Quantum Collision-Finding in Non-Uniform Random Functions

Marko Balogh[1], Edward Eaton[2], and Fang Song[3]

[1] Department of Physics, Portland State University `marko_balogh@me.com`
[2] Department of Combinatorics and Optimization, University of Waterloo `eeaton@uwaterloo.ca`
[3] Department of Computer Science, Portland State University `fsong@pdx.edu`

Abstract. We study *quantum* attacks on finding a collision in a *non-uniform* random function whose outputs are drawn according to a distribution of min-entropy k. This can be viewed as showing *generic* security of hash functions under *relaxed* assumptions in contrast to the standard heuristic of assuming uniformly random outputs. It is useful in analyzing quantum security of the Fujisaki-Okamoto transformation [32]. In particular, our results close a gap left open in [31].
Specifically, let D be a distribution of min-entropy k on a set Y. Let $f : X \to Y$ be a function whose output $f(x)$ is drawn according to D for each $x \in X$ independently. We show that $\Omega(2^{k/3})$ quantum queries are necessary to find a collision in f, improving the previous bound $\Omega(2^{k/9})$ [31]. In fact we show a stronger lower bound $2^{k/2}$ in some special case. For most cases, we also describe explicit quantum algorithms matching the corresponding lower bounds.

1 Introduction

Hash functions are central and prominent in modern cryptography, and there have been many ingenious designs of cryptographic hash functions [27,4,13,2]. One significant property of a cryptographic hash function H, backed with intensive tests in practice, is *collision resistance*. Namely, it should be computationally unfeasible to find a *collision*, which is a pair of distinct input strings (x, x') with $H(x) = H(x')$. Because of this nice feature, hash functions are being used in numerous cryptographic constructions and applications, e.g. protecting passwords [1], constructing message authentication codes and digital signature schemes, as well as various crypto-currencies exemplified by BitCoin [26].

Theoretical analysis of a hash function H often refers to *generic* security, where one ignores the internal design of H and views it as a black box. Moreover, the output of H is assumed to have been drawn *uniformly* at random from some codomain of size N. The complexity of finding a collision is then measured by the number of evaluations of H, i.e. queries to the black box. By the well-known *birthday* bound, $\Theta(\sqrt{N})$ queries are both sufficient and necessary to find a collision in H. These principles are extended and formalized as the *random oracle* model, in which a hash function is treated as a truly random function that is publicly available but only through oracle queries [11]. This heuristic has been widely adopted to construct more efficient cryptosystems and facilitate security reduction proofs which are otherwise challenging or unknown [12,21].

However, in reality, there are attacks that perform significantly better than the plain birthday attack. The recent explicit break of full SHA-1 by Google [30], where two PDF files can be generated that collide on the same 160-bit digest, only takes $\sim 2^{61}$ hash evaluations instead of the 2^{80} expected via the birthday attack. These attacks are possible because the internal structure of H may create opportunities for more effective cryptanalysis. A natural reaction would be to figuratively open up the black box and take into account the inner workings case-by-case when analyzing a hash function. Alternatively, *can we prove generic security bounds, but under relaxed and/or more accurate assumptions?*

The approaching era of quantum computing will make these challenges more worrisome. The power of quantum computers, while promising in accelerating the resolution of fundamental problems in many areas such as chemistry, biology, etc., raises a tremendous threat to cryptography. Many public key cryptosystems will be broken due the Shor's efficient quantum algorithm for the factoring and discrete logarithm problems upon which they are based [28]. In addition, new features of quantum adversaries are difficult and subtle to deal with, especially in the setting of cryptographic protocols. In fact many classical security analyses become inapplicable or even fail completely in the presence of quantum adversaries [17,34,23].

Pertaining to hash functions, a quantum adversary is able to implement the hash function as a quantum circuit and evaluate it in quantum *superposition*. Therefore, if H is treated as a black box, it is reasonable to allow a quantum adversary query H in quantum superposition: $\sum_x \alpha_x |x, 0\rangle \mapsto \sum_x \alpha_x |x, H(x)\rangle$. Although this does not imply that the adversary can learn the entirety of H in one query, an immediate difficulty, for example, is the failure of the "lazy sampling" trick, where one can simulate a random function by sampling random responses on-the-fly. Indeed, much effort has been devoted to extending the results and useful techniques in the classical random oracle model to the quantum setting (formalized as the quantum random oracle model) [14,39,9,19]. Notably, Zhandry [38] shows that $\Theta(N^{1/3})$ quantum queries are both sufficient and necessary to find a collision in a uniformly random function. This establishes the generic security of uniformly random hash functions. But as classical attacks have illustrated, assuming uniform randomness is sometimes too optimistic and risky. Such concerns are becoming more pressing due to recent advances in the physical realization of quantum computers [3,5]. Optimized architectures are also reducing the cost of implementing quantum algorithms (e.g., see an estimation of Grover's quantum search algorithm [10]).

This motivates the question we study in this work: *what is the complexity of finding a collision in a* ***non-uniform*** *random function, under quantum attacks in particular*? Specifically we consider a distribution D_k on set Y which has min-entropy k, i.e. the most likely element occurs with probability 2^{-k}. We want to find a collision in a function $H : X \to Y$ where for each $x \in X$,

$H(x)$ is drawn independently according to D_k. We call it a rand-min-k function hereafter. Note that if D_k is uniform over Y (hence $|Y| = 2^k$), this becomes the standard uniformly random function. Given H as a black-box, we are interested in the number of queries needed by a quantum algorithm to find a collision in H. As a result, this will establish the generic security of hash functions under a *relaxed condition* where the outputs of a hash function are drawn from a distribution of min-entropy k rather than a strictly uniform distribution. This condition might be a more realistic heuristic for a good hash function. Roughly speaking, a hash function designer will only need to make sure that there is no single value $y \in Y$ that has a large set of preimages (i.e., $f^{-1}(y) := \{x \in X : f(x) = y\}$ with $|f^{-1}(y)| \leq 2^k$). In contrast, modeling a hash function as a uniformly random function would require certain *regularity* such that the preimage set of every codomain element has roughly the same size, which may be difficult to justify and test in practice. We also note that a concrete application of collision finding in rand-min-k functions appears in the famous Fujisaki-Okamoto transformation [21], whose quantum security has been studied in [32].

Classically, it is not difficult to derive a variation of the birthday bound, which gives $\Theta(2^{k/2})$ as the query complexity in typical cases. In the quantum setting, Targhi et al. [31] prove that $\Omega(2^{k/9})$ queries are necessary for any quantum algorithm to find a collision with constant probability. Compared to the tight bound $2^{k/3}$ in the uniform case, the bound is unlikely to be optimal and the gap seems significant. In addition, no quantum algorithms are described or analyzed formally.Overall, our understanding of finding a collision in non-uniform random functions is far from satisfying as far as quantum attacks are concerned.

1.1 Our contributions

In this work, we characterize the complexity of finding collisions in a rand-min-k function when it is given as an oracle to a quantum algorithm. We are able to prove matching upper and lower bounds in many cases. The results are summarized in Table 1.

D_k	M, N, k **settings**	**Upper bound**	**Lower bound**
All	$M = o(\beta^{1/2})$ (inj. by Lemma 2)	∞	∞
All	$M = \Omega(\beta^{1/2})$	$\beta^{1/3}$ (Thm. 5)	$2^{k/3}$ (Cor. 2)
flat-k	$M = \Omega(2^{k/2})$	$2^{k/3}$ (Thm. 5)	$2^{k/3}$ (Cor. 2)
δ-min-k	$M = \Omega(N^{1/2})$, $2^k \leq N < 2^{3k/2}$	$N^{1/3}$ (Thm. 5)	$N^{1/3}$ (Cor. 3)
	$M = \Omega(N^{1/2})$, $2^{3k/2} \leq N < 2^{2k}$	$2^{k/2}$ (Thm. 6)	$2^{k/2}$ (Cor. 3)
	$M = \Omega(2^k)$, $N \geq 2^{2k}$	$2^{k/2}$ (Thm. 6)	$2^{k/2}$ (Cor. 3 also Cor. 5)

Table 1. Summary of quantum collision finding in rand-min-k functions. $\beta := \frac{1}{\Pr[x=y:x,y\leftarrow D]}$ is the collision variable, which equals 2^k for flat-distributions (i.e., uniform on a subset of size 2^k), and lies in $[2^k, 2^{2k}]$ for δ-min-k distributions (i.e., peak at one element, and uniform elsewhere), as well as for general min-k distributions. Here M refers to the size of the domain and N refers to the size of the codomain.

A simple special case is the flat distribution, which is uniform on a subset of size 2^k. In this case, not surprisingly, the same bound $2^{k/3}$ for the uniform random function holds. Another special case, which represents the hardest instances, concerns the δ-min-k distributions, where there is a mode element with probability mass 2^{-k} and the remaining probability mass is distributedly uniformly throughout the rest of the codomain. Here we show that $2^{k/2}$ queries are both sufficient and necessary. For general min-k distributions, the complexity is characterized by the *collision variable* $\beta(D)$ for a distribution D, which is the reciprocal of the probability that two independent samples from D collide. We prove a generic upper bound $\beta^{1/3}$, and a lower bound $2^{k/3}$. For comparison, classically one can show that the (generalized) birthday bound $\Theta(\beta^{1/2})$, which equals $\Theta(N^{1/2})$ for uniform distributions, precisely depicts the hardness of finding a collision.

Technical overview. For the generic lower bound $2^{k/3}$, we follow the natural idea of reducing from collision finding in uniform random functions (Theorem 3). We describe a sequence of reductions showing that finding a collision in uniform random functions of codomain size 2^k reduces to that in flat distributions, and then to general min-k distributions. Therefore the $2^{k/3}$ lower bound follows. This approach is in contrast to that in [31], where they basically extract close-to uniform bits from the output of a rand-min-k function f by composing f with a universal hash function h. Note that a collision in f is also a collision in $h \circ f$. In addition, $h \circ f$ can be shown to be quantum indistinguishable from a uniformly random function by a general theorem of Zhandry [37], which relates sample-distinguishability to oracle-distinguishability. Therefore any adversary for rand-min-k can be turned into an adversary for $h \circ f$, contradicting the hardness for uniformly random functions. However, the discrepancy between $h \circ f$ and a uniformly random function gets accumulated and amplified in the sample-to-oracle lifting step, and this may explain the slackness in their lower bound $2^{k/9}$.

