5-26-2017

An Improved Lower Bound for the Quantum Query Complexity of Collision Finding in Hash Functions with Non-Uniformly Distributed Images

Marko Balogh
Portland State University

Recommended Citation

Balogh, Marko, "An Improved Lower Bound for the Quantum Query Complexity of Collision Finding in Hash Functions with Non-Uniformly Distributed Images" (2017). *University Honors Theses.* Paper 425.

10.15760/honors.421

An Improved Lower Bound for the Quantum Query Complexity of Collision Finding in Hash Functions with Non-Uniformly Distributed Images

by

Marko Balogh

An undergraduate honors thesis submitted in partial fulfillment of the

requirements for the degree of

Bachelor of Science

in

University Honors

and

Physics

Thesis Adviser

Dr. Fang Song

Portland State University

2017

Abstract

This work proves an improved lower bound for the quantum query complexity of collision-finding in hash functions whose images are distributed according to a non-uniform distribution. Recent work by [TTU16] applied the leftover hash lemma to show that at least $\Omega(2^{k/9})$ quantum queries are necessary to find a collision when the image distribution has min-entropy k. In comparison, a result by [Zha15] implies that $\Omega(2^{k/3})$ quantum queries are necessary if the distribution is uniform. In this paper the lower bound complexity of [Zha15] is extended directly to the non-uniform case using a chain of reductions which convert collisions in min-entropy k distributions into collisions in the uniform distribution with constant probability. This result shows a minimum security guarantee for hash functions under more general assumptions in which the image distribution may not be uniform.

1 Introduction

A hash function is a cryptographic primitive which exploits a property called *collision resistance* for various purposes [KL15]. Collision resistance refers to the difficulty of locating a collision in the hash function, that is, finding two distinct inputs which the function maps to the same output. This property is particularly non-trivial because the domain of a hash function is usually binary strings of arbitrary length, while the co-domain is a binary string of a fixed length. It follows that a large number of collisions must exist, and the hash function must be designed so that these collisions are difficult to locate in the domain. The collision resistance property is what makes hash functions useful in cryptographic applications like message authentication [BCK96] and password hashing [Kal00].

For the analysis of hash function collision-resistance, the hash function's output is typically assumed to be uniformly random and accessible to attackers only as a black box oracle, a paradigm known as the random oracle model [KL15]. The collision-resistance of the hash function is then identified with the asymptotic complexity (in terms of the number of queries to the oracle) of finding a collision. However, in practice, some properties of specific hash functions may be exploited by attackers, making the real-world security of the hash functions less than the theoretical analysis in the random oracle model suggests. For example, some minimal information about the distribution of the images of the hash function may be exploited. In this work, we analyze the query complexity of collision-finding in hash functions known to have an image distribution of a certain min-entropy. We do so by proving a chain of reductions that extends a previously established lower bound query complexity in the random oracle model to hash functions with non-uniform image distributions of min-entropy k. We do this entirely in the setting of quantum attacks, although the reductions we show can be easily applied in the classical setting as well.

2 Preliminaries

Here we introduce the reader to a few important notations and definitions.

We consider functions $f : X \to Y$ for some arbitrary X and Y and use Y^X to denote the set of all such functions. The notation $f \leftarrow Y^X$ indicates that f is a function sampled uniformly from Y^X. In general, we consider distributions on the co-domain Y. The notation

$f \leftarrow D^X$ indicates that f is a function from X to Y sampled from the distribution of functions induced by sampling each image independently from a distribution D on Y.

Definition 2.1 (Min-Entropy)**.** Let D be a distribution on some set Y. Let $D(y)$ denote the probability mass corresponding to a $y \in Y$ under distribution D. D is said to have min-entropy k if $k = -\log_2(\max_{x \in X}\{D(x)\})$. We refer to a distribution of min-entropy k as a min-k distribution or simply a k-distribution.

Definition 2.2 (Flat-k-Distribution)**.** We call a k-distribution D on set Y with support S a flat-k-distribution, denoted $D_{k,\flat}$, if the cardinality of S is exactly 2^k. It follows that $\forall x \in S$, $D(x) = 2^{-k}$.

Definition 2.3 (δ-k-Distribution)**.** We call a k-distribution D on set Y a δ-k-distribution if there is a unique $m \in Y$ such that $\forall y \in Y$

$$D(y) = \begin{cases} 2^{-k} & \text{if } y = m\,; \\ \frac{1-2^{-k}}{|Y|-1} & \text{otherwise}\,, \end{cases}$$

and may denote such a distribution $D_{k,\delta}$. Notice that m is the mode of D, since D has min-entropy k. The support of D is the entire set Y, and remaining probability mass $1 - 2^{-k}$ is distributed uniformly among all elements in Y other than the mode.

Definition 2.4 (Function of min-entropy k)**.** Let D be a distribution with min-entropy k on some set Y. Let $f : X \to Y$ be a (in general non-uniformly) random function such that the image of each input under f is an independent random sample from set Y according to distribution D. We denote sampling of a function in this way as $f \leftarrow D^X$. We say that f is a function of min-entropy k.

Definition 2.5 (Collision Problem)**.** Let $f : X \to Y$ be a function of min-entropy k, whose images are distributed according to D. A pair of elements $x_1 \in X$ and $x_2 \in X$ such that $x_1 \neq x_2$ and $f(x_1) = f(x_2)$ is called a *collision* in f. We refer to the problem of producing such a pair as the *collision finding problem* or *collision finding problem* in D. Suppose an algorithm $\mathcal{A}$ outputs a collision in f. Then we say that $\mathcal{A}$ has *solved* collision finding in D.

