Fixing Cracks in the Concrete: Random Oracles with Auxiliary Input, Revisited

Yevgeniy Dodis[1]*, Siyao Guo[2]**, and Jonathan Katz[3]***

[1] New York University, USA
dodis@cs.nyu.edu
[2] Simons Institute, UC Berkeley
siyao.guo@berkeley.edu
[3] University of Maryland, USA
jkatz@cs.umd.edu

Abstract. We revisit the security of cryptographic primitives in the random-oracle model against attackers having a bounded amount of *auxiliary information* about the random oracle. This situation arises most naturally when an attacker carries out offline preprocessing to generate state (namely, auxiliary information) that is later used as part of an on-line attack, with perhaps the best-known example being the use of rainbow tables for function inversion. It also models *non-uniform* attackers that may depend on the random oracle.

Unruh (Crypto 2007) introduced a generic technique for analyzing security in this model: a random oracle about which S bits of arbitrary auxiliary information are known can be replaced by a random oracle whose value is fixed in some way on P points; the two are distinguishable with probability at most $O(\sqrt{ST/P})$ by attackers making at most T oracle queries. Unruh conjectured that the distinguishing advantage could be made negligible for a sufficiently large polynomial P.

We show that Unruh's conjecture is *false* by proving that the distinguishing probability when using the above technique is at least $\Omega(ST/P)$. Faced with this negative general result, we establish new security bounds in this setting for specific applications of random oracles including the construction of one-way functions, collision-resistant hash functions, pseudorandom generators/functions, and message authentication codes. We also explore the effectiveness of *salting* as a mechanism to defend against auxiliary information, and give quantitative bounds demonstrating that salting provably helps for the aforementioned applications.

1 Introduction

The random-oracle model [4] often provides a simple and elegant way of analyzing the concrete security of cryptographic schemes based on hash functions.

* Work done while visiting the University of Maryland. Partially supported by gifts from VMware Labs and Google, and NSF awards #1619158, #1319051, and #1314568.
** Work done while at NYU and visiting the University of Maryland.
*** Work supported in part by NSF award #1223623.

To take a canonical example, consider (naïve) password hashing where a password pw is stored as $\mathcal{O}(pw)$ for $\mathcal{O}$ a cryptographic hash function, and we are interested in the difficulty of recovering pw from $\mathcal{O}(pw)$ (i.e., we are interested in understanding the *one-wayness* of $\mathcal{O}$). In practice, when $\mathcal{O}$ is instantiated using various cryptographic hash functions, it is empirically difficult to recover pw from $\mathcal{O}(pw)$ when pw is chosen from *any* high-entropy distribution; in fact, quantitatively speaking, if pw has min-entropy k then the best known attacks take $O(2^k)$ work to recover pw. But it seems difficult to formalize natural, concrete assumptions about $\mathcal{O}$ that would allow us to prove such statements, or to show that known attacks are the best possible. If we model $\mathcal{O}$ as a random oracle, however, then statements such as these can be proven easily.

Importantly, the above discussion assumes that no preprocessing is done. That is, we imagine an attacker who does no work prior to being given $\mathcal{O}(pw)$ or, more formally, we imagine that the attacker is fixed before the random oracle $\mathcal{O}$ is chosen. In that case, the only way an attacker can learn information about $\mathcal{O}$ is by making explicit queries to an oracle for $\mathcal{O}$ during an *online phase* (i.e., after being given $\mathcal{O}(pw)$), and the above-mentioned bounds hold. Note, however, that $\mathcal{O}$ might be a standardized hash function that is known in advance, and *offline preprocessing attacks*—during which the attacker can query and store arbitrary information about $\mathcal{O}$—can be a significant threat in practice.