Instead, given an oracle f whose images are distributed according to a distribution D, our reductions employ a *redistribution function* to simulate an oracle f' whose images are distributed according to another distribution D' on Y'. A redistribution function r maps a pair $(x, f(x))$ to an element in Y', and r is sampled from a proper distribution such that $f'(x) := r(x, f(x))$ is distributed according to D', taking into account the random choice of f as well. We show algorithms for sampling appropriate redistribution functions, called *redistribution function samplers*, for the distributions we are concerned with. As a result, we can use an adversary for the collision-finding problem in D' to attack the collision-finding problem in D. To complete the reductions, we show that a collision found in the simulated oracle for f' will indeed be a valid collision in f with probability at least $1/2$.

Along the same lines, it is possible to demonstrate that collision-finding in δ-min-k distributions is the hardest case. In fact, we are able to establish rigorously a *strengthened* lower bound in this case (Theorem 4). Our proof proceeds by showing indistinguishability between a random δ-min-k function on a codomain of size N and a uniformly random function on the same codomain. Then the lower bound in the uniform case translates to a lower bound for the δ-min-k case. The exact bounds vary a bit for different relative sizes of N and k.

Establishing upper bounds is relatively easy (Theorem 5). We adapt the quantum algorithm of [38] in the uniform case. Basically we partition the domain of a rand-min-k function f into subsets of proper size, so that when restricting f on each subset, there exists a collision with at least constant probability. Next, we can invoke the collision finding algorithm by Ambainis [8] on each restricted function, and with a few iterations, a collision will be found.

Moreover, we give alternative proofs showing the lower bound for δ-min-k distributions (Theorem 7) and upper bound for all min-k distributions (Theorem 6). They are helpful to provide more insight and explain the bounds intuitively. Specifically, we reduce an average-case search problem, of which the hardness has been studied [24], to finding a collision in a δ-min-k random function. On the other hand, when the mode element of a min-k distribution is known, we show that applying Grover's quantum search algorithm almost directly will find a collision within $O(2^{k/2})$ queries. This actually improves the algorithms above in some parameter settings.

1.2 Discussion

Collision finding is an important problem in quantum computing, and a considerable amount of work in this context exists. Brassard et al. [16] give a quantum algorithm that finds a collision in any two-to-one function $f : [M] \to [N]$ with $O(N^{1/3})$ quantum queries. Ambainis [8] gives an algorithm based on quantum random walks that finds a collision using $O(M^{2/3})$ queries whenever there is at least one collision in the function. Aaronson and Shi [6] and Ambainis [7] give an $\Omega(N^{1/3})$ lower bound for a two-to-one function f with the same domain and co-domain of size N. Yuen [36] proves an $\Omega(N^{1/5}/\text{polylog}N)$ lower bound for finding a collision in a uniformly random function with a codomain at least as large as the domain. This is later improved by Zhandry [38] to $\Theta(N^{1/3})$ for general domain and codomain as we mentioned earlier.

We stress that, typically in quantum computing literature, the lower bounds are proven for the worst-case scenario and with constant success probability. This in particular does not rule out adversaries that succeed with an inverse polynomial probability which is usually considered a break of a scheme in cryptography. Hence a more appropriate goal in cryptography would be showing the number of queries needed for achieving any (possibly low) success probability, or equivalently bounding above the success probability of any adversary with certain number of queries. Our results, as in [38,31], are proven in the strong sense that is more appropriate in cryptographic settings.

Our work leaves many interesting possible directions for future work. One immediate unsatisfying feature of our reductions is that they may take a long time to implement. Can they be made time efficient? We have been mainly concerned with finding one collision; it is interesting to investigate the complexity of finding *multiple* collisions in a non-uniform random function. There are other important properties of hash functions such as preimage resistance and second-preimage resistance, which are both weaker than and implied by collision resistance. Hence, our lower bound results also demonstrate the hardness of finding a preimage and second preimage, though the bounds are not necessarily tight. In Appendix D we extend the techniques for collision resistance and prove tight bounds for preimage- and second-preimage resistance against quantum generic attacks. Finally, we note that a stronger notion for hash functions called *collapsing* has been proposed which is very useful in the quantum setting [33]. Can we prove that rand-min-k functions are collapsing? Note that a uniform random function is known to be collapsing, and more recently it has been shown that the sponge construction in SHA-3 is collapsing (in the quantum random oracle model) [18].

Independent work. In a concurrent and independent work by Ebrahimi and Unruh [20], they give twelve bounds for quantum collision finding of min-k random functions. We observe that ten of them coincide with our bounds, and in particular, they present essentially the same quantum collision-finding algorithms as ours. The remaining two are generic lower bounds improving upon their prior work [31], which are $\Omega(2^{k/5})$ and $\Omega(\beta^{1/9})$ (in our notation). Our bounds are stronger – $\Omega(2^{k/3})$ and $\Omega(\beta^{1/6})$ (by noting that $\beta \leq 2^{2k}$) respectively.

2 Preliminaries

Here we introduce a few notations and definitions. We also discuss basic results concerning the collision probability and birthday bound in min-k distributions.

Let D be a discrete probability distribution on set Y defined by probability mass function $D(y) := \Pr_{z \leftarrow D}[z = y]$. The support of D is $\mathrm{Supp}(D) := \{y \in Y : D(y) > 0\}$. We denote $Y^X := \{f : X \to Y\}$ the set of functions for some domain X and codomain Y. The notation $f \leftarrow Y^X$ indicates that f is a function sampled uniformly from Y^X.

Definition 1 (Min-Entropy). *Let D be a distribution on set Y. D is said to have min-entropy k if $k = -\log_2(\max_{y \in Y}\{D(y)\})$. We refer to a distribution of min-entropy k as a min-k distribution or simply a k-distribution.*

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

Definition 3 (δ-k-Distribution). *We call a k-distribution D on set Y a δ-k-distribution if there is a unique mode element $m \in Y$ such that $\forall y \in Y$*

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

We denote such a distribution $D_{k,\delta}$. It is implicit that $|Y| > 2^k$. 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 4 (Function of min-entropy k). *Let D be a min-k distribution on set Y. We define D^X to be the distribution on Y^X such that for every $x \in X$, its image is sampled independently according to D. $f \leftarrow D^X$ denotes sampling a function in this way, and we say that f is a function of min-entropy k.*

Definition 5 (Collision problem). *Let $f \leftarrow D^X$ be a function of min-entropy k. 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 in D.*

Definition 6 (Quantum oracle access). *A quantum oracle $\mathcal{O}$ for some function f implements a unitary transformation: $\sum \alpha_{x,y,z}|x, y, z\rangle \overset{\mathcal{O}}{\mapsto} \sum \alpha_{x,y,z}|x, y + f(x), z\rangle$. An algorithm $\mathcal{A}$ algorithm that makes (quantum superposition) queries to $\mathcal{O}$ is said to have quantum oracle access to f, and is denoted $\mathcal{A}^f$.*

2.1 Collision probability and non-uniform birthday bound

Definition 7. *The* collision probability *of a probability distribution D is defined to be the probability that two independent samples from D are equal. Namely*

$$\mathrm{CP}(D) := \Pr_{y_1, y_2 \leftarrow D}[y_1 = y_2] = \sum_{y \in Y} D(y)^2 \,.$$

We call $\beta(D) := \frac{1}{\mathrm{CP}(D)}$ the collision variable of D.

$\beta(D)$ will be an important variable determining the complexity of collision finding. In fact we can derive an birthday bound for collisions in an arbitrary distribution D in terms of $\beta(D)$, analogous to the case of uniform distributions, using a key lemma by Wiener [35].

Lemma 1. *([35, Theorem 3]) Let R_D be the random variable denoting the number of i.i.d. samples from a distribution D until a collision appears for the first time. Let $q \geq 1$ be an integer and $\gamma_q := \frac{q-1}{\sqrt{\beta(D)}}$*

$$\Pr(R_D > q) \leq e^{-\gamma_q}(1 + \gamma_q) \,.$$

Corollary 1. *Let $y_1, \dots, y_q$ be i.i.d. samples from D, and let $\textsc{Col}^q(D)$ be the event that $y_i = y_j$ for some $i, j \in [q]$. There is a constant $c > 2$ such that if $q \geq c\sqrt{\beta(D)}$, then $\Pr(\textsc{Col}^q(D)) \geq 2/3$.*

Proof. Let E be the event that $y_i = y_j$ for some $i, j \in [q]$. Then

$$\Pr[E] \geq 1 - \Pr[X_D > q] \geq 1 - e^{-\gamma_q}(1 + \gamma_q) \geq 2/3 \,,$$

when $q \geq c\sqrt{\beta(D)}$ because $\frac{1+\gamma_q}{e^{\gamma_q}} < 0.3$ whenever $\gamma_q = \frac{q-1}{\sqrt{\beta(D)}} > 2$.

We can also derive an upper bound on $\Pr[\textsc{Col}^q(D)]$ by standard approach.

Lemma 2. $\Pr[\textsc{Col}^q(D)] \leq \frac{q^2}{\beta(D)}$.

Proof. For any pair $i \in [q]$ and $j \in [q]$, Let $\textsc{Col}_{ij}$ be the event that $y_i = y_j$. Then $\Pr[\textsc{Col}_{ij}] = \mathrm{CP}(D)$. Therefore by union bound, we have

$$\Pr[\textsc{Col}^q(D)] = \Pr[\cup_{i,j\in[q]}\textsc{Col}_{ij}] \leq \binom{q}{2} \cdot \mathrm{CP}(D) \leq \frac{q^2}{\beta(D)} .$$

As a result, when $q = o(\sqrt{\beta(D)})$, essentially no collision will occur. Namely q needs to be $\Omega(\sqrt{\beta(D)})$ to see a collision, which is also sufficient by Corollary 1. This is summarized below as a birthday bound for general distributions.

Theorem 1. *$\Theta(\sqrt{\beta(D)})$ samples according to D are sufficient and necessary to produce a collision with constant probability for any classical algorithms.*

Finally, we characterize $\beta(D)$ for min-k distributions.

Lemma 3. *Let D_k be a min-k distribution on Y with $|Y| = N \geq 2^k$ and $k \geq 1$.*

- *For a flat-k distribution $D_{k,\flat}$, $\beta(D_{k,\flat}) = 2^k$.*
- *For δ-min-k distribution $D_{k,\delta}$, $\beta(D_{k,\delta}) \approx \begin{cases} N & \text{if } N < 2^{2k}; \\ 2^{2k} & \text{if } N \geq 2^{2k}. \end{cases}$*
- *For a general min-k distribution D_k, $\beta(D_k) \in [2^k, 2^{2k}]$.*

Proof. For flat-k D_k, $D_k(y) = \frac{1}{2^k}$ for all $y \in Y' \subseteq Y$ with $|Y'| = 2^k$. Hence $\beta(D_k) = \frac{1}{\sum_{y\in Y'} 2^{-2k}} = 2^k$. For $D_{k,\delta}$ distribution

$$\beta(D_{k,\delta}) = \frac{1}{\mathrm{CP}(D_{k,\delta})} = \frac{1}{2^{-2k} + \frac{(1-2^{-k})^2}{N-1}} = \frac{2^{2k}(N-1)}{N - 2\cdot 2^k + 2^{2k}} \approx \frac{2^{2k}\cdot N}{2^{2k}+N} .$$

Different ranges of N give the estimation for $\beta(D_{k,\delta})$. For general D_k, it is easy to see that $2^{-2k} \leq \mathrm{CP}(D_k) \leq 2^{-k}$ and hence $\beta(D_k) \in [2^k, 2^{2k}]$.