Definition 2.6 (Quantum Oracle Access)**.** Suppose $\mathcal{O}$ is an oracle for some function f. Suppose $\mathcal{A}$ is a quantum algorithm which can submit queries to $\mathcal{O}$ in a quantum superposition (called a *quantum query*) and receive $\mathcal{O}$'s responses in a quantum superposition. Specifically, $\mathcal{A}$ can implement $\mathcal{O}$ as a unitary transformation $\sum \alpha_{x,y,z}|x, y, z\rangle \mapsto \sum \alpha_{x,y,z}|x, y + f(x), z\rangle$. Then we say that $\mathcal{A}$ has *quantum oracle access* to f and denote this as $\mathcal{A}^f$.

3 Lower bounds: finding a collision is difficult

To prove our query complexity lower bounds, we make use of proof by reduction. We show that collision finding in any min-k distribution is at least as difficult as collision finding in a uniform distribution on a set of size 2^k. We begin by demonstrating a reduction of collision finding in a uniform distribution to collision finding in a flat-k distribution. Then we show a reduction of collision finding in a flat-k distribution to collision finding in a general k-distribution. These reductions show that any algorithm which is not able to solve collision finding in a uniform distribution on a set of size 2^k is also not able to solve collision finding in any k-distribution. This then allows us to extend the query complexity lower bound for collision finding in a uniform distribution due to Zhandry [Zha15] to collision finding in any k-distribution.

Each of the following reduction proofs describe an algorithm (which we may refer to as 'the reduction') attempting to find a collision in a random function f to which it has oracle access. The algorithm will run as a subroutine another algorithm which is capable of finding a collision in another random function g when given oracle access to g. Therefore, the proofs must account for how the reduction algorithm satisfies two requirements: A) that it interacts with the subroutine algorithm in such a way that, from the perspective of the subroutine, the subroutine is interacting with an oracle for the random function g, and B) that the collision in g returned by the subroutine can be converted into a collision in f with sufficiently high probability. The way the reduction satisfies requirement B by necessity depends on the way it satisfied requirement A, since converting a collision in g into a collision in f necessarily depends on how the values of g are generated from the values of f. Hence a single procedure should be used by the reduction to satisfy both requirements. We will refer to such a procedure as a *collision-conversion procedure*. Intuitively, a collision-conversion procedure is an algorithm that in some way processes the responses of the an oracle f so that its output (perfectly) simulates an oracle for g, and then inverts that process to convert a collision in g into a collision in f. The formal definition follows.

Definition 3.1 (Collision-conversion procedure)**.** Let $\mathcal{C}$ be an algorithm that is given access to an oracle f whose responses are distributed according some distribution D. For each unique query submitted to it, $\mathcal{C}$ returns an independent random sample drawn according to some distribution D'. Furthermore, when given as input a pair (x_1, x_2), $\mathcal{C}$ returns either some collision in f, (x_1', x_2'), in which case we say $\mathcal{C}$ *succeeds*, or returns $\perp$ otherwise. Then we call $\mathcal{C}$ a collision-conversion procedure from D to D'. We say $\mathcal{C}$ *succeeds with probability at least* p if the conditional probability that $\mathcal{C}$ succeeds, given that $\mathcal{C}$'s responses to queries x_1 and x_2 were equal, is at least p, regardless of the values x_1 and x_2.

In the context of proving lower bounds for query complexity, we must make conservative assumptions with respect to security and therefore bold assumptions about the capabilities of the reduction algorithm—giving the algorithm any advantages it could conceivably have. In particular, we assume the following:

- The reduction has a full description of the distribution in which the subroutine can solve the collision problem, since it is possible that an adversary will have some or even all information relating to the image distribution of the function it is attacking. Formally, suppose the subroutine solves the collision problem in D, a distribution on Y. For any $y \in Y$, let $D(y)$ denote $\Pr[y' = y : y' \leftarrow D]$, the probability that a random sample from D is equal to y. We assume that the reduction has perfect knowledge of $D(y)$ for all $y \in Y$.

- The reduction has access to a 'sufficiently large' amount of randomness. While we will be more explicit about this later, the intuition is that the reduction will need to sample from a distribution generated by the distribution D. As we will not put any restriction on this distribution (other than having min-entropy k), it is not clear how much randomness is needed to sample from it. However it is clear that a large but finite amount of randomness is sufficient to sample arbitrarily close to this distribution.

- The reduction is not computationally limited. It is conceivable that an adversary with advance knowledge of the function it is attacking may perform an exponential amount of precomputation to speed up collision finding.

Before diving into the details of the proofs, a brief comment: the proofs to follow are all described as fundamentally *classical* algorithms. This is especially visible in the fact that individual queries are processed sequentially by oracles and the algorithms that interact with them. The definition of a collision-conversion procedure above is also implicitly classical in this way. Nonetheless, the reductions below can easily be generalized to the quantum setting because nowhere do the reductions utilize knowledge of the characteristics of the set of prior queries and responses. This is an important feature because quantum algorithms can query an oracle on a superposition of inputs and receive the responses in a superposition. Hence any sort of dynamic behavior in which an oracle's response depends on some analysis of prior behavior is not possible in the quantum setting. The fact that the reductions we present below can be (with the exception of the final step) parallelized over the set of all possible queries indicates their compatibility with quantum computing.