Concretely, let $\mathcal{O} : [N] \to [N]$ (where $[n] = \{1, \ldots, n\}$) and assume pw is uniform in $[N]$. The obvious attack to recover pw from $\mathcal{O}(pw)$ is an exhaustive-search attack which uses time $T = N$ in the *online phase* (equating time with the number of queries to $\mathcal{O}$) to recover pw. But an attacker could also generate the entire function table for $\mathcal{O}$ during an offline preprocessing phase; then, given $\mathcal{O}(pw)$, the attacker can recover pw in $O(1)$ time using a table lookup. In the example just mentioned the data structure generated during the offline phase requires $S = O(N)$ space (ignoring $\log N$ factors), but Hellman [12] showed a more clever construction that, in particular, gives an attack using $S = T = O(N^{2/3})$ when $\mathcal{O}$ is treated as a random function (see [14, Section 5.4.3] for a self-contained description). *Rainbow tables* implementing this approach along with later improvements (most notably by Oechslin [17]) are widely used in practice, and must be taken into account in any practical analysis. Further work has explored improving these attacks and designing rigorous versions of them that work for *all* function $\mathcal{O}$, as well as showing bounds on how well such attacks can perform [21, 9, 10, 2, 7].

The above discussion in the context of function inversion gives a practical example of where *auxiliary information* about a random oracle (in this case, in the form of rainbow tables generated using the random oracle) can quantitatively change the security of a given application that uses the random oracle. For a more dramatic (but less practically relevant) example, consider the case of collision finding. Given a random function $\mathcal{O} : [N] \to [M]$ where $N > M$, one can show that $O(\sqrt{M})$ queries are needed in order to find a *collision* in $\mathcal{O}$ (i.e., distinct points x, x' with $\mathcal{O}(x) = \mathcal{O}(x')$). But clearly a collision in $\mathcal{O}$ can be found during an offline pre-processing phase and stored using $O(1)$ space, after which it is trivial to output that collision in an online phase in $O(1)$ time. The overall conclusion is that in settings where offline preprocessing is a possibility, security proofs in the random-oracle model must be interpreted carefully. (We refer the reader to [19, 5], as well as many of the references below, for further discussion.)

Another motivation for studying auxiliary information in the random-oracle model comes from the desire to obtain accurate security bounds against *non-uniform attackers* that may depend on the random oracle; this is again motivated by the fact that random oracles are usually instantiated by a hash function that is fixed in advance. The distinction between uniform and non-uniform attackers in the random-oracle model has led to some confusion; we refer to [18, 5] for discussion. As yet another example to illustrate the point, consider using a length-increasing random oracle $\mathcal{O} : [N] \to [M]$ as a pseudorandom generator. It is easy to show that a uniform attacker making T oracle queries can distinguish the output of $\mathcal{O}$ from uniform with probability at most $O(T/N)$. But it is also known [1, 8, 5] that there always exists a *non-uniform* attacker with $O(\log N)$-bit advice about the random oracle, and making *no* random oracle queries, that can distinguish the output of $\mathcal{O}$ from uniform with probability $\Omega(1/\sqrt{N})$.

Random oracles with auxiliary input. While somewhat different, the two motivating applications above effectively reduce to the following extension of the traditional random-oracle model, first highlighted explicitly by Unruh [19], that we refer to as the *random-oracle model with auxiliary input* (ROM-AI): First, as in the traditional random-oracle model, a function $\mathcal{O}$ is chosen uniformly from the space of functions on some domain and range. Then, a computationally unbounded attacker A_0 can compute S bits of information $\mathsf{st}_{\mathcal{O}}$ about $\mathcal{O}$ in an offline phase before attacking the system (e.g., before keys are chosen by honest parties). In a later, online phase, an attacker A_1 is given $\mathsf{st}_{\mathcal{O}}$ and can additionally make at most T queries to $\mathcal{O}$ while attacking the system. This naturally maps to the preprocessing model discussed earlier, and also can be used to analyze security against non-uniform circuits of size C by setting $S = T = C$.[1]

1.1 Handling Random Oracles with Auxiliary Input

Broadly speaking, there are three ways one can address the issue of auxiliary input in the random-oracle model: (1) by using a generic approach that applies to all uses of the random oracle, (2) by using an application-specific approach to analyze existing schemes, or (3) by modifying existing schemes in an attempt to defeat the use of auxiliary input. We discuss prior work along these lines below.