3 Lower bounds: finding a collision is difficult

We prove our quantum query lower bounds for min-k collision finding by security reductions. Recall the hardness result for uniform distributions by Zhandry [38].

Lemma 4 ([38] Theorem 3.1). *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 $C(q+1)^3/N$ for some universal constant C.*

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. Therefore we prove the following results.

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

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

Corollary 2. *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.*

Each of the proofs describe an algorithm (i.e., a reduction) attempting to find a collision in a random function f to which it has oracle access. The reduction will run as a subroutine another algorithm which finds a collision in another random function g when given oracle access to g (these random functions are not necessarily sampled from the same distribution). To adopt the subroutine which finds collisions in g for the task of finding a collision in f, the reduction simulates an oracle for g by building an oracle converter from the oracle for f and a suitable redistribution function. In general the redistribution function must be random, sampled from a particular distribution so that the distribution of its images equals that of g. Given some distributions from which the images of f and g are sampled, only some special sampling procedures will produce a redistribution function suitable for building the oracle converter needed. We formalize the concept of a *redistribution function sampler* as a generally randomized algorithm that performs such a sampling procedure specific to the oracles the reduction has access to and needs to simulate.

Definition 8 ($D \to D'$ Redistribution Function Sampler). *Suppose $f : X \to Y$ is a random function whose images are distributed according to a distribution D. Let D' be a distribution on Y'. We call an algorithm S a $D \to D'$* redistribution function sampler *if it returns a function $r : X \times Y \to Y'$ such that* $\Pr[r(x, f(x)) = y] = D'(y)$ *for all $y \in Y'$ and $x \in X$, where the probability is taken over the random choice of f and the randomness of S.*

We use the term *redistribution function* to refer to a function returned by a redistribution function sampler, explicitly stating the distributions when necessary. The redistribution function naturally induces an oracle converter.

Definition 9 (Oracle Converter). *Suppose $f \leftarrow D^X$ is a random function whose images are distributed according to a distribution D on Y. Let D' be a distribution on Y', and $r : X \times Y \to Y'$ be a $D \to D'$ redistribution function. An algorithm $\mathcal{C}$, having oracle access to f and r, is called an* oracle converter from f to g *if $\mathcal{C}$ computes a function $g : X \to Y'$ defined by $g(x) := r(x, f(x))$.*

We may denote $g = \mathcal{C}_{f,r}$. We can immediately observe that g is distributed as if the images were sampled independently according to D', when f and r are sampled according to the above definition.

Lemma 5. *The oracle converter defined above computes a function g that is distributed identically to ${D'}^X$, i.e., its images are independently distributed according to D', if $f \leftarrow D^X$ is chosen randomly and r is generated by a $D \to D'$ redistribution function sampler.*

We will be concerned with finding collisions in f and g. In particular, we are interested in whether a collision of g constitutes a collision of f. We define the collision-conversion rate to capture this property of an oracle converter.

Definition 10 (Collision-conversion rate). *Let $\mathcal{C}$ be an oracle converter from f to g. We say that it has* collision-conversion rate *p if for any (x, x') such that $g(x) = g(x')$, $f(x) = f(x')$ also holds with probability at least p. The probability is over the random choices of $f \leftarrow D^X$ and of a $D \to D'$ redistribution function r.*

With these notions available, our reduction proofs basically sample a proper redistribution function, and then simulate a correct oracle g distributed according to ${D'}^X$ using an oracle converter accessing the given oracle $f \sim D^X$. Then we run a collision-finding adversary on D' with oracle g. Whenever it outputs a collision, we can conclude that a collision is also found in f with probability p by the collision-conversion rate, which will lead to the desired contradiction. For each of the reductions, we will describe a suitable redistribution function sampler and show that it has at least constant collision-conversion rate. To do so, we assume that the reductions have full information about D and D', as well as sufficient randomness. This is fine as far as query complexity is concerned, and it is an interesting open question to make them time-efficient. We also remark that, for the sake of clarity, the distribution of images of our redistribution function is defined to be *exactly* matching distribution D'. It suffices to approximate distribution D' up to some negligible statistical distance.

Now we provide a generic formal description for all of our reductions, leaving the redistribution function sampler as a modular component which we can describe individually for each collision finding problem (for now we assume that each reduction has access to an adequate redistribution function sampler in each case). We do this in part to formally demonstrate how our reductions are compatible with *quantum* adversaries, allowing them to submit queries in quantum superposition and receive the oracle responses in quantum superposition. We will show that the oracle converters can be implemented as quantum oracles, so that the reduction can simulate the collision-finding problem for a quantum adversary who submit quantum queries. As usual, we consider a reduction solving collision-finding in D using an adversary for collision-finding in D'.

We emphasize that the functions f and r are random functions sampled before the adversary begins the attack (the *attack* referring to the query-response phase in which interaction with the oracle occurs), as f is simply a model for what would be a fixed, publicly known hash function in a practical security setting, and r would be chosen by the adversary according to some procedure specific to the hash function (this is the role played by redistribution function sampler). Implementing the converter as a quantum-accessible oracle is straightforward as

Algorithm 1 Generic reduction via oracle converter

```
Input: Let $f \leftarrow D^X$ be a random function whose images are sampled according to $D$
    on a set $Y$. Let $D'$ be a distribution on a set $Y'$. Let $S$ be a $D \to D'$ redistribution
    function sampler. Let $\mathcal{A}$ be an adversary for collision-finding in $D'$.
Output: A possible collision $(x_1, x_2)$ in $f$.
 1: Run $S$ and store its output as $r$. Implement an oracle for $r$.
 2: Construct an oracle converter $\mathcal{C}$ using the oracles for $f$ and $r$. The responses of $\mathcal{C}$
    are now distributed according to $D'$. Refer to the function implemented by $\mathcal{C}$ as $g$.
 3: Initialize $\mathcal{A}$. For each query made by $\mathcal{A}$, forward the query to $\mathcal{C}$ and return the
    response to $\mathcal{A}$.
 4: When $\mathcal{A}$ returns a collision $(x_1, x_2)$ in $g$, output $(x_1, x_2)$.
```

shown below (Fig. 1). Note that the function r can be turned into a unitary operator by standard technique $|x, \tilde{x}, y\rangle \overset{r}{\mapsto} |x, \tilde{x}, y \oplus r(x, \tilde{x})\rangle$. f is given as a quantum oracle, which we just need to query twice to answer each query to g.

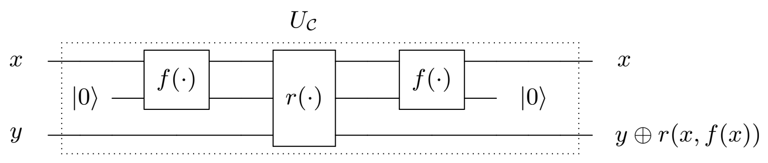


Fig. 1. Quantum circuit that implements function $g = \mathcal{C}_{f,r}$ using two oracle calls to f.

Now that we have a generic construction for our reductions, we will show a simple reusable general result that will allow us to quickly construct reductions and extend query complexity lower bounds by simply demonstrating the existence of a satisfactory redistribution function sampler for use in each reduction. In this context we say that an reduction algorithm *succeeds* if the output pair indeed forms a collision in the given oracle function.

Lemma 6. *Suppose there exists an algorithm $\mathcal{A}$ which solves collision finding in a distribution D' with probability at least $P_{\mathcal{A}}$, using q queries to an oracle for a function g whose responses are distributed according to D'* [4]*. Suppose there exists a $D \to D'$ redistribution function sampler S such that the induced converter has collision-conversion rate at least p. Then Algorithm 1 initialized with S and $\mathcal{A}$, denoted $\mathcal{R}_{S,A}$, solves collision finding in D with probability at least $p \cdot P_{\mathcal{A}}$ using $2q$ queries to an oracle for f whose images are distributed according to D.*

Proof. Lemma 6 follows trivially from the suppositions stated. Let A denote the event that $\mathcal{A}$ succeeds, E denote the event that the a collision of g is also a

[4] The probability $P_{\mathcal{A}}$ reflects the randomness of oracle's responses and of $\mathcal{A}$.

collision of f, and R denote the event that $\mathcal{R}_{S,A}$ suceeds. Then

$$\Pr[R] \geq \Pr[E \cap A] = \Pr[E|A] \cdot \Pr[A] = \Pr[E] \cdot \Pr[A]\,,$$

because E and A are independent ($\Pr[E]$ is simply the collision-conversion rate, which is a property specific to the oracle converter used in algorithm 1). Since $\Pr[E] \geq p$ and $\Pr[A] \geq P_{\mathcal{A}}$, $\Pr[R] \geq p \cdot P_{\mathcal{A}}$. The observation that $\mathcal{R}_{S,A}$ uses twice the number of oracle queries as $\mathcal{A}$ proves the lemma.

Therefore to prove Theorem 2 and 3, all that is left is to show suitable redistribution function samplers.

Lemma 7. *Let U_{2^k} be a uniform distribution on a set Y of size 2^k. Let $D_{k,\flat}$ be a flat-k distribution on a set Y_1, and D_k a general min-k distribution on a set Y_2. There exist $U_{2^k} \to D_{k,\flat}$ and $D_{k,\flat} \to D_k$ redistribution function samplers, and the induced oracle converters have collision-conversion rates at least 1/2.*

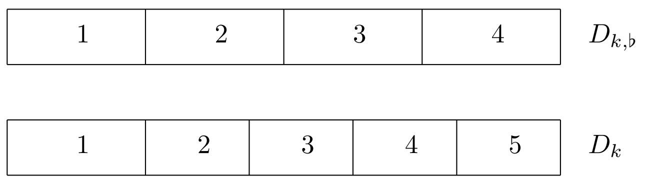
Proof. We describe the two samplers below.

$U_{2^k} \to D_{k,\flat}$ sampler. In this case the redistribution function sampler is nearly trivial because a simple relabeling of samples from the distribution U_{2^k} will suffice to simulate samples from the distribution $D_{k,\flat}$. Let f be a function $f : X \to Y$ whose images are distributed according to U_{2^k}, to which oracle access is available. Let $m : Y \to Y_1$ be any injective mapping. Define S_1 as a one-step algorithm that returns a function $r_1(x, y) = m(y)$.