We begin by showing some reusable general results that will allow us to quickly construct reductions and extend query complexity lower bounds by simply demonstrating the existence of a satisfactory collision-conversion procedure for use in each reduction.

Lemma 3.2. *Suppose there exists an algorithm $\mathcal{A}$ which solves collision finding in a particular distribution $D_{\mathcal{A}}$ with probability at least $P_{\mathcal{A}}$, using q queries to an oracle whose responses are distributed according to $D_{\mathcal{A}}$. Suppose there exists a collision-conversion procedure from $D_{\mathcal{A}}$ to a distribution $D_{\mathcal{R}}$ that succeeds with probability at least 1/2. Then there exists an algorithm which solves collision finding in $D_{\mathcal{R}}$ with probability at least $P_{\mathcal{A}}/2$ using q queries to an oracle whose responses are distributed according to $D_{\mathcal{R}}$.*

Proof. We describe such an algorithm, which we call $\mathcal{R}$ for reduction. Upon initialization, run $\mathcal{A}$ and the collision-conversion procedure, which we call $\mathcal{C}$, simultaneously. Let $f_{\mathcal{R}}$ denote the oracle whose responses are distributed according to distribution $D_{\mathcal{R}}$. Grant $\mathcal{C}$ access to the oracle $f_{\mathcal{R}}$ and access to all other resources available to the reduction. For each of $\mathcal{A}$'s queries, forward the query to $\mathcal{C}$, and return $\mathcal{C}$'s response back to $\mathcal{A}$. When $\mathcal{A}$ outputs a collision candidate, forward the collision candidate to $\mathcal{C}$. Return the output of $\mathcal{C}$.

That this algorithm works is clear. By the definition of a collision-conversion procedure, the algorithm $\mathcal{A}$ interacts with an oracle (call it $f_{\mathcal{A}}$) whose responses are distributed according to $D_{\mathcal{A}}$. Hence $\mathcal{A}$ runs as usual, producing a collision in $f_{\mathcal{A}}$ with probability $P_{\mathcal{A}}$. Again, by the definition of a collision-conversion procedure, $\mathcal{C}$ will return a collision in $f_{\mathcal{R}}$ at least half of the time that it is given collision in $f_{\mathcal{A}}$. It follows that, at a minimum, the probability that $\mathcal{R}$ outputs a collision in $f_{\mathcal{R}}$ is $P_{\mathcal{A}}/2$. Observe also that $\mathcal{R}$ uses exactly one query for every query used by $\mathcal{A}$. Hence lemma 3.2 follows. □

Corollary 3.3. *Suppose there is some upper bound $G(q)$ on the probability that any algorithm making q queries to an oracle $f_{\mathcal{R}}$, whose responses are distributed according to a distribution $D_{\mathcal{R}}$, finds a collision in $f_{\mathcal{R}}$. Suppose there exists a collision-conversion procedure from a distribution $D_{\mathcal{A}}$ to $D_{\mathcal{R}}$ that succeeds with probability at least 1/2. Then $2 \cdot G(q)$ is an upper bound on the probability that any algorithm making q queries to an oracle $f_{\mathcal{A}}$, whose responses are distributed according to $D_{\mathcal{A}}$, finds a collision in $f_{\mathcal{A}}$.*

Proof. This follows directly from contraposition of lemma 3.2. The negation of the consequent of lemma 3.2 is that the probability that any algorithm solves collision finding in $D_{\mathcal{R}}$ using q queries to an oracle whose responses are distributed according to $D_{\mathcal{R}}$ must be less than $P_{\mathcal{A}}/2$. The negation of the antecedent is that the probability any algorithm solves collision finding in $D_{\mathcal{A}}$ using q queries to an oracle whose responses are distributed according to $D_{\mathcal{A}}$ must be less than $P_{\mathcal{A}}$. Expressing this contrapositive with $2 \cdot G(q)$ taking the role of $P_{\mathcal{A}}$ gives the above corollary. □

Lemma 3.4 ([Zha15] Theorem 3.1)**.** *There is a universal constant C such that the following holds. Let $f : [M] \to [N]$ be a uniformly random function. Then any algorithm making q quantum queries to f outputs a collision in f with probability at most $C(q+1)^3/N$.*

Theorem 3.5. *Let $f_{flat} \leftarrow D_{k,\flat}^{X}$ be a random function whose outputs are chosen according to a flat-k-distribution $D_{k,\flat}$. Then any algorithm making q queries to f_{flat} outputs a collision with probability at most $C(q+1)^3/2^k$, for some constant C.*

Proof. This result follows from corollary 3.3 combined with lemma 3.4, with a 2^k replacing N. Hence all that needs to be demonstrated to show theorem 3.5 is that there exists a collision-conversion procedure from a uniform distribution on a set of size 2^k to a flat-k-distribution $D_{k,\flat}$ which succeeds with probability at least 1/2.

In this case the collision-conversion procedure $\mathcal{C}$ is nearly trivial. Let $f : X \to Y$, where $|Y| = 2^k$, be a uniformly random function. Let $f_{flat} : X' \to Y'$ be a function whose outputs are chosen according to a flat-k-distribution $D_{k,\flat}$ on Y' with support S. Upon initialization, prepare any injective mapping $g : S \to Y$. This can be done by randomly sampling from Y, without replacement, for each element in S. For each query x, forward the query to the oracle f which $\mathcal{C}$ has access to, and respond with $g^{-1}(f(x))$. When a pair (x_1, x_2) is received, output (x_1, x_2).

