distinct pair $(x, F(x))$ with $x \in K$.
2. Sort L according to the second entry in each item of L.
3. Compute $y_0 =$ **Grover**$(H, 1)$ where $H : Y \to \{0, 1\}$ denotes the function defined by $H(y) = 1$ if and only if a pair $(x, G(y))$ appears in L for some arbitrary $x \in K$.
4. Find $(x_0, G(y_0)) \in L$.
5. Output the claw (x_0, y_0).

Theorem 4. *Given two one-to-one functions $F : X \to Z$ and $G : Y \to Z$ with $N = |X| = |Y| = |Z|$ and an integer $1 \leq k \leq N$, algorithm* **Claw**(F, G, k) *returns a claw after k evaluations of F and an expected number of $\Theta(\sqrt{N/k})$ evaluations of G, and uses space $\Theta(k)$. In particular, when $k = \sqrt[3]{N}$ then* **Claw**(F, G, k) *evaluates F and G an expected number of $\Theta(\sqrt[3]{N})$ times and uses space $\Theta(\sqrt[3]{N})$.*

Proof. Similar to the proof of Theorem 1. □

The case in which both F and G are r-to-one for some $r \geq 2$ and $N = |X| = |Y| = r|Z|$ is handled similarly. However, it becomes necessary in step 1 of algorithm **Claw** to select the elements of K so that no two of them are mapped to the same point by F. This will ensure that the call on **Grover**$(H, 1)$ at step 3 has exactly kr solutions to choose from. The simplest way to choose K is to pick random elements in X until $|F(K)| = k$. As long as $k \leq |Z|/2$, this requires trying less than $2k$ random elements of X, except with vanishing probability. The proof of the following theorem is again essentially as before.

Theorem 5. *Given two r-to-one functions $F : X \to Z$ and $G : Y \to Z$ with $N = |X| = |Y| = r|Z|$ and an integer $1 \leq k \leq N/2r$, modified algorithm* **Claw**(F, G, k) *returns a claw after an expected number of $\Theta(k)$ evaluations of F and $\Theta(\sqrt{N/rk})$ evaluations of G, and uses space $\Theta(k)$. In particular, when $k = \sqrt[3]{N/r}$ then* **Claw**(F, G, k) *evaluates F and G an expected number of $\Theta(\sqrt[3]{N/r})$ times and uses space $\Theta(\sqrt[3]{N/r})$.* □

3 Discussion

When we say that our quantum algorithms require $\Theta(k)$ space to hold table L, this corresponds unfortunately to the amount of *quantum* memory, a rather scarce resource with current technology. Note however that this table is built classically in the initial steps of algorithms **Collision** and **Claw**, and it contains classical information. Even though it has to be accessible in quantum superposition of addresses, it may be that the classical nature of the information it contains would make it easier to implement than a memory that can be used to store and retrieve quantum information. At the very least, as Eric Rains pointed out to us, it may require a simpler error-correction process than a general quantum memory would.

We considered only the number of evaluations of F in the analysis of algorithm **Collision**. The time spent sorting L and doing binary search in L should also be taken into account if we wanted to analyse the running time of our algorithm. If we assume that it takes time T to compute the function (rather than assuming that it is given as a black box), then it is straightforward to show that the algorithm considered in Theorem 2 runs in expected time

$$O\left((k + \sqrt{N/rk})(T + \log k)\right) .$$

Thus, the time spent sorting is negligible only if it takes $\Omega(\log k)$ time to compute F. Similar considerations apply to algorithm **Claw**. It is tempting to try using universal hashing [4] to bypass the need for sorting, as in the simple classical algorithm, but it is not clear that this approach saves time here because our use of quantum parallelism when we apply Grover's algorithm will take a time that is given by the *maximum* time taken for all requests to the table, which is unlikely to be constant even though the expected *average* time is constant.

References

1. Michel Boyer, Gilles Brassard, Peter Høyer and Alain Tapp, "Tight bounds on quantum searching", *Proceedings of Fourth Workshop on Physics and Computation — PhysComp '96*, November 1996, pp. 36–43. Final version to appear in *Fortschritte Der Physik*.
2. Gilles Brassard, David Chaum and Claude Crépeau, "Minimum disclosure proofs of knowledge", *Journal of Computer and System Sciences*, Vol. 37, no. 2, October 1988, pp. 156–189.
3. Gilles Brassard and Peter Høyer, "An exact quantum polynomial-time algorithm for Simon's problem", *Proceedings of Fifth Israeli Symposium on Theory of Computing and Systems — ISTCS '97*, June 1997, IEEE Computer Society Press, pp. 12–23.
4. J. Larry Carter and Mark N. Wegman, "Universal classes of hash functions", *Journal of Computer and System Sciences*, Vol. 18, no. 2, 1979, pp. 143–154.
5. Shafi Goldwasser, Silvio Micali and Ronald L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks", *SIAM Journal on Computing*, Vol. 17, 1988, pp. 281–308.
6. Lov K. Grover, "A fast quantum mechanical algorithm for database search", *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 1996, pp. 212–219.
7. Lov K. Grover, "Quantum mechanics helps in searching for a needle in a haystack", *Physical Review Letters*, Vol. 79, no. 2, 14 July 1997, pp. 325–328.
8. Eric Rains, talk given at AT&T, Murray Hill, New Jersey, 12 March 1997.
9. Daniel R. Simon, "On the power of quantum computation", *SIAM Journal on Computing*, Vol. 26, no. 5, October 1997, pp. 1474–1483.