By the definition of r_1, $\Pr[r_1(x, f(x)) = y'] = \Pr[m(f(x)) = y']$ for all $x \in X$ and $y' \in Y_1$. Since m implements an injective mapping from Y to Y_1, $\Pr[m(f(x)) = y'] = \Pr[f(x) = m^{-1}(y')]$. Since, by the definition of f, $\Pr[f(x) = y] = U_{2^k}(y)$ for all $y \in Y$, $\Pr[f(x) = m^{-1}(y')] = U_{2^k}(m^{-1}(y')) = 2^{-k}$. Hence $\Pr[r_1(x, f(x)) = y'] = D_{k,\flat}(y')$ for all $x \in X$ and $y' \in Y_1$, since $D_{k,\flat}(y') = 2^{-k}$ for all $y' \in Y_1$. It follows that S_1 is a $U_{2^k} \to D_{k,\flat}$ redistribution function sampler. We now show that the collision-conversion rate of the induced oracle converter is exactly 1. Let (x_1, x_2) be a collision in g, the function implemented by the oracle converter. Then $r_1(x_1, f(x_1)) = r_1(x_2, f(x_2))$, from which it follows that $m(f(x_1)) = m(f(x_2))$. Since m is an injective mapping,we can conclude that $f(x_1) = f(x_2)$, which shows that (x_1, x_2) is necessarily a collision in f.

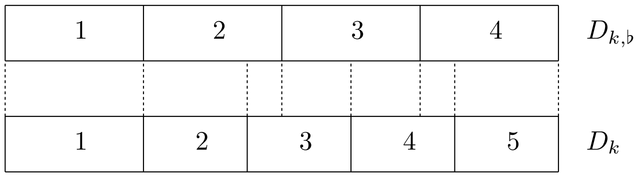
$D_{k,\flat} \to D_k$ sampler. We provide an overview of the $D_{k,\flat} \to D_k$ redistribution function sampler in the following few paragraphs. The complete redistribution function sampler is given in appendix A, along with a detailed explanation of the reasoning behind it. We reiterate that the redistribution function can be prepared before oracle access to the hash function under attack is obtained, allowing the query-response phase of the attack to be implemented as a quantum algorithm without concern for the quantum implementation of the redistribution function sampler.

The basic challenge that must be solved by the redistribution function sampler is to provide a mapping from the support of one distribution to the support of another distribution in such a way that the output is actually distributed according to the second distribution, which we call D_k, when the input is distributed according to the first, which we call $D_{k,\flat}$.[5] In order to maximize the probability that Algorithm 1 succeeds, the mapping must maximize the probability that two identical outputs correspond with two identical inputs, i.e. the collision-conversion rate. Our construction for this redistribution function sampler, which we call S_2 (and which returns a function which we call r_2), ensures that this probability is no less than one half by allowing at most two elements of the support of the $D_{k,\flat}$ be mapped to each element of the support of D_k. To provide intuition for how this is achieved, we recommend visualizing each distribution as a rectangle divided into 'bins' representing the elements of its support, with each bin's width proportional to the probability mass of the corresponding element under the distribution. We refer to this as the *rectangular representation* of the distribution. An example is shown below. We let $D_{k,\flat}$ be a flat distribution of min-entropy 2, and D_k be a (non-flat) distribution of min-entropy 2. We label each bin with a number indexing the elements of the support in each case.



For each of the elements of the support of $D_{k,\flat}$, we must decide what the probability mass corresponding to that element in $D_{k,\flat}$ should 'be sent to' by the redistribution function, in the sense that whatever that element is mapped to will occur with the same probability as that of sampling the element from $D_{k,\flat}$. A natural solution that would correctly produce the distribution D_k is to in some sense 'project' the distribution $D_{k,\flat}$ onto D_k, so that each 'location' in the rectangular representation of $D_{k,\flat}$ is mapped to a 'location' in the rectangular representation of D_k (by 'location' here we refer to horizontal position a rectangular representation, selecting some specific probability density). We illustrate this sort of projection by drawings lines between the two rectangular representations that show where the boundaries between the elements of each distribution's support fall in the other distribution, shown below.

[5] A redistribution function formally is also provided the query x that is associated with the sample from the first distribution, which is (in Algorithm 1) the response from an oracle whose output is distributed according to the first distribution. This is necessary in cases where the second distribution has a larger support than the first, since the image of the redistribution function cannot be larger than the domain. It can safely be ignored otherwise (as in the construction for r_1).



From the fact that the width of each bin is proportional to the probability mass associated with each element of each distribution, it follows that, if, for a given sample from $D_{k,\flat}$, we sample an element from the support of D_k according to the available probability mass *inside* the projected bin from $D_{k,\flat}$, the sampling result will be distributed exactly according to the distribution D_k. This is difficult to communicate verbally, but visually, one can imagine receiving a sample from $D_{k,\flat}$ as it 'selecting' the bin associated with the sampled value in the rectangular representation of the distribution. Then, following the lines bordering that bin, we find that the probability mass associated with the sample from $D_{k,\flat}$ is mapped to probability mass corresponding to several elements of the support of distribution D_k. If we now sample from these elements according to their share of the probability mass corresponding to the sample from $D_{k,\flat}$, our samples will be distributed according to D_k. For example, with reference specifically to the graphic above, suppose that we receive element 2 as a sample from $D_{k,\flat}$. Following the lines down from the bin corresponding to element 2 in the rectangular representation of $D_{k,\flat}$, we see that elements 2 and 3 in the support of D_k both partially reside in the space corresponding to bin 2 in the rectangular representation of $D_{k,\flat}$. In particular, element 2 in the support of D_k consumes much more of the space than element 3. Hence we sample either 2 or 3, with a bias toward 2 exactly equal to how much more of the space element 2 consumes (recall that *space* in these rectangular representations corresponds to probability mass). Similarly, had we received element 3 as a sample from $D_{k,\flat}$, we would have sampled from elements 3 and 4 in the support of D_k with little or no bias, since these seem to roughly evenly split the space inside the boundaries of the bin corresponding to element 3 in the support of $D_{k,\flat}$.

It should be clear now that this procedure will produce samples distributed according to D_k when given samples distributed according to $D_{k,\flat}$, at the cost of needing additional randomness to perform the sub-sampling. Generating the redistribution function r_2 is then simply a matter of saving the resulting samples in a look-up table. Although this procedure is conceptually simple, its rigorous mathematical description is exceedingly tedious, so we provide it in the appendix. Also in the appendix is a proof that the redistribution function sampler S_2 has a collision-conversion rate of at least one-half. The intuition behind this property is that a sample from D_k produced by the redistribution function could have been generated by, at most, 2 distinct samples from $D_{k,\flat}$, since each bin in the rectangular representation of D_k resides within the boundaries of, at most, 2 bins in the rectangular representation of $D_{k,\flat}$.

We have shown that S_1 and S_2, as just described (and formally described in the appendix in the case of S_2), are $U_{2^k} \to D_{k,\flat}$ and $D_{k,\flat} \to D_k$ redistribution function samplers, respectively. Finally, Theorem 2 and 3 follow easily. Note that we write some of the constant factors in the probabilities with the common notation 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.

Proof (Proof of Theorem 2 & Theorem 3). By Lemma 7, there exists a $U_{2^k} \to D_{k,\flat}$ redistribution function sampler S_1 for which the induced collision-conversion rate is at least one-half. Therefore Lemma 6 implies that our reduction algorithm is an collision-finding adversary making $2q$ queries to a uniformly random function f with success probability at least $P_{\mathcal{A}}/2$. However, Lemma 4 tells us that any $2q$-query adversary can succeed with probability at most $C(2q+1)^3/2^k$. Therefore the success probability $P_{\mathcal{A}}$ of any q-query adversary $\mathcal{A}$ is $O(q+1)^3/2^k$, which proves Theorem 2.

Theorem 3 is proved in the same fashion by invoking the $D_{k,\flat} \to D_k$ redistribution function sampler S_2 in Lemma 7.

3.1 Stronger lower bound for δ-min-k distributions

Note that following the same strategy, one can show a reduction of collision finding in an arbitrary min-k distribution D 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. Clearly, if no elements in the support of D are associated with a probability mass less than $1/N$, the proof of Theorem 3 can be adapted by 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. The general case where D has elements associated with smaller probability mass than $1/N$ may be resolved by considering the distribution removing these elements and showing that it is computationally indistinguishable from the original.

In this section we give further evidence and establish an even stronger bound for finding collision in the δ-k-distribution case.

Theorem 4. *For any q-query algorithm A,*

$$\Pr_{f \leftarrow D_{k,\delta}{}^X}[f(x) = f(x') : (x, x') \leftarrow A^f(\cdot)] \le O\left(\frac{(q+2)^2}{2^k} + \frac{(q+2)^3}{N}\right).$$

We give two proofs. The one presented here relies on a technique by Zhandry (Lemma 8). We give an alternative proof in Section C based on a reduction from an average version of a search problem which is hard to solve from the literature. This may serve as an intuitive explanation of the hardness of non-uniform collision finding. It also connects to the quantum algorithm we develop in Sect. 4.1 based on Grover's search algorithm.

Lemma 8. *[37, Theorem 7.2] Fix q, and let F_λ be a family of distributions on Y^X indexed by $\lambda \in [0,1]$. Suppose there is an integer d such that for every $2q$ pairs $(x_i, y_i) \in X \times Y$, the function $p_\lambda := \Pr_{f \leftarrow F_\lambda}(f(x_i) = y_i, \forall i \in \{1, \ldots, 2q\})$ is a polynomial of degree at most d in λ. Then any quantum algorithm A making q queries can only distinguish F_λ from F_0 with probability at most $2\lambda d^2$.*

This lemma enables us to prove the following proposition.

Proposition 1. *For any q-query algorithm A,*

$$\left| \Pr_{f \leftarrow {D_{k,\delta}}^X}(A^f(\cdot) = 1) - \Pr_{f \leftarrow Y^X}(A^f(\cdot) = 1) \right| \leq 8q^2/2^k + 1/N\,.$$

Proof. For every $\lambda \in [0,1]$, define D_λ on Y such that there is an element $m \in Y$ with $D_\lambda(m) = \lambda$ and for any $y \neq m$ $D_\lambda(y) = \frac{1-\lambda}{|Y|-1}$. Then define a family of distributions F_λ on Y^X where $F_\lambda := {D_\lambda}^X$, i.e., the output of each input is chosen independently according to D_λ.

For any $\{(x_i, y_i)\}_{i=1}^{2q}$, $p_\lambda := \Pr_{f \leftarrow F_\lambda}(f(x_i) = y_i, \forall i \in [2q]) = \lambda^t (\frac{1-\lambda}{|Y|-1})^{2q-t}$, where t is the number of occurrences of m in $\{y_i\}_{i=1}^{2q}$. Clearly p_λ is a polynomial in λ with degree at most $2q$.