This is, indeed, a collision-conversion procedure from a flat-k-distribution to a uniform distribution on a set of size 2^k. Since g is a permutation, g^{-1} will map a uniform element of Y to a uniform element of S. Since $f(x)$ is, by the definition of a uniformly random function, a uniform element of Y, $g^{-1}(f(x))$ is a uniform element of S. This means that the distribution of $g^{-1}(f(x))$ is the flat-k-distribution on support S. Therefore, when queried, $\mathcal{C}$ returns an independent random sample from the flat-k-distribution, as required. If the responses of $\mathcal{C}$ to the queries x_1 and x_2 are equal, then $g^{-1}(f(x_1)) = g^{-1}(f(x_2))$. Since g is a permutation, this implies $f(x_1) = f(x_2)$. Hence the conditional probability that $f(x_1) = f(x_2)$ given that the responses of $\mathcal{C}$ to the queries x_1 and x_2 are equal is 1, so $\mathcal{C}$ is a collision-conversion procedure from a flat-k-distribution to a uniform distribution on a set of size 2^k which succeeds with probability 1. □

We will show a few theorems of nearly identical structure as theorem 3.5, and in general we will write all of the constant factors in the probabilities as C, even though they will not all take the same numerical value, in recognition that they are not interesting for the study of asymptotic query complexity. The extention of Zhandry's upper bound on the probability of solving collision finding in a uniform distribution was to solving collision finding in a flat distribution with an equally large support was trivial, since the difference between the two problems is little more than finding a permutation between the two distributions' supports. Now we extend Zhandry's upper bound from flat k-distributions to general k-distributions.

This is significantly more difficult because the distributions' supports may differ in size, and the collision-conversion procedure will necessarily be non-deterministic, since the Shannon entropy may differ between a flat k-distribution and an arbitrary k-distribution.

Theorem 3.6. *Let $f_D \leftarrow D^X$ be a random function whose outputs are chosen according to a distribution D of min-entropy k. Then any algorithm making q queries to f_D outputs a collision with probability at most $C(q+1)^3/2^k$, for some constant C.*

Corollary 3.7. *Any quantum algorithm needs at least $\Omega(2^{k/3})$ queries to find a collision with constant probability in a random function $f_D \leftarrow D^X$ whose outputs are chosen according to a distribution D of min-entropy k.*

Proof of Theorem 3.6. Once again this result follows from Theorem 3.5 applied to corollary 3.3. Hence to demonstrate theorem 3.6 a collision-conversion procedure from a flat-k-distribution to an arbitrary k-distribution, which succeeds with probability at least $1/2$, needs to be shown. We provide an example below.

Let D be an arbitrary k-distribution on support S. Let D_{flat} be a flat-k-distribution on support S_{flat}. Let $\mathcal{C}$ denote the collision-conversion procedure. Upon initialization, $\mathcal{C}$ does the following:

- Prepare to store a lookup table for a function $c : S \times S_{flat} \rightarrow [0, 1]$
- Sort and label the elements y_i of S in order of decreasing probability mass under distribution D, so that $D(y_1) = 2^{-k}$ (by the definition of min-entropy, there must be one or more $y_i \in S$ with $D(y_i) = 2^{-k}$)
- Arbitrarily label the elements z_j of S_{flat} with index $j = 1, 2, \ldots, 2^k$.
- Iterate the following over $i = 1, 2, \ldots, |S|$:
  - If $D(y_i) = 2^{-k}$, set $c(y_i, z_i) = 1$. Set $c(y_i, z_j) = 0$ for all $j \neq i$. Continue to the next value of i.
  - If $D(y_i) < 2^{-k}$, compute $\sum_{j=1}^{i-1} D(y_j)$ (here j is a dummy-index for the sum and is unrelated to the labeling of the elements of S_{flat}) and save the result as the image of i under a function $g : [|S|] \rightarrow [0, 1]$. Check if $g(i)$ is a multiple of 2^{-k}.
    - If so, set $c(y_i, z_{((g(i)/2^{-k})+1)}) = 2^k D(y_i)$ and $c(y_i, z_j) = 0$ for all $j \neq (g(i)/2^{-k}) + 1$.
    - If not, set $c(y_i, z_{\lceil g(i)/2^{-k} \rceil}) = 2^k \min(\lceil g(i)/2^{-k} \rceil - g(i), D(y_i))$, set $c(y_i, z_{\lceil g(i)/2^{-k} \rceil + 1}) = 2^k (D(y_i) - \min(\lceil g(i)/2^{-k} \rceil - g(i), D(y_i)))$, and set $c(y_i, z_j) = 0$ for all remaining $z_j \in S_{flat}$
- For each $z_j \in S_{flat}$, construct the set W_j containing all $y_i \in S$ for which $c(y_i, z_j) \neq 0$. Store the set W_j as the image of z_j in a function $b : S_{flat} \rightarrow \mathcal{P}(S)$.