A generic approach. Unruh [19] was the first to propose a generic approach for dealing with auxiliary input in the random-oracle model. We give an informal overview of his results (a formal statement is given in Section 2). Say we wish to bound the success probability ϵ (in some experiment) of an online attacker

[1] However, separating S and T is handy for applications to non-uniform RAM computation, where S corresponds to the memory size, and T — to the number of random access queries.

making T random-oracle queries, and relying on S bits of (arbitrary) auxiliary information about the random oracle. Unruh showed that it suffices to analyze the success probability $\epsilon'(P)$ of the attack in the presence of a "pre-sampled" random oracle that is chosen uniformly subject to its values being fixed in some arbitrary way on P points (where P is a parameter), and no other auxiliary information is given; ϵ is then bounded by $\epsilon'(P) + O(\sqrt{ST/P})$.

This is an impressive result, but it falls short of what one might hope for. In particular, P must be super-polynomial in order to make the "security loss" $O(\sqrt{ST/P})$ negligible, but in many applications if P is too large then the bound $\epsilon'(P)$ one can prove on an attacker's success probability in the presence of a pre-sampled random oracle with P fixed points becomes too high. Unruh conjectured that his bound was not tight, and that it might be possible to bound the "security loss" by a negligible quantity for P a sufficiently large polynomial.

Application-specific approaches. One might hope to prove tighter bounds for specific constructions and applications. Bounds on the effectiveness of rainbow tables are given in [21, 2]. De et al. [7] adapted the "compression paradigm" of Gennaro and Trevisan [11, 10] to show limitations on attacks for inverting a *permutation* $\mathcal{O}$ (see also the appendix of [13]) as well as for distinguishing the output of a specific pseudorandom generator based on $\mathcal{O}$. However, the lower bounds of De et al. are for attacks that work for *all* $\mathcal{O}$, rather than for attacks that work with high probability for random $\mathcal{O}$. In particular, they do not directly apply when using random oracle *itself* as a one-way function or a pseudorandom generator.

"Salting." One defense against auxiliary input, which has been explicitly suggested as early as 1979 [16] and is widely used to defeat such attacks in the context of password hashing, is to use *salting*. Roughly, this involves choosing a random but public value a and using $\mathcal{O}(a, \cdot)$ in place of $\mathcal{O}(\cdot)$. Thus, in the context of password hashing we would choose a uniform salt a and store $(a, \mathcal{O}(a, pw))$; in the context of collision-resistant hashing we would choose and publish a and then look at the hardness of finding collisions in the function $\mathcal{O}(a, \cdot)$; and in the context of pseudorandom generators we would choose a and then look at the pseudorandomness of $\mathcal{O}(a, x)$ (for uniform, secret x) given a.

De et al. [7] study the effect of salting for inverting a *permutation* $\mathcal{O}$ as well as for a specific pseudorandom generator based on one-way permutations; as noted above, however, they were interested in bounds for inverting/distinguishing all $\mathcal{O}$ rather than random $\mathcal{O}$. Chung et al. [6] study the effects of salting in the design of collision-resistant hash functions, and used Unruh's pre-sampling technique to argue that salting defeats pro-processing in this important case. However, they only considered particular values of S, T and salt space K (namely, $S = T = M^{1/10}$, $K = M^4$), and obtained security $\varepsilon = M^{-1/10}$ which is much weaker than the "expected" bound $\varepsilon = M^{-4/5}$ given by the classical "birthday attack". The use of salting for obtaining non-uniform security was further advocated by Mahmoody and Mohammed [15], who used this technique for obtaining certain non-uniform black-box separation results, and conjectured that this technique should find further applications — a conjecture we will validate in this work (by studying the effect of salting on a variety of natural cryptographic primitives).

Beyond that we are not aware of any analysis of the effectiveness of salting for defeating preprocessing in any other contexts.[2] We highlight that although it may appear "obvious" that salting defeats, say, rainbow tables, it is not at all clear what is the *quantitative* security benefit of salting, and it is not clear whether rainbow tables can perhaps be adapted to give a (possibly different) online/offline tradeoff when salting is used.