Notice that $F_{2^{-k}}$ is exactly δ-min-k distribution $D_{k,\delta}$, and F_0 is uniformly random on $\hat{Y}^X$, where $\hat{Y} := Y \backslash \{m\}$. Therefore by Lemma 8,

$$\left| \Pr_{f \leftarrow {D_{k,\delta}}^X}(A^f(\cdot) = 1) - \Pr_{f \leftarrow \hat{Y}^X}(A^f(\cdot) = 1) \right| \leq 2(2q)^2 \cdot 2^{-k} = 8q^2/2^k\,.$$

Since Y^X and $\hat{Y}^X$ has statistical distance $\frac{1}{2}(N-1)(\frac{1}{N-1} - \frac{1}{N}) + \frac{1}{2}(\frac{1}{N} - 0) = 1/N$, we get that $\left|\Pr_{f \leftarrow {D_{k,\delta}}^X}(A^f(\cdot) = 1) - \Pr_{f \leftarrow Y^X}(A^f(\cdot) = 1)\right| \leq 8q^2/2^k + 1/N$.

We are now ready to prove the stronger complexity for finding collision in a δ-min-k random function.

Proof (Proof of Theorem 4). Suppose that there is an A with

$$\Pr_{f \leftarrow {D_{k,\delta}}^X}[f(x) = f(x') : (x, x') \leftarrow A^f(\cdot)] = \varepsilon$$

using q queries. Then construct A' which on input oracle f, runs A and receives (x, x') from A. A' then output 1 iff. $f(x) = f(x')$. By definition, we have that $\Pr_{f \leftarrow {D_{k,\delta}}^X}(A'^f(\cdot) = 1) = \varepsilon$. Meanwhile, note that A' makes $q+2$ queries. Therefore by Zhandry's lower bound on finding collision in uniform random function (Lemma 4), we know that $\Pr_{f \leftarrow Y^X}(A'^f(\cdot) = 1) \leq O(\frac{(q+3)^3}{N})$. Then Proposition 1 implies that

$$\varepsilon \leq O(\frac{(q+3)^3}{N}) + 8(q+2)^2/2^k + 1/N = O(\frac{(q+2)^2}{2^k} + \frac{(q+3)^3}{N})\,.$$

Corollary 3. *Any quantum algorithm needs $\min\{2^{k/2}, N^{1/3}\}$ queries to find a collision with constant probability. Specifically we need $\Omega(N^{1/3})$ if $2^k \leq N < 2^{\frac{3k}{2}}$, and $\Omega(2^{k/2})$ when $N \geq 2^{\frac{3k}{2}}$.*

4 Upper bounds: (optimal) quantum algorithms

We derive a generic upper bound for finding collision in any min-k random functions. We adapt Ambainis's algorithm (Lemma 9) and describe a quantum algorithm NU-ColF (Algorithm 2).

Lemma 9. *([8, Theorem 3]) Let $f : X' \to Y$ be a function that has at least one collision. Then there is a quantum algorithm ColF making $O(|X'|^{2/3})$ quantum queries to f that finds the collision with constant bounded error.*

Algorithm 2 Collision Finding in Non-uniform Function NU-ColF

```
Input: $f \leftarrow {D_k}^X$ as an oracle. Let $s, t$ be parameters to be specified later.
Output: Collision $(x, x')$ or $\perp$.
1: Divide $X$ in to subsets $X_i$ of equal size (ignoring the boundary case) $|X_i| = s$.
2: Construct $f_i : X_i \to Y$ as the restriction of $f$ on $X_i$.
3: For $i = 1, \dots, t$, Run Ambainis's algorithm ColF on $f_i$, and get candidate collision
   $x_i$ and $x'_i$. if $f(x_i) = f(x'_i)$, output $(x_i, x'_i)$ and abort.
4: Output $\perp$.
```

Theorem 5. *Let $\beta := \beta(D_k)$. Let X be a set with $|X| = M = \Omega(\sqrt{\beta})$. Algorithm 2 NU-ColF finds a collision in $f \leftarrow X^{D_k}$ within $O(\beta^{1/3})$ queries with constant probability. Moreover with $O(k\beta^{1/3})$ queries the algorithm succeeds except with probability negligible in k.*

Proof. Since f is generated according to the min-k distribution, when restricting to any subset X_i, we can think of drawing each function value independently from D_k. Namely $f_i \sim {D_k}^{X_i}$ holds for all i. Therefore, by Lemma 1, we have that when $s \geq c\sqrt{\beta(D)}$ for some $c > 2$, f_i contains a collision with constant probability. If that is the case, Ambainis's algorithm will find a collision with constant probability using $O(|X_i|^{2/3}) = O(\beta(D)^{1/3})$ queries. We only need to repeat $t = O(k)$ times to succeed except with error negligible in k.

Note that our algorithm NU-ColF is generic, and needs no additional information about D_k. By our characterization of $\beta(D_k)$ in Lemma 3, we obtain specific bounds for the two special distributions.

Corollary 4. *There exists a quantum algorithm that finds a collision with constant probability using the following numbers of queries:*

- *flat-k: $O(\beta^{1/3}) = O(2^{k/3})$ and it is tight when $M = \Omega(2^{k/2})$.*
- *δ-min-k: $O(\beta^{1/3}) = \begin{cases} O(N^{1/3}) \; 2^k \leq N < 2^{2k}, \text{ tight when } N \leq 2^{3k/2} \\ O(2^{\frac{2k}{3}}) \;\; N \geq 2^{2k} \end{cases}$*

4.1 Quantum algorithm for min-k distribution with a mode known

We design an alternative collision finding algorithm (Algorithm 3), which performs slightly better in some settings. It is based on a version of Grover's algorithm [22,15,29] for multiple marked items stated below.

Lemma 10. *Let $f : X \to \{0,1\}$ be an oracle function and let $Z_f = |\{x \in X : f(x) = 1\}|$. Then there is a quantum algorithm* QSEARCH *using q queries that finds an $x \in X$ such that $f(x) = 1$ with success probability $\Omega(q^2 \frac{Z_f}{|X|})$.*

Algorithm 3 Collision Finding in Non-uniform Function **with a mode known** NU-ColF-Mode

Input: $f \leftarrow {D_k}^X$ as an oracle. A mode element m of D_k.
Output: Collision (x, x') or $\perp$.

```
1: Run Grover's algorithm QSEARCH on $f$ to find $x$ with $f(x) = m$.
2: Run Grover's algorithm QSEARCH on $f$ to find $x'$ with $f(x) = m$ and $x' \neq x$.
3: Output $\perp$ if any run of the Grover's algorithm failed. Otherwise output $(x, x')$.
```

Theorem 6. *NU-ColF-Mode finds a collision using $O(2^{k/2})$ queries with constant probability.*

Proof. Let $Z_f := |f^{-1}(m)|$. Let p_f be the probability that f is chosen, when drawn from ${D_k}^X$. Since we invoke QSEARCH twice, we find (x, x') with probability $\Omega\left((\frac{q^2 Z_f}{|X|})^2\right)$. Then algorithm NU-ColF-Mode succeeds with probability

$$\sum_f p_f \Omega\left(\frac{q^4}{M^2} Z_f^2\right) = \Omega\left(\frac{q^4}{M^2} \sum_f p_f Z_f^2\right) = \Omega(\frac{q^4}{M^2}\mathbb{E}[Z_f^2])\,.$$

To compute $\mathbb{E}[Z_f^2]$, we define for every $x \in X$ an indicator variable $Z_x = \begin{cases} 1 & \text{if } f(x) = m; \\ 0 & \text{otherwise.} \end{cases}$, where $f \leftarrow {D_k}^X$, and clearly $Z_f = \sum_{x\in X} Z_x$. Since each output of x is drawn independently according to $D_{k,\delta}$, $\mathbb{E}[Z_x] = \varepsilon := 2^{-k}$ for all x, it follows that $\mathbb{E}[Z_x] = \mathbb{E}[Z_x^2] = \varepsilon$, and $\mathbb{E}[Z_x \cdot Z_{x'}] = \mathbb{E}[Z_x] \cdot \mathbb{E}[Z_{x'}] = \varepsilon^2$ for any $x \neq x'$ by independence. Therefore

$$\mathbb{E}[Z_f^2] = \sum_x \mathbb{E}[Z_x^2] + \sum_{x \neq x'} \mathbb{E}[Z_x Z_{x'}] = \Omega(M^2 \varepsilon^2)\,.$$

Hence the algorithm succeeds with probability $\Omega(q^4 \varepsilon^2) = \Omega((\frac{q^2}{2^k})^2)$. As a result, with $q = O(2^{k/2})$ many queries, we find a collision with constant probability.

Remark 1. Note that we still need $M = \Omega(\sqrt{\beta(D)})$ to ensure existence of collisions. When $N \geq 2^{3k/2}$, Theorem 6 gives a better bound $(2^{k/2})$ than Theorem 5 ($N^{1/3}$ when $2^{3k/2} \leq N < 2^{2k}$ and $2^{2k/3}$ when $N \geq 2^{2k}$).