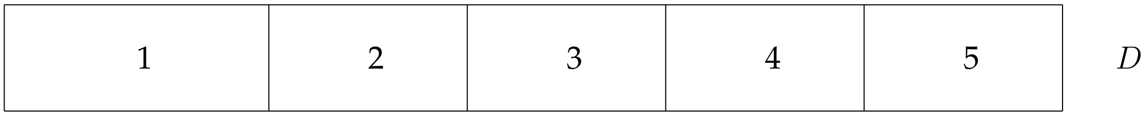
Now $\mathcal{C}$ enters the query-response phase. Recall that $\mathcal{C}$ has access to an oracle whose responses are distributed according to D_{flat}; denote this oracle f_{flat}. Suppose that $\mathcal{C}$ is queried on some $x \in S$. $\mathcal{C}$ responds via the following:

- Query f_{flat} on x. Fix some mapping from an index t to each element in $b(f_{flat}(x))$. Compute $m_x(y_t) = c(y_t, f_{flat}(x))$ for each $y_t \in b(f_{flat}(x))$. Sample $r \leftarrow [0, 1]$. Note: In practice, sampling such an r would of course take an infinite amount of randomness. If we only use a finite amount of randomness, then for some distributions we may introduce some small amount of error. However by increasing this amount of randomness, we can make this error arbitrarily small such that any q query adversary A cannot detect the error. As we do not care about the efficiency of the reduction $\mathcal{C}$, only the query complexity, the actual amount of randomness needed for this is not relevant.

- Iterate through $i = 1, \ldots, |b(f_{flat}(x))|$ until i has a value such that the following condition holds: $\sum_{t=1}^{i} m_x(y_t) \geq r$. Return y_i.

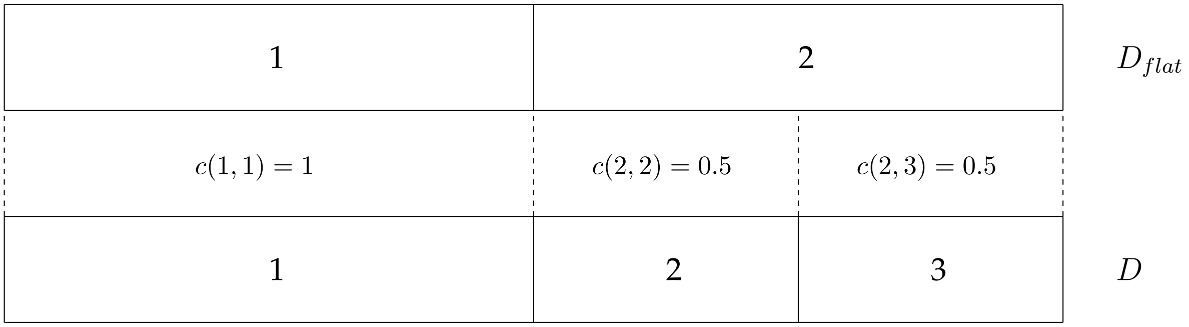
Finally, when $\mathcal{C}$ receives a pair (x_1, x_2), it outputs the same pair (x_1, x_2).

To facilitate intuitive understanding of how $\mathcal{C}$ fulfills the requirements of a collision-conversion procedure, we visualize a distribution D as a rectangle divided into disjoint regions, each region representing one element of the support S. The total area of the rectangle is 1, representing the total probability mass in D. For simplicity, consider momentarily a distribution of min-entropy 2 on a set of size 5, whose elements we label with the first 5 positive integers. In the distribution represented below, the 1 element has 25% of the probability mass, while the others share the remaining 75%. Thus 1 is the mode element and its probability mass determines the min-entropy of the distribution.



Denote the oracle simulated by the query-response phase of $\mathcal{C}$ as f_D. Before it terminates, $\mathcal{C}$ receives a pair (x_1, x_2), which it attempts to convert into a collision in f_{flat}. In order to convert a collision in f_D into a collision in f_{flat}, $\mathcal{C}$ must map the elements of S into the elements of S_{flat}. Additionally, this mapping must be consistent with $\mathcal{C}$'s responses to queries, so that a collision another algorithm discovers from $\mathcal{C}$'s responses is likely to correspond to a collision in the oracle f_{flat}.

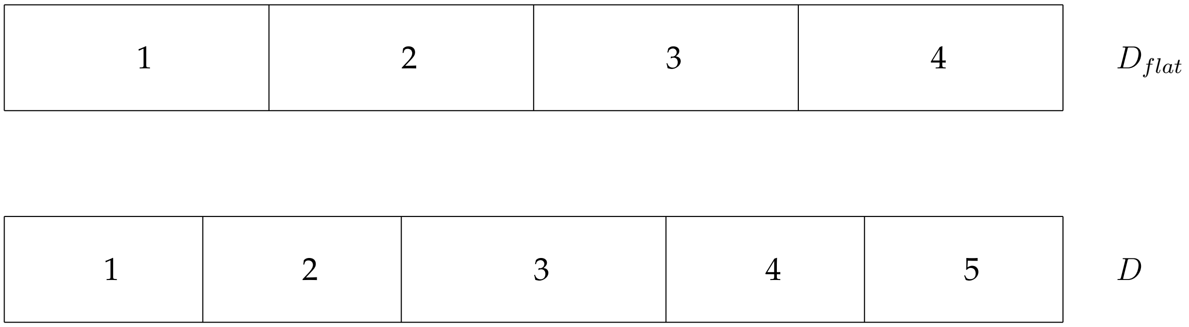
Clearly, an oracle f_D with responses distributed according to D generally cannot be perfectly simulated by a deterministic mapping of the responses of the oracle f_{flat}. Additional randomness generally must be added by $\mathcal{C}$ because the distribution D may have higher Shannon entropy than the distribution D_{flat}, as $|S| \geq |S_{flat}|$. However, the oracle f_D can be simulated by treating the elements of S_{flat} as 'bins', each associated with one or more elements of S. In order to generate a sample from S, a sample from S_{flat} first selects a 'bin', and then one of the elements of S associated with that bin is chosen randomly according a conditional probability distribution such that the marginal probability of sampling that element is equal to probability associated with that element under distribution D. This process can be visualized intuitively by vertically aligning rectangular representations of the distributions D_{flat} and D. The conditional probability of sampling each element in S given a bin in S_{flat} can be illustrated by projecting the dividers between elements in the rectangle representing D into the space between the two rectangles. We show a trivial example below, using distributions of min-entropy 1.



