Term project - Fall 2021
Quantum Cryptology and Security

Quantum Algorithm for the Collision Problem

TUFAN SINGHA MAHAPATRA, CRS2014

M.Tech. Cryptology and Security
Indian Statistical Institute Kolkata

This document is submitted as the term project for the course Quantum Cryptology and Security, for the MTech CRS coursework at ISI Kolkata. This term project is a based on a *arXiv 1997 paper* by Gilles Brassard,Alain Tapp from Universite de Montreal, and Peter Hoyer from Odense University .

Contents

1 Introduction

The emergence of large quantum computers is expected to pose a threat to most of the well-known public-key cryptographic primitives. We have already encountered that the primitives that rely on order-finding problems, for example factoring and computing Discrete Logarithms, can be broken using Shor's algorithm. In comparision to that, Symmetric primitives are expected to get impacted lesser. Though searching in an unstructured database to find a marked element can attain a quadratic speedup through Grover's Algorithm, which is indeed a notable improvement on previously available classical exhaustive search. Therefore, Cryptographers also started to rethink the notions of security of various schemes which are believed to be 'hard' to crack. One of the most common wayout is to double the length of the keys to maintain the same level of security.

In this project we focused on the main aspects of Grover's Algorithm and how it can help to present better algorithms for quantum collision. This project is based on a paper mentioned above.

Basically collision Finding 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 on the presumed difficulty of finding collisions in such functions.This quantum algorithm which is discussed on the above mentioned paper extends to this task. This has consequences for the security of classical signature and bit commitment schemes. In this paper they gave a quantum algorithm that finds collisions in $r - to - one$ functions after only $\mathcal{O}((N/r)^{1/3})$ expected evaluations of the function. Assuming the function is given by a black box, this is more efficient than the best possible classical algorithm, even allowing probabilism. This approach uses Grover's quantum searching algorithm in a novel way.

2 Quantum Computation

2.1 Quantum Oracle for functions

Any function $f : \{0,1\}^n \to \{0,1\}^k$ with a known classical circuit description can be effectively implemented as a quantum unitary O_f on $n+k$ qubits such that

$$U_f : |x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$$

2.2 Grover's Algorithm

Grover's algorithm is one of the most popular quantum algorithms that is used for searching an element in an unsorted array. Searching an element of interest among N elements in one of the basic tasks in computer science.The best known classical algorithm must query all the N elements in the Worst case.Surprisingly,Grover's algorithm can find the element, using just $\mathcal{O}(\sqrt{N})$ evaluations of the function, where N is the size of the function's domain. evaluations.

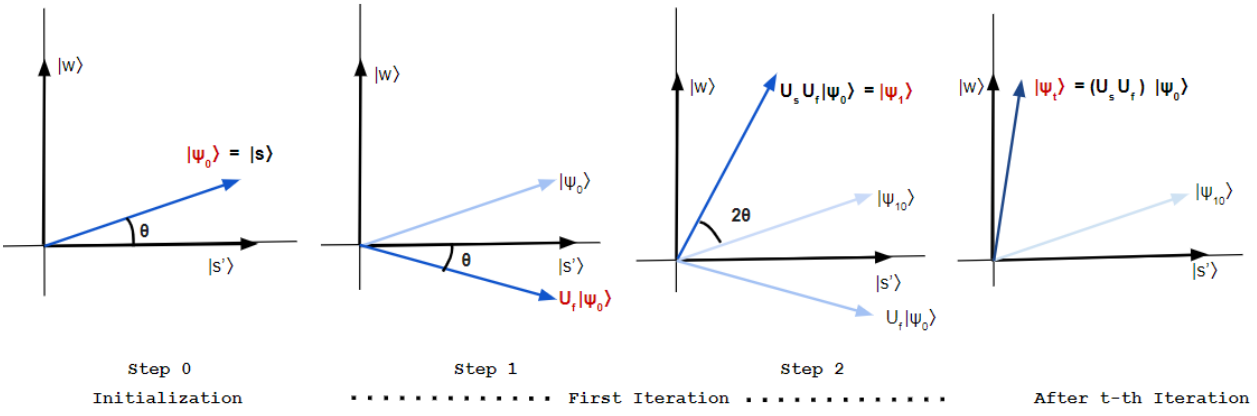
We are given an efficiently computable function $f : \{0,1\}^n \mapsto \{0,1\}$ and we want to find an element x such that $f(x) = 1$.

2.2.1 Problem Statement

Suppose $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let, $G = \{x \in \{0,1\}^n : f(x) = 1\}$ and $B = \{x \in \{0,1\}^n : f(x) = 0\}$. We assume that $|G| = a$ and $|B| = N - a = b$, where $N = 2^n$. The problem is to find an element of G, if exists or G is empty.