References

1. Password hashing competition (2012), https://password-hashing.net/
2. National Institute of Standards and Technology. SHA-3 standard: Permutation-based hash and extendable-output functions (2014), available at http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf
3. IBM Q quantum experience (2017), https://www.research.ibm.com/ibm-q/
4. National Institute of Standards and Technology. FIPS 180-1: Secure hash standard (April 1995)
5. People of ACM - John Martinis (May 16, 2017), https://www.acm.org/articles/people-of-acm/2017/john-martinis
6. Aaronson, S., Shi, Y.: Quantum lower bounds for the collision and the element distinctness problems. Journal of the ACM (JACM) 51(4), 595–605 (2004)
7. Ambainis, A.: Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. Theory of Computing 1(3), 37–46 (2005), http://www.theoryofcomputing.org/articles/v001a003
8. Ambainis, A.: Quantum walk algorithm for element distinctness. SIAM Journal on Computing 37(1), 210–239 (2007), preliminary version in FOCS 2004. Available at arXiv:quant-ph/0311001.
9. Ambainis, A., Rosmanis, A., Unruh, D.: Quantum attacks on classical proof systems (the hardness of quantum rewinding). In: FOCS 2014. pp. 474–483. IEEE (October 2014), preprint on IACR ePrint 2014/296
10. Amy, M., Di Matteo, O., Gheorghiu, V., Mosca, M., Parent, A., Schanck, J.: Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. arXiv preprint arXiv:1603.09383 (2016)
11. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on Computer and Communications Security. pp. 62–73. ACM (1993)
12. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: Advances in Cryptology–EUROCRYPT 1994. pp. 92–111. Springer (1994)
13. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak sponge function family (2007), http://keccak.noekeon.org/
14. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Advances in Cryptology – ASIACRYPT 2011. pp. 41–69. Springer (2011)
15. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. arXiv preprint quant-ph/9605034 (1996)
16. Brassard, G., Hoyer, P., Tapp, A.: Quantum algorithm for the collision problem. arXiv preprint quant-ph/9705002 (1997)
17. Crépeau, C., Salvail, L., Simard, J.R., Tapp, A.: Two provers in isolation. In: Advances in Cryptology–ASIACRYPT 2011, pp. 407–430. Springer (2011)
18. Czajkowski, J., Bruinderink, L.G., Hülsing, A., Schaffner, C., Unruh, D.: Post-quantum security of the sponge construction. Cryptology ePrint Archive, Report 2017/771 (2017), https://eprint.iacr.org/2017/771
19. Eaton, E., Song, F.: Making existential-unforgeable signatures strongly unforgeable in the quantum random-oracle model. In: 10th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC'15. LIPIcs, vol. 44, pp. 147–162. Schloss Dagstuhl (2015)
20. Ebrahimi, E., Unruh, D.: Quantum collision-resistance of non-uniformly distributed functions: Upper and lower bounds. Cryptology ePrint Archive, Report 2017/575 (2017)

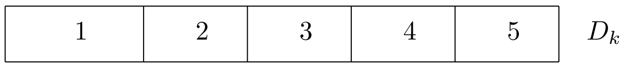
21. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology 26(1), 80–101 (2013), preliminary version in CRYPTO 1999
22. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219. ACM (1996)
23. Hallgren, S., Smith, A., Song, F.: Classical cryptographic protocols in a quantum world. In: Advances in Cryptology–CRYPTO 2011. pp. 411–428. Springer (2011)
24. Hülsing, A., Rijneveld, J., Song, F.: Mitigating multi-target attacks in hash-based signatures. In: Public-Key Cryptography – PKC 2016, pp. 387–416. Springer (2016)
25. Jain, R., Kolla, A., Midrijanis, G., Reichardt, B.W.: On parallel composition of zero-knowledge proofs with black-box quantum simulators. Quantum Information & Computation 9(5), 513–532 (2009), available at arXiv:quant-ph/0607211
26. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), https://bitcoin.org/bitcoin.pdf
27. Rivest, R.L.: RFC 1321: The MD5 message-digest algorithm (April 1992), https://www.ietf.org/rfc/rfc1321.txt
28. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (1997)
29. Song, F.: Early days following Grover's quantum search algorithm. arXiv preprint arXiv:1709.01236 (2017)
30. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. Cryptology ePrint Archive, Report 2017/190 (2017), https://shattered.io/
31. Targhi, E.E., Tabia, G.N., Unruh, D.: Quantum collision-resistance of non-uniformly distributed functions. In: International Workshop on Post-Quantum Cryptography. pp. 79–85. Springer (2016)
32. Targhi, E.E., Unruh, D.: Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In: Theory of Cryptography Conference. pp. 192–216. Springer (2016)
33. Unruh, D.: Computationally binding quantum commitments. In: Advances in Cryptology–EUROCRYPT 2016. pp. 497–527. Springer (2016)
34. Watrous, J.: Zero-knowledge against quantum attacks. SIAM J. Comput. 39(1), 25–58 (2009), preliminary version in STOC 2006
35. Wiener, M.J.: Bounds on birthday attack times. Cryptology ePrint Archive, Report 2005/318 (2005), http://eprint.iacr.org/2005/318
36. Yuen, H.: A quantum lower bound for distinguishing random functions from random permutations. Quantum Information & Computation 14(13-14), 1089–1097 (2014)
37. Zhandry, M.: How to construct quantum random functions. In: FOCS 2012. pp. 679–687. IEEE (2012), full version at http://eprint.iacr.org/2012/182
38. Zhandry, M.: A note on the quantum collision and set equality problems. Quantum Information and Computation 15(7 & 8) (2015)
39. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. International Journal of Quantum Information 13(4) (2015), early version in Crypto 2012. Full version at http://eprint.iacr.org/2012/076

A Details relating to Lemma 7: $D_{k,\flat} \to D_k$ redistribution function sampler

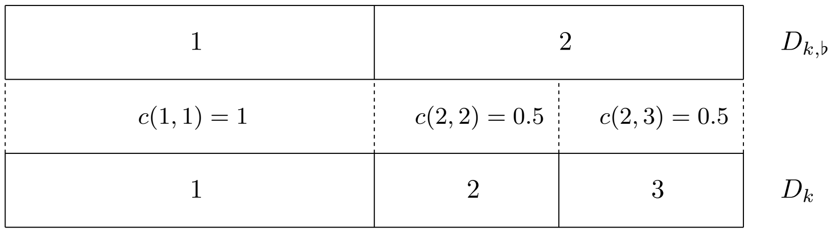
Algorithm 4 Redistribution Function Sampler S_2

```
Input: Let $D_k$ be an arbitrary $k$-distribution on support $S_k$. Let $D_{k,\flat}$ be a flat-$k$-distribution on support $S_{k,\flat}$. Let $f_\flat : X \to S_{k,\flat}$ be an oracle for a function whose images are distributed according to $D_{k,\flat}$
1: Prepare to store a lookup table for a function $c : S_k \times S_{k,\flat} \to [0,1]$
2: Sort and label the elements $y_i$ of $S_k$ in order of decreasing probability mass under distribution $D_k$, so that $D_k(y_1) = 2^{-k}$ (by the definition of min-entropy, there must be one or more $y_i \in S_k$ with $D_k(y_i) = 2^{-k}$)
3: Arbitrarily label the elements $z_j$ of $S_{k,\flat}$ with index $j = 1, 2, \ldots, 2^k$.
4: Iterate the following over $i = 1, 2, \ldots, |S_k|$:
     – If $D_k(y_i) = 2^{-k}$, set $c(y_i, z_i) = 1$. Set $c(y_i, z_j) = 0$ for all $j \neq i$.
     – If $D_k(y_i) < 2^{-k}$, compute $\sum_{j=1}^{i-1} D_k(y_j)$ (here $j$ is a dummy-index for the sum and is unrelated to the labeling of the elements of $S_{k,\flat}$) and save the result as the image of $i$ under a function $g : [|S_k|] \to [0,1]$. Check if $2^{-k}$ divides $g(i)$:
         • If so, set $c(y_i, z_{((g(i)/2^{-k})+1)}) = 2^k D_k(y_i)$ and $c(y_i, z_j) = 0$ for all $j \neq (g(i)/2^{-k}) + 1$.
         • If not, set $c(y_i, z_{\lceil g(i)/2^{-k} \rceil}) = 2^k \min(\lceil g(i)/2^{-k} \rceil - g(i), D_k(y_i))$, set $c(y_i, z_{\lceil g(i)/2^{-k} \rceil + 1}) = 2^k (D_k(y_i) - \min(\lceil g(i)/2^{-k} \rceil - g(i), D_k(y_i)))$, and set $c(y_i, z_j) = 0$ for all remaining $z_j \in S_{k,\flat}$
5: For each $z_j \in S_{k,\flat}$, construct the set $W_j$ containing all $y_i \in S_k$ for which $c(y_i, z_j) \neq 0$. Store the set $W_j$ as the image of $z_j$ in a function $b : S_{k,\flat} \to \mathcal{P}(S_k)$.
6: For each combination of one $x \in X$ and one $z \in S_{k,\flat}$, repeat the following (we will gradually store a lookup table for $r_2$):
     – Fix some mapping from an index $t$ to each element in $b(z)$. Compute $m_x(y_t) = c(y_t, z)$ for each $y_t \in b(z)$. Sample $a \leftarrow [0,1]$.
     – Iterate through $i = 1, \ldots, |b(z)|$ until $i$ has a value such that the following condition holds: $\sum_{t=1}^{i} m_x(y_t) \geq a$. Define $r_2(x, z) = y_i$.
7: Return $r_2$.
```

To facilitate intuitive understanding of why S_2 is a suitable redistribution function sampler for use in reducing collision finding in $D_{k,\flat}$ to collision finding in D_k, we visualize a distribution D_k as a rectangle divided into disjoint regions, each region representing one element of the support S_k. The total area of the rectangle is 1, representing the total probability mass in D_k. 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 oracle converter (which we denote $\mathcal{C}$ as f_k. In order to be used in Algorithm 1, the redistribution function's output must be distributed according the distribution D_k when given a query $x \in X$ and a response from an oracle for $f_\flat$. Clearly, in order to satisfy this requirement, there must be additional randomness 'built in' to the way the redistribution function is sampled because the distribution D_k may, in general, have higher Shannon entropy than the distribution $D_{k,\flat}$, as $|S_k| \geq |S_{k,\flat}|$. We can accomplish this by treating the elements of $S_{k,\flat}$ as 'bins', each associated with one or more elements of S_k. In order to generate a sample from S_k, a sample from $S_{k,\flat}$ first selects a 'bin', and then one of the elements of S_k associated with that bin is chosen randomly according to a conditional probability distribution such that the marginal probability of sampling that element (across all of the bins) is equal to the probability associated with that element under distribution D_k. This process can be visualized intuitively by vertically aligning rectangular representations of the distributions $D_{k,\flat}$ and D_k. The conditional probability of sampling each element in S_k given a bin in $S_{k,\flat}$ can be illustrated by projecting the dividers between elements in the rectangle representing D_k into the space between the two rectangles. We show a trivial example below, using distributions of min-entropy 1.

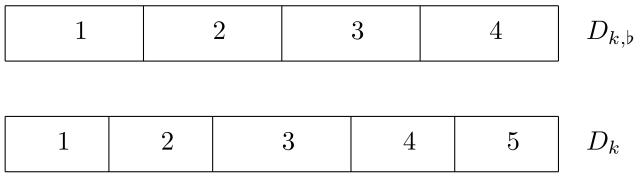


The diagram above also illustrates the role played by the function $c : S_k \times S_{k,\flat} \to [0,1]$. This function specifies the conditional probability that r_2 returns a sample of a certain element of S_k given that the response of $f_\flat$ to a certain query is a certain element from $S_{k,\flat}$. Formally, $c(y,z) = \Pr[r_2(x, f_\flat(x)) = y | f_\flat(x) = z]$, where x is a query produced by the adversary in algorithm 1. Then the output of r_2 on input (x,z) is simply determined by sampling from the chosen bin according to the conditional probabilities that $c(y,z)$ specifies—this sampling step is where algorithm 4 injects the additional randomness that will be needed to convert an oracle for $f_\flat$ to an oracle for f_k. The values encoded into the function c therefore make up the non-trivial part of algorithm 4.