The diagram above also illustrates the role played by the function $c : S \times S_{flat} \to [0,1]$. This function specifies the conditional probability that $\mathcal{C}$ returns a sample of a certain element of S given that the oracle f_{flat} responded with a certain element from S_{flat}. Formally, $c(y,z) = \Pr[f_D(x) = y | f_{flat}(x) = z]$, where x is a query and $f_D(x)$ is $\mathcal{C}$'s response. All that remains of the collision-conversion procedure then is sampling from the chosen bin according to the conditional probabilities that $c(y,z)$ specifies. The values encoded into the function c therefore make up the non-trivial part of the protocol.

In the example above, the elements 2 and 3 have a total probability mass of 0.5 in D, so they can be 'binned' into element 2 in D_{flat}. Supposing that the 2 element is returned by the oracle f_{flat}, a single uniformly random bit would determine whether $\mathcal{C}$ will return 2 or 3 as the response of f_D, since each are equally likely under D. If element 1 is returned by f_{flat}, no additional randomness is needed, as $c(1,1) = 1$. This is always the case for the mode element, because its probability mass under D must exactly equal the probability mass of any of the elements of D_{flat}, since D and D_{flat} have equal min-entropy.

This procedure will perfectly simulate responses from an oracle f_D whose responses are distributed according to D, since the marginal probability of sampling each element in this fashion exactly replicates the associated probability in D. In this simple case, any collision found in f_D will necessarily be a collision in f_{flat}. However, this is not true in general. If the elements of S cannot be grouped into bins each with total probability mass under D equal to 2^{-k}, then some elements of S must have their probability mass split among multiple bins. An example of such a case is shown below.

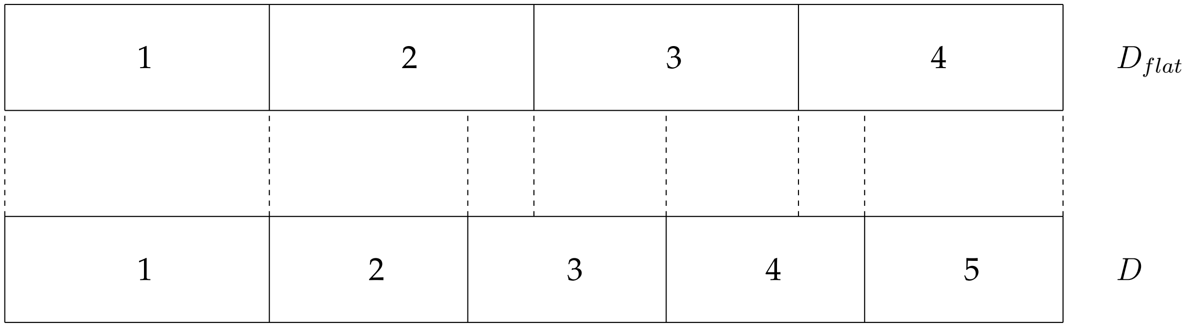


In cases like these, it is possible that a pair of identical responses from $\mathcal{C}$, constituting an apparent collision in f_D, do not actually originate from identical responses from f_{flat}, and therefore do not constitute an actual collision in f_{flat}. Luckily, it is possible to construct the function c such that the probability that a collision in f_D corresponds to a collision in f_{flat} is bounded below by one half, so that C qualifies as a collision-conversion procedure and theorem 3.6 follows. The initialization stage of $\mathcal{C}$ contains a general method for constructing such a function, which we explain now.

The first step is to sort the elements of S in order of decreasing probability under distribution D. The utility of this is that it guarantees that any elements of S which can be trivially associated with an element in S_{flat}, because they have a probability mass equal to 2^{-k}, are mapped to a single element in S_{flat} with probability 1.

Next, $\mathcal{C}$ iterates over the elements y_i of S, setting the value of $c(y_i, z_j)$ for all elements z_j of S_{flat} for each. This may be visualized as moving across the rectangular representations of D and D_{flat} from left to right, determining the values of the collision conversion function along the way. If the probability mass of y_i under D is 2^{-k}, then all of its probability mass is associated with the element in S_{flat} with the same index, so $c(y_i, z_i) = 1$ (and of course zero for all other z_j). If the probability mass corresponding to y_i in D is less than 2^{-k}, then the probability mass from y_i will not 'occupy' an entire bin in D_{flat}, so it must share a bin with other elements of S. But if the current bin is already partially occupied, it may be the case that the probability mass of y_i has to be split between the current bin and the next bin, where the 'current bin' and 'next bin' refer to the elements of S_{flat} which, at this point in execution of $\mathcal{C}$, have the highest index out of the elements for which c has already been assigned and the lowest index out of the elements for which c has not already been assigned, respectively.