2.2.2 Geometric Interpretation

The algorithm starts with the uniform superposition $|s\rangle$, which by construction is $|s\rangle = H^{\otimes n} |0\rangle^n = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$. At time $t = 0$, the initial state is $|\psi_0\rangle = |s\rangle$. We here introduce an additional state $|s'\rangle$ in the $\text{SPAN}(|w\rangle, |s\rangle)$, which is obtained from $|s\rangle$ in such a way that for $\theta = \arcsin \langle s|w\rangle = \arcsin \frac{1}{\sqrt{N}}$, we have $|s\rangle = \sin\theta |w\rangle + \cos\theta |s'\rangle$. Note that, $|s'\rangle$ is perpendicular to $|w\rangle$. We start with assuming $|\psi_0\rangle = |s\rangle$. Firstly, the transformation U_f is a reflection about the orthogonal vector $|s'\rangle$ of the target vector $|w\rangle$. Hence, in the second figure, the $U_f |\psi_0\rangle$ is having same angle as $|\psi_0\rangle$ is having with $|s'\rangle$, but in opposite direction. We apply U_s again, which boosts the negative amplitude of $|w\rangle$ to roughly three times its original value. Hence, we reach closer to desire $|w\rangle$ vector. The next figure clears How the algorithm works in each step.



2.3 Comparison with Classical Algorithms

The aim of this algorithm is to find any element in the domain of $f : \{0,1\}^n \to \{0,1\}$, which maps to 1 or to conclude that it's an all-zero constant function. Grover's algorithm solves this problem in $\mathcal{O}(\sqrt{N})$ time.

2.4 Generalization Of Grover's Algorithm

A natural generalization of this searching problem occures when $F : X \to$ is an arbitrary function.Given $y_0 \in Y$,we are asked to find an $x \in X$ such that $F(x) = y_0$,provided such an x exists. If $t = |\{x \in X | F(x) = y_0\}|$ denotes the number of different solutions.Generalization of Grover's algorithm that can find a solution whenever it exists ($t \geq 1$) after an expected number of $\mathcal{O}(\sqrt{N/t})$ evaluations of F . From now on, we refer to this generalization of Grover's algorithm as **Grover**(F, y_0). Nevertheless Grover's algorithm is considerably more efficient than classical brute-force searching.

3 Collision Finding Problem

3.1 Problem Statement

Given access to a random function, find two distinct element, which maps to same element *i.e.* for random F: $\{0,1\}^n \to \{0,1\}^n$, find $x, y \in \{0,1\}^n$ with $x \neq y$ such that $F(x) = F(y)$.

3.2 Classical Algorithm

A function F is said to be r-to-one if every element in its image has exactly r many distinct Pre-Images. For now, we assume that function is guaranteed to be either one-to-one or two-to-one function. Hence, the main objective is to find if the function is injective or two-to-one function. Say, 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.

Now, this problem has a very simple well-known classical probabilistic algorithmic solution. Since F is two-to-one function, the most efficient classical algorithm possible for the collision problem requires an expected $\mathcal{O}(2^{\frac{n}{2}})$ evaluations of F, where 2^n the cardinality of the domain. This classical algorithm uses a principle reminiscent of the birthday paradox. The algorithm consists of three steps. First, it selects a random subset $K \subset X$ of cardinality $k = c2^{\frac{n}{2}}$ 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, if F is two-to-one then this algorithm returns a collision with probability at least $\frac{1}{2}$ provided c is sufficiently large .

3.3 Quantum Algorithm

The simple quantum algorithm for two-to-one functions also consists of three steps. The basic idea is that at first, it picks an arbitrary element $x_0 \in X$. Then, it computes $x_1 =$ **Grover**$(H, 1)$ where $H : X \to \{0,1\}$ denotes the function defined by $H(x) = 1$ if and only if $x \neq x_0$ and $F(x) = F(x_0)$. 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 $\mathcal{O}(2^{\frac{n}{2}})$, still to succeed with probability $\frac{1}{2}$, but constant space suffices. Our new algorithm, denoted Collision and given below, can be thought of as the 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, the cardinality of K. The complete algorithm is described in the following. Here, we present the algorithm for collision search from using Grover's Algorithm. This algorithm has query complexity $\mathcal{O}(2^{\frac{n}{3}})$.

Algorithm 1 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, goto step 6 otherwise go to next step.
4: Compute $x_1 =$ **Grover**$(H, 1)$ where $H : X$ß$0, 1$ 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 $x \neq x_0$. (Note that x_0 is unique if it exists since we already checked that there are no collisions in L ,otherwise we can simply go to step 6 after step 3.)
5: Find $(x_0, F(x_1)) \in L$.
6: Output the collision $\{x_0, x_1\}$.