In the example above, the elements 2 and 3 have a total probability mass of 0.5 in D_k, so they can be 'binned' into element 2 in $D_{k,\flat}$. Supposing that element

2 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_k, since each are equally likely under D_k. 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_k must exactly equal the probability mass of any of the elements of $D_{k,\flat}$, since D_k and $D_{k,\flat}$ have equal min-entropy.

This procedure will ensure that the output of r_2 is distributed according to D_k when the 2nd input is distributed according to $D_{k,\flat}$, since the marginal probability of sampling each element in this fashion exactly replicates the associated probability in D_k. In this simple case, any collision found in f_k will necessarily be a collision in $f_\flat$. However, this is not true in general. If the elements of S_k cannot be grouped into bins each with total probability mass under D_k equal to 2^{-k}, then some elements of S_k must have their probability mass split among multiple bins. An example of such a case is shown below.

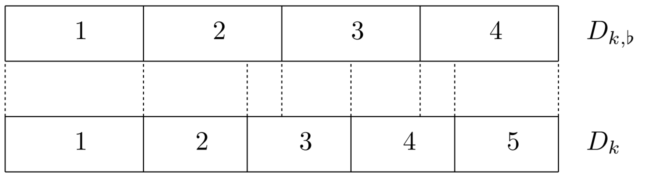


In cases like these, it is possible that a pair of identical outputs from r_2, constituting an apparent collision in f_k from the perspective of the adversary in algorithm 1, do not actually originate from identical responses from $f_\flat$, and therefore do not constitute an actual collision in $f_\flat$, the function which 1 attacks. Luckily, it is possible to construct the function c such that the probability that a collision in f_k corresponds to a collision in $f_\flat$ is bounded below by one half (giving $\mathcal{C}$ a collision-conversion rate of at least one half), so that Algorithm 1 instantiated with S_2 can preserve the query complexity of the adversary it leverages, from which Theorem 3 follows. Algorithm 4 contains a general method for constructing such a function, which we explain now.

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

Next, Algorithm 4 iterates over the elements y_i of S_k, setting the value of $c(y_i, z_j)$ for all elements z_j of $S_{k,\flat}$ for each. This may be visualized as moving across the rectangular representations of D_k and $D_{k,\flat}$ from left to right, determining the values of the function c along the way. If the probability mass of y_i under D_k is 2^{-k}, then all of its probability mass is associated with the element in $S_{k,\flat}$ 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_k is less than 2^{-k}, then the probability mass from y_i will not 'occupy' an entire bin in $D_{k,\flat}$, so it must share a bin with other elements of S_k. 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 $S_{k,\flat}$ which, at this point in execution of Algorithm 4, 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, Algorithm 4 computes the total probability mass in S_k 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 $\lceil g(i)/2^{-k} \rceil$, because for example 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 $D_{k,\flat}$, 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. 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_k 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.



$c(1,1) = 1$
$c(2,2) = 0.75$
$c(3,2) = 0.25$
$c(3,3) = 0.5$
$c(4,3) = 0.5$
$c(4,4) = 0.25$
$c(5,4) = 0.75$

In the next step, for each element in $S_{k,\flat}$, Algorithm 4 saves the set of all elements in S_k 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_k whose responses are distributed according to D_k.

Now that the appropriate conditional probability distributions are calculated (in the c function), all Algorithm 4 has to do is sample from the conditional distribution specified by c for each possible combination of a query $x \in X$ and a response $z \in S_{k,\flat}$ of oracle $f_\flat$. Essentially, the steps before this point were defining the distribution over the set of all functions needed to sample a suitable redistribution function, and the remaining steps sample the function from this distribution, one image at a time. It is simple to verify that the last few steps in Algorithm 4 do exactly this via inverse transform sampling.

B Proof that the oracle converter induced by S_2 has a collision-conversion rate of at least $\frac{1}{2}$

Let $D_{k,\flat}$ denote the flat-k-distribution on support $S_{k,\flat}$, and D_k denote a k-distribution on support S_k. Let $f_\flat$ be a function $f_\flat : X \to S_{k,\flat}$ whose images are distributed according to $D_{k,\flat}$, to which oracle access is available. Let $r_2 : X \times S_{k,\flat} \to S_k$ be defined as it is in algorithm 4. Let $f_k : X \to S_k$ denote the oracle simulated by the oracle converter induced by S_2. Let y denote $f_k(x_1)$, this oracle's response to query x_1, and likewise let y' denote $f_k(x_2)$. Let z denote $f_\flat(x_1)$ and z' denote $f_\flat(x_2)$. Then the oracle converter's collision-conversion rate can be expressed as $\Pr[z = z'|y = y']$. To help prevent confusion, we stress that the probability here is taken over the random choice of $f_\flat \leftarrow {D_{k,\flat}}^X$ and the randomness in 4 when preparing r_2.

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

$$\begin{aligned}
\Pr[z = z'] &= \sum_{j=1}^{2^k} \Pr[z' = z_j|z = z_j] \cdot \Pr[z = z_j] \\
&= \sum_{j=1}^{2^k} \Pr[z' = z_j] \cdot \Pr[z = z_j] \quad (z \;\&\; z' \;\textit{independent}) \\
&= \sum_{j=1}^{2^k} (2^{-k})^2 = 2^{-k} \quad (\text{definition of a flat-}k\text{-distribution})\,.
\end{aligned}$$

Following the same reasoning just used other than the final step, we may also write $\Pr[y = y'] = \sum_{i=1}^{|S_k|}(D_k(y_i))^2$.

Then equation (1) can be rewritten as

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

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 $D_{k,\flat}$ have probability 2^{-k}, we can see that the ratio in the above equation must be exactly one. Hence we get

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

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

$$\begin{aligned}&\Pr[y = y'|z = z'] \\ &= 2^{-k} \sum_{j=1}^{2^k} \sum_{i=1}^{|S_k|} \Pr[y = y'|z = z' = z_j \wedge y = y_i] \cdot \Pr[y = y_i|z = z' = z_j] \\ &= 2^{-k} \sum_{j=1}^{2^k} \sum_{i=1}^{|S_k|} \Pr[y' = y_i|z' = z_j] \cdot \Pr[y = y_i|z = z_j].\end{aligned}$$

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 in algorithm 4 by $c(y, z) = \Pr[f_k(x) = y|f_\flat(x) = z]$. 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

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

in which we have reversed the order of summation. From the details of how the values of c are assigned in algorithm 4, 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 $2^k D_k(y_i)$. 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 $2^k D_k(y_i)$. In this case, the sum of the squares of these values will be less than $2^k D_k(y_i)$. We can express this sum as $c_1^2 + c_2^2 = c_1^2 + (2^k D_k(y_i) - c_1)^2$. The minimum value for this parabola is $2^{2k-1} D_k(y_i)^2$, when $c = 2^{k-1} D_k(y_i)$. Therefore we may write,

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

Substituting this expression into equation (2), we get

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

Therefore we conclude that the oracle converter induced by S_2 has collision-conversion rate at least $\frac{1}{2}$.

C Alternative proof of lower bound

Theorem 7. *Suppose $|X| = M = o(\sqrt{N})$. Any q-query quantum algorithm finds a collision in $f \leftarrow X^{D_{k,\delta}}$ with probability $O(q^2/2^k)$.*

We prove this by reducing a variant of Grover's search problem in [24] to finding a collision here. Define the following distribution E_λ on $\mathcal{F} : X \to \{0, 1\}$: if $F \leftarrow E_\lambda$, then for any $x \in X$

$$F(x) = \begin{cases} 1 & \text{with prob. } \lambda\,; \\ 0 & \text{with prob. } 1 - \lambda\,. \end{cases}$$

It has been shown in [24] that searching for a preimage of q in a function drawn according to F_λ is difficult. More precisely, for any quantum algorithm A making q queries, we define its success probability as

$$\mathsf{Succ}_{A,q}^{\lambda} := \Pr_{F \leftarrow E_\lambda} [F(x) = 1 : x \leftarrow A^F(\cdot)]\,.$$

Lemma 11. *([24, Theorem 1]) $\mathsf{Succ}_{A,q}^{\lambda} \leq 8\lambda(q+1)^2$ holds for any quantum algorithm A making at most q queries*

Proof (Proof of Theorem 7). Let A be any quantum algorithm that makes at most q queries to $f \leftarrow {D_{k,\delta}}^X$ and finds a collision in f with probability ε. We show how to construct $\mathcal{B}$ that solves the above search problem for $\lambda = 1/2^k$

making $2q$ queries with probability $\varepsilon' = \varepsilon - \gamma$ and Lemma 11 then implies that $\varepsilon \leq O(q^2/2^k)$.

$\mathcal{B}$ is given quantum access to $F \leftarrow F_\lambda$, and the mode m of $D_{k,\delta}$. Let $h : X \to Y \backslash \{m\}$ be a random function [6] It simulates $\hat{f} : X \to Y$ which answers the queries from A:

$$\hat{f}(x) = \begin{cases} m & \text{if } F(x) = 1\,; \\ h(x) & \text{o.w.}\,. \end{cases}$$

After $\mathcal{A}$ has made q queries to $\hat{f}$, $\mathcal{A}$ outputs x and x'. $\mathcal{B}$ outputs one of them, e.g., x.

Note that $\mathcal{B}$ can implement each evaluation of $\hat{f}$ by two queries to F. Therefore $\mathcal{B}$ makes $2q$ queries to F at most. Next observe that $\hat{f}$ is distributed *identically* as $f \leftarrow X^{D_{k,\delta}}$, because $\Pr[\hat{f}(x) = m] = \Pr_{F \leftarrow F_\lambda}[F(x) = 1] = 1/2^k$ and the rest of $\hat{f}(x)$ is uniform over $Y \backslash \{m\}$. Therefore we know that $\hat{f}(x) = \hat{f}(x')$ with probability ε. Finally notice that when $M = o(\sqrt{N})$, h will be injective except with probability negligible in k. Therefore the collision only occurs at the mode, which implies that $F(x) = 1$ and B successfully finds a marked element in F.

Corollary 5. *Any quantum algorithm needs $\Omega(2^{k/2})$ queries to find a collision in $X^{D_{k,\delta}}$ with constant probability even if the mode m of D is known, when $M = o(\sqrt{N})$.*

Remark 2. To see that this result is basically subsumed by Theorem 4. Note that when $N < 2^{2k}$, $\beta(D) = N$. Therefore $M = o(\sqrt{N}) = o(\sqrt{\beta(D)})$, and the function drawn is almost always injective. Hence the lower bound trivially holds. When $N \geq 2^{2k}$, Corollary 3 gives the same lower bound $2^{k/2}$.

The same proof strategy also works for general M, but then the probability that the collision occurs elsewhere other than the mode will introduce error, and it will match the bound we obtain from Theorem 4.