To check whether this is the case, $\mathcal{C}$ computes the total probability mass in S that has already been assigned, which it saves as $g(i)$, and checks whether this value is a multiple of 2^{-k}. If it is, then the current bin must be completely occupied, and the probability mass corresponding to y_i will fit completely inside the next bin. The next bin will in this case be indexed by $(g(i)/2^{-k}) + 1$. If $g(i)$ is not a multiple of 2^{-k}, then the probability mass from element y_i may need to be split between the current bin and the next bin. In this case, the index of the current bin will be the ceiling of $g(i)/2^{-k}$, because if $g(i)/2^{-k} = 2.1$ then the first two bins are completely occupied, and one tenth of the third bin is completely occupied, making the index of the current bin 3. The index of the next bin will thus be the index of the current bin plus one. The value of $c(y_i, z_{\lceil g(i)/2^{-k} \rceil})$, representing the conditional probability of sampling y_i given the current bin has been sampled from D_{flat}, is set to $2^k \min(\lceil g(i)/2^{-k} \rceil - g(i), P(y_i))$. The minimum function guarantees that if it is possible to fit all the probability mass from y_i into the current bin then this is done, and if not, whatever probability mass can fit into the current bin is assigned to the current bin. Naturally, whatever probability mass is not assigned to the current bin must be assigned to the next bin, to conserve marginal probability. The factors of 2^k come from the denominator of 2^{-k} in the conditional probability. Continuing like this, the entire collision-conversion function is constructed. It should be clear that the procedure just described would lead to a c function like the one illustrated below, for the example distribution D which we introduced earlier. In general, the c function that results from this procedure can be visualized by projecting the dividers between elements in *both* rectangles into the space between the two rectangles.



$c(1,1) = 1$

$c(2,2) = 0.75$

$c(3,2) = 0.25$

$c(3,3) = 0.5$

$c(4,3) = 0.5$

$c(4,4) = 0.25$

$c(5,4) = 0.75$

At the end of the initialization stage, for each element in S_{flat}, $\mathcal{C}$ saves the set of all elements in S which are associated with it via the c function. These sets are saved in a function b and will be used to 'invert' (using the term loosely) the c function for the purpose of simulating the responses of an oracle f_D whose responses are distributed according to D.

Now we explain the query-response phase of C. Suppose x is one of the queries submitted to $\mathcal{C}$. The query is forwarded to the oracle f_{flat}, and then retrieves the set $b(f_{flat}(x))$ and the associated conditional probabilities from values of the c function. Next, all $\mathcal{C}$ has to do is sample from the conditional distribution specified by c. It is simple to verify that the next few steps properly do so via inverse transform sampling.

Finally, $\mathcal{C}$ outputs whatever pair of elements is given to it. In order to qualify as a collision-conversion procedure, the probability that $\mathcal{C}$'s output is a collision in f_{flat} conditioned on the fact that it is given a pair that is a collision in f_D must be greater than 1/2. We now show that this is indeed the case.

Suppose $\mathcal{C}$ is given a pair (x_1, x_2). Let y denote $f_D(x_1)$, $\mathcal{C}$'s response to query x_1, and likewise let y' denote $f_D(x_2)$. Let z denote $f_{flat}(x_1)$ and z' denote $f_{flat}(x_2)$. Then the probability that $\mathcal{C}$ *succeeds* in the sense defined in definition 3.1 can be expressed as $\Pr[z = z'|y = y']$. By Bayes' theorem,

$$\Pr[z = z'|y = y'] = \frac{\Pr[z = z']}{\Pr[y = y']}\Pr[y = y'|z = z']. \tag{1}$$

Applying the law of total probability, we may write

$$\Pr[z = z'] = \sum_{j=1}^{2^k} \Pr[z = z'|z = z_j] \cdot \Pr[z = z_j],$$

$$\Pr[z = z'] = \sum_{j=1}^{2^k} \Pr[z' = z_j|z = z_j] \cdot \Pr[z = z_j],$$

and, because z and z' are *independent* random samples according to distribution D_{flat}, this becomes

$$\Pr[z = z'] = \sum_{j=1}^{2^k} \Pr[z' = z_j] \cdot \Pr[z = z_j].$$

By the definition of a flat-k-distribution, the probability that a sample according to D_{flat} takes any specific value z_j is equal to 2^{-k}. Hence we find that

$$\Pr[z = z'] = \sum_{j=1}^{2^k} (2^{-k})^2 = 2^{-2k} \sum_{j=1}^{2^k} 1 = 2^{-k}.$$

Following the same reasoning just used other than the final step, we may also write

$$\Pr[y = y'] = \sum_{i=1}^{|S|} (D(y_i))^2.$$

Then equation (1) can be rewritten as

$$\Pr[z = z' | y = y'] = \frac{2^{-k}}{\sum_{i=1}^{|S|} (D(y_i))^2} \Pr[y = y' | z = z']. \tag{2}$$

Now we turn our attention to the conditional probability on the right. Applying the law of total conditional probability, we decompose the conditional probability into a sum over all possible values of the random variable z, so

$$\Pr[y = y' | z = z'] = \sum_{j=1}^{2^k} \Pr[y = y' | z = z' \wedge z = z_j] \cdot \Pr[z = z_j | z = z'].$$