3.4 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 $\mathcal{O}(k + \sqrt{N/k})$ evaluations of F and uses space $\Theta(k)$. In particular when $k = N^{1/3}$ then Collision(F, k) evaluates F an expected number of $\mathcal{O}(N^{1/3})$ times and uses space $\Theta(N^{1/3})$.

Proof:
We now count the number of evaluations of F . In the first step, the algorithm uses k such evaluations. Set $t = |\{x \in X | H(x) = 1\}|$ so $t = |\{x \in X | F(x) = F(x_0)\}|$. By the previous section,algorithm **Collision**(F, k) subroutine Grover in step 4 uses an expected number of $\mathcal{O}(\sqrt{N/t})$ evaluations of the function H to find one of the t solutions,because Grover's algorithm that can find a solution whenever it exists ($t \geq 1$) after an expected number of $\mathcal{O}(\sqrt{N/t})$ evaluations of F . Each evaluation of H can be implemented by using only one evaluation of F,because each evaluation of H can output only 0 or 1, for any output we need to verify either $F(x) = F(x_0)$ for $x \neq x_0$ or otherwise. Finally, our algorithm evaluates F once in the last step, giving a total expected number of $k + \mathcal{O}(\sqrt{N/t}) + 1$ evaluations of F .As in the **Collision**(F, k) algorithm $L = \{(x, F(x)) : x \in K\}$, and we sort L according to the second entry in each item of L. So each $F(x)$ is distinct in L .As $x_0 \in K$ and F is two-to-one so find an element $(x_0, F(x)) \in L$ and $x \neq K$ so t equals the cardinality of K,so $t = k$,so total expected number of $k + \mathcal{O}(\sqrt{N/k}) + 1$ evaluations of F is needed,so first part of the theorem is follows.
In particular if $k = N^{1/3}$ algorithm returns a collision of F after $\mathcal{O}(N^{1/3})$ and use the space as cardinality of K so the theorem is follows.

3.5 Theorem 2:

Given a $r-to-one$ 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 $\mathcal{O}(k + \sqrt{N/(rk)})$ evaluations of F and uses space $\Theta(k)$. In particular when $k = (N/r)^{1/3}$ then Collision(F, k) evaluates F an expected number of $\mathcal{O}((N/r)^{1/3})$ times and uses space $\Theta((N/r)^{1/3})$.

Proof:
We now count the number of evaluations of F . In the first step, the algorithm uses k such evaluations. Set $t = |\{x \in X|H(x) = 1\}|$ so $t = |\{x \in X|F(x) = F(x_0)\}|$. By the previous section,algorithm **Collision**(F, k) subroutine Grover in step 4 uses an expected number of $\mathcal{O}(\sqrt{N/t})$ evaluations of the function H to find one of the t solutions,because Grover's algorithm that can find a solution whenever it exists ($t \geq 1$) after an expected number of $\mathcal{O}(\sqrt{N/t})$ evaluations of F . Each evaluation of H can be implemented by using only one evaluation of F,because each evaluation of H can output only 0 or 1, for any output we need to verify either $F(x) = F(x_0)$ for $x \neq x_0$ or otherwise. Finally, our algorithm evaluates F once in the last step, giving a total expected number of $k + \mathcal{O}(\sqrt{N/t}) + 1$ evaluations of F .As in the **Collision**(F, k) algorithm $L = \{(x, F(x)) : x \in K\}$, and we sort L according to the second entry in each item of L. So each $F(x)$ is distinct in L .As $x_0 \in K$ and F is $r-to-one$ so find an element $(x_0, F(x)) \in L$ and $x \neq K$ so t equals the cardinality of rK,so $t = rk$, so total expected number of $k + \mathcal{O}(\sqrt{N/(rk)}) + 1$ evaluations of F is needed,so first part of the theorem is follows.
In particular if $k = (N/r)^{1/3}$ algorithm returns a collision of F after $\mathcal{O}((N/r)^{1/3})$ and use the space as cardinality of K so the theorem is follows.

4 Conclusion

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 needs to live in quantum memory for read purposes only. In practice, it may be easier to build large read-only quantum memories than general read/write memories. 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.

References

[1] Brassard, Gilles & Høyer, Peter & Tapp, Alain (1997). *Quantum Algorithm for the Collision Problem.*

[2] Tharrmashastha SAPV , Debajyoti Bera , Arpita Maitra , Subhamoy Maitra *Quantum Algorithms for Cryptographically Significant Boolean Functions*