D Preimage- and second-preimage resistance of non-uniform random functions

In this section we sketch out a proof that the same redistribution function sampler S_2 from Algorithm 4 can be used to demonstrate similar results to Theorem 3, but relating to (second) preimage-resistance. First we give a formal definition for the problem of finding a (Second) Preimage:

Definition 11 ((Second-)Preimage finding problem). *Let $f \leftarrow D^X$ be a function of min-entropy k. For any $y \in Y$, we call $x \in X$ a* preimage *in f of y if $f(x) = y$. We refer to the problem of finding a preimage of y which has been generated by sampling a uniformly random $x \in X$ and computing $y = f(x)$ as the preimage finding problem in D. We refer to the problem of being given a*

[6] This can be efficiently simulated by a $2q$-wise independent hash function as justified by [38, Theorem 6.1] and [25, Lemma 2].

uniformly sampled $x \in X$ and finding a $x' \in X$ such that $x \neq x'$ and $f(x') = f(x)$ as the second-preimage finding problem in D.

The reduction algorithm will then be similar to that of Algorithm 1, but specified to the problem of (second-)preimage resistance, and using Redistribution Function Sampler S_2 from Algorithm 4.

Algorithm 5 Preimage converter via oracle converter

```
Input: Let $f \leftarrow {D_{k,\flat}}^X$ be a random function whose images are sampled according to
    $D_{k,\flat}$ on a set $Y$. Let $D'$ be a distribution on a set $Y'$ with min-entropy $k$. Let $S_2$
    be the $D_{k,\flat} \to D'$ redistribution function sampler from Algorithm 4. Let $\mathcal{A}$ be an
    adversary for (second-)preimage-finding in $D'$. Let $y$ be the preimage target, or let
    $x$ be the second-preimage target.
Output: A possible (second-)preimage, $x'$.
 1: Run $S_2$ and store its output as $r$. Implement an oracle for $r$.
 2: Construct an oracle converter $\mathcal{C}$ using the oracles for $f$ and $r$. The responses of $\mathcal{C}$
    are now distributed according to $D'$. Refer to the function implemented by $\mathcal{C}$ as $g$.
 3: For the second-preimage problem, provide $x$ as the second-preimage target. For the
    preimage problem, sample a uniformly random $x \in X$ and provide $y' = r(x, y)$ as
    the target.
 4: Initialize $\mathcal{A}$. For each query made by $\mathcal{A}$, forward the query to $\mathcal{C}$ and return the
    response to $\mathcal{A}$.
 5: When $\mathcal{A}$ returns a (second-)preimage $x'$ in $g$, return $x'$.
```

We then show a similar lemma to that of Lemma 6.

Lemma 12. *Suppose there exists an algorithm $\mathcal{A}$ which solves (second-)preimage finding in a distribution D' with probability at least $P_{\mathcal{A}}$, using q queries to an oracle for a function g whose responses are distributed according to D'. Then Algorithm 5 initialized with S_2 and $\mathcal{A}$, denoted $\mathcal{R}_{S_2,A}$, solves (second-)preimage finding in $D_{k,\flat}$ with probability at least $P_{\mathcal{A}}/2$ using $2q$ queries to an oracle for f whose images are distributed according to $D_{k,\flat}$.*

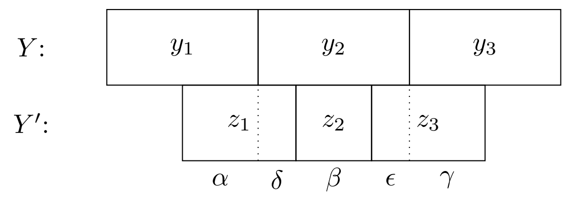
Proof. We have already shown that the function g presented to $\mathcal{A}$ in Algorithm 5 will exactly have the distribution D'. Next we need to ensure that the target for the (second-)preimage problem has the same distribution. For the second-preimage problem, this can be seen to be the case, because the target is simply a uniformly random $x \in X$, as required.

For the preimage problem things are slightly more complex. Only the point y is provided to $\mathcal{R}_{S_2,A}$, and so to sample a point $y' \in Y'$, we choose a uniformly random point $x \in X$ and set $y' = r(x, y)$. We must show that the distribution of y' is exactly D'. Note that this is a stronger requirement than what is guaranteed by Definition 8, which only states that the distribution of $r(x, f(x))$ is exactly D', and does not specify what happens for $r(x, y)$ when $y \neq f(x)$. However, S_2 does in fact satisfy this stronger property, because we construct r for each $x \in X$ and $y \in Y$, not just $y = f(x)$.

So our remaining task is to show that $\mathcal{R}_{S_2,A}$ succeeds with probability at least $P_{\mathcal{A}}/2$.

At the end of the execution of $\mathcal{R}_{S_2,A}$, with probability $P_{\mathcal{A}}$, $\mathcal{A}$ produces a $x' \in X$ such that $g(x') = r(x', f(x')) = y' = f(x, y)$. For the second preimage problem there is the further restriction that $x' \neq x$ and that $y = f(x)$.

We then have solved the (second-)preimage problem, as long as $f(x') = y$. So we need to consider the probability that $f(x') = y$, given that $r(x', f(x')) = r(x, y)$.



We show that with probability at least 1/2, it is the case that $f(x') = y$. Consider the following case. The 'target point' y that we are trying to find a (second) preimage to is y_2.

Our redistribution function, with the second half of the output being y_2, may return many possible outputs. For simplicity, and without loss of generality, we consider the case where there are three possible outputs: z_1, z_2, or z_3. We will explain in more detail in a moment why we may do this without loss in generality. While $z_2 = r(a, b)$ means $b = y_2$, if $r(a, b) = z_1$ then $b = y_1$ or y_2, and if $r(a, b) = z_3$ then $b = y_2$ or y_3.

We define the following probabilities, taken over uniformly random a and b:

$$\delta_i := \Pr[r(a,b) = z_i \wedge b = y_2], \quad i = 1,2,3\,;$$
$$\alpha := \Pr[r(a,b) = z_1 \wedge b = y_1], \quad \beta := \Pr[r(a,b) = z_3 \wedge b = y_3]\,.$$

Then we can perform the calculation of the probability that $f(x') = y_2$ (conditioned on the fact that $y = y_2$) as follows:

$$\begin{aligned}
\Pr[f(x') = y_2] =& \Pr[f(x') = y_2 \wedge y' = z_1] + \Pr[f(x') = y_2 \wedge y' = z_2] \\
&+ \Pr[f(x') = y_2 \wedge y' = z_3] \\
=& \Pr[y' = z_1] \Pr[f(x') = y_2 | y' = z_1] \\
&+ \Pr[y' = z_2] \Pr[f(x') = y_2 | y' = z_2] \\
&+ \Pr[y' = z_3] \Pr[f(x') = y_2 | y' = z_3] \\
=& 2^k \delta_1 \frac{\delta_1}{\alpha + \delta_1} + 2^k \delta_2 + 2^k \delta_3 \frac{\delta_3}{\beta + \delta_3} \\
=& 2^k \left(\delta_2 + \frac{\delta_1^2}{\alpha + \delta_1} + \frac{\delta_3^2}{\delta_3 + \beta} \right).
\end{aligned}$$

We can see that we additionally constrained by the facts that $\alpha + \delta_1 \leq \frac{1}{2^k}$ and $\beta+\delta_3 \leq \frac{1}{2^k}$. So we can see that this equation is minimized by setting: $\delta_2 = 0$, $\alpha = \beta = \delta_1 = \delta_3 = \frac{1}{2^{k+1}}$. In this case we get

$$2^k \left(0 + \frac{1/2^{2k+2}}{1/2^k} + \frac{1/2^{2k+2}}{1/2^k}\right) = 1/2.$$

The reason that this is without loss of generality is because the equation was minimized by having $\beta = 0$. In other words, the probability of finding a correct second preimage was minimized when there was *no* $z \in Y'$ such that $f(a,b) = z$ guaranteed that $b = y$. Then because there are at most 2 $z \in Y'$ such that there are multiple $b \in Y$ with $f(a,b) = z$, we have that this case with $\beta = 0$ does indeed minimize the probability of recovering a (second-)preimage.

Using this lemma, we can then compose and prove a formulation of Theorem 3, but with respect to (second-)preimage resistance.

Theorem 8. *Let $f_D \leftarrow D^X$ be a random function whose outputs are chosen independently according to a distribution D of min-entropy k. Then any quantum algorithm making q queries to f_D solves the (second-)preimage problem with probability at most $16 \cdot 2^{-k}(2q+1)^2$*

Proof. We have shown that an adversary $\mathcal{A}$ that can solve the (second-)preimage problem on a distribution D in q queries with probability $P_{\mathcal{A}}$ implies the existence of a way to solve the (second-)preimage problem on a flat distribution function $f_{D_{k,\flat}}$ in $2q$ queries with probability $P_{\mathcal{A}}/2$.

We can then reduce the hard average-case search problem in Lemma 11, to finding preimiage and second-preimage in a flat distribution, following similar analysis for uniform random functions as in [24]. As a result, any adversary's success is at most $8 \cdot 2^{-k}(q+1)^2$ in q queries.

We have shown that if a quantum algorithm exists that can solve the (second-) preimage problem on f_D in q queries with probability $P_{\mathcal{A}}$, then there exists an algorithm that can solve (second-)preimage resistance on a flat distribution in $2q$ queries with probability $P_{\mathcal{A}}/2$. This tells us that

$$P_{\mathcal{A}}/2 \leq 8 \cdot 2^{-k}(2q+1)^2,$$

from which the result follows.

Next, we apply (generalized) Grover's search algorithm to solve the preimage and second-preimage problems in min-k distributions.

Theorem 9. *Let $\beta := \beta(D_k)$. Let X be a set with size $\Omega(\beta)$. Then in q quantum queries the* QSearch *algorithm referred to in Lemma 10 solves (second) preimage problems with probability $\Omega(q^2/\beta)$.*

Proof. Given $x \leftarrow X, f \leftarrow \mathcal{F}XD_k$, let $Z_{f,x} := |\{z \in X : f(z) = f(x)\}|$ be the size of the preimage of $f(x)$. By Lemma 10, QSearch will succeed in finding a preimage of $f(x)$ with probability

$$\sum_{f,x} p_{f,x} \Omega(q^2 \frac{Z_{f,x}}{|X|}) = q^2 \frac{1}{|X|} \mathbb{E}_{f,x}(Z_{f,x}) \,.$$

We can estimate $\mathbb{E}_{f,x}(Z_{f,x}) = |X| \cdot \sum_y D_k(y)^2 = |X|/\beta$. Therefore, we find a preimage with success probability $\Omega(q^2/\beta)n$. The same analysis works for finding a second preimage.