Applying Bayes' Theorem again, this time to the conditional expression on the far right, inside the summand, we get

$$\Pr[y = y' | z = z'] = \sum_{j=1}^{2^k} \Pr[y = y' | z = z' \wedge z = z_j] \cdot \frac{\Pr[z = z_j]}{\Pr[z = z']} \cdot \Pr[z = z' | z = z_j].$$

Using our prior result for $\Pr[z = z']$, and the fact that all samples according to D_{flat} have probability 2^{-k}, we can see that the ration in the above equation must be exactly one. Hence we get

$$\Pr[y = y' | z = z'] = \sum_{j=1}^{2^k} \Pr[y = y' | z = z' = z_j] \cdot \Pr[z' = z_j] = \sum_{j=1}^{2^k} \Pr[y = y' | z = z' = z_j] \cdot 2^{-k}.$$

Once again applying the law of total conditional probability, this time decomposing the conditional probability in the summand into a sum over all possible values of the random variable y, we get

$$\Pr[y = y' | z = z'] = 2^{-k} \sum_{j=1}^{2^k} \sum_{i=1}^{|S|} \Pr[y = y' | z = z' = z_j \wedge y = y_i] \cdot \Pr[y = y_i | z = z' = z_j],$$

which we rewrite as

$$\Pr[y = y'|z = z'] = 2^{-k} \sum_{j=1}^{2^k} \sum_{i=1}^{|S|} \Pr[y' = y_i|z' = z_j] \cdot \Pr[y = y_i|z = z_j].$$

This step is only possible because y' and z are independent, so the presence of z' in the conditional involving y' can be ignored. The same goes for y and z' in the second conditional. Now, recall that the function c is defined by $c(y, z) = \Pr[f_D(x) = y|f_{flat}(x) = z]$. It follows, by the definitions of y, y', z, and z', that each of the factors in the summand are equal to $c(y_i, z_j)$. Hence we may write

$$\Pr[y = y'|z = z'] = 2^{-k} \sum_{i=1}^{|S|} \sum_{j=1}^{2^k} (c(y_i, z_j))^2,$$

in which we have reversed the order of summation. From the details of how the values of c are assigned when $\mathcal{C}$ is initialized, the number of j values for which $c(y_i, z_j)$ is non-zero is either one (if all of y_i's probability mass is associated with a single bin), or two (if the probability mass is split over two bins). If, for a given i only one value of j corresponds to a non-zero $c(y_i, z_j)$, then $c(y_i, z_j)$ must be $2^k D(y_i)$. But we are interested in the worst case, in order to establish a lower bound. If, for a given i, two values of j correspond to a non-zero $c(y_i, z_j)$, then we know that the two values of c must sum to $2^k D(y_i)$. In this case, the sum of the squares of these values will be less than $2^k D(y_i)$. We can express this sum as $c_1^2 + c_2^2 = c_1^2 + (2^k D(y_i) - c_1)^2$. The minimum value for this parabola is $2^{2k-1} D(y_i)^2$, when $c = 2^{k-1} D(y_i)$. Therefore we may write,

$$\Pr[y = y'|z = z'] \geq 2^{-k} \sum_{i=1}^{|S|} 2^{2k-1} (D(y_i))^2 = 2^{k-1} \sum_{i=1}^{|S|} (D(y_i))^2.$$

Substituting this expression into equation (2), we get

$$\Pr[z = z'|y = y'] \geq \frac{2^{-k}}{\sum_{i=1}^{|S|} (D(y_i))^2} \cdot 2^{k-1} \sum_{i=1}^{|S|} (D(y_i))^2 = \frac{1}{2}.$$

Therefore we conclude that

$$\Pr[\mathcal{C}\ \mathit{succeeds}] \geq \frac{1}{2}.$$

Hence $\mathcal{C}$ satisfies the definition of a collision-conversion procedure from a flat-k-distribution to an arbitrary k-distribution.

□

Although Theorem 3.5's upper bound on the success probability of any algorithm in collision finding clearly extends to a δ-k-distribution, since a δ-k-distribution is merely a special case of a k-distribution for which the theorem holds, it is possible to show a reduction of collision finding in an arbitrary k-distribution to collision finding in a δ-k-distribution. This is interesting because it affirms that the δ-k-distribution case is the most difficult out of all k-distributions. The proof is easy to find by mimicking the proof of theorem 3.5 but replacing all references of 2^{-k} as the probability of sampling each element from the flat distribution with a general probability $D(x)$, and replacing the general distribution D with a

δ-k-distribution D_δ. This works in the case that no elements in the support of D are associated with a probability mass less than $1/N$. One has to employ the idea of computational indistinguishability to extend this result to the case that there are some elements associated with smaller probability mass than $1/N$.

References

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. pages 1–15. Springer-Verlag, 1996.

[Kal00] B Kaliski. Password-based cryptography specification, 2000. Available at https://www.ietf.org/rfc/rfc2898.txt.

[KL15] Jonathan Katz and Yehuda Lindell. Introduction to modern cryptography, 2015.

[TTU16] Ehsan Ebrahimi Targhi, Gelo Noel Tabia, and Dominique Unruh. Quantum collision-resistance of non-uniformly distributed functions. In *International Workshop on Post-Quantum Cryptography*, pages 79–85. Springer, 2016.

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