1.2 Our Results

We address all three approaches outlined in the previous section. First, we investigate the generic approach to proving security in the random-oracle model with auxiliary input, and specifically explore the extent to which Unruh's pre-sampling technique can be improved. Here, our result is negative: disproving Unruh's conjecture, we show that there is an attack for which the "security loss" stemming from Unruh's approach is at least $\Omega(ST/P)$. Although there remains a gap between our lower bound and Unruh's upper bound that will be interesting to close, the upshot (as we discuss next) is that Unruh's technique is not sufficient for proving strong concrete-security bounds in the random-oracle model when preprocessing is a possibility.

Consider, e.g., the case of function inversion. One can show that the probability of inverting a random oracle $\mathcal{O} : [N] \to [N]$ for which P points have been "pre-sampled" is $O(P/N+T/N)$. Combined with the security loss of $O(\sqrt{ST/P})$ resulting from Unruh's technique, and plugging in the optimal value of P, we obtain a security bound of $O((ST/N)^{1/3}+T/N)$ for algorithms making T oracle queries and using S bits of auxiliary input about $\mathcal{O}$. Further, our negative result shows that the best bound one could hope to prove by using Unruh's approach is $O((ST/N)^{1/2} + T/N)$. Both bounds fall short of the best known attacks, which succeed with probability $\Omega\left(\min\left\{\frac{T}{N}, (\frac{S^2T}{N^2})^{1/3}\right\} + \frac{T}{N}\right)$. Similar gaps exist for other cryptographic primitives.

Faced with this, we turn to studying a direct approach for proving tighter bounds for specific applications of hash functions, namely, their use as one-way functions, pseudorandom generators/functions (PRGs/PRFs), or message authentication codes (MACs). Here we show much tighter, and in many cases optimal, bounds for the security of these primitives (cf. Table 1 with $K = 1$); our bounds always beat what can be shown using the provable version of Unruh's pre-sampling technique.

The bounds we show are weaker than what might hope for—but in many cases this is inherent since attacks using preprocessing that match our bounds are known to exist. Given this inherent limitation as compared to the traditional random-oracle model, we formally examine the effects of "salting" as a way of mitigating or even defeating the effects of pre-processing. Here we look at

[2] Bellare et al. [3] study security of salting for the purposes of multi-instance security, but they do not address the issue of preprocessing.

	Security bounds (here)	Best known attacks
OWFs	$\frac{ST}{K\alpha} + \frac{T}{\alpha}$	$\min\left\{\frac{ST}{K\alpha}, (\frac{S^2T}{K^2\alpha^2})^{1/3}\right\} + \frac{T}{\alpha}$
CRHFs	$\frac{S}{K} + \frac{T^2}{M}$	$\frac{S}{K} + \frac{T^2}{M}$
PRGs	$(\frac{ST}{KN})^{1/2} + \frac{T}{N}$	$(\frac{S}{KN})^{1/2} + \frac{T}{N}$
PRFs	$(\frac{ST}{KN})^{1/2} + \frac{T}{N}$	$(\frac{S}{KN})^{1/2} + \frac{T}{N}$
MACs	$\frac{ST}{KN} + \frac{T}{N} + \frac{T}{M}$	$\min\left\{\frac{ST}{KN}, (\frac{S^2T}{K^2N^2})^{1/3}\right\} + \frac{T}{N} + \frac{1}{M}$

Table 1. Security bounds and best known attacks using space S and time T for "salted" constructions of primitives based on a random oracle. The first three (unkeyed) primitives are constructed from a random oracle $\mathcal{O} : [K] \times [N] \to [M]$, where $[K]$ is the domain of the salt, $[N]$ is the domain of the secret and $\alpha = \min(N, M)$; the final two (keyed) primitives are constructed from a random oracle $\mathcal{O} : [K] \times [N] \times [L] \to [M]$, where $[L]$ is the domain of the input. For simplicity, logarithmic terms and constant factors are omitted.

the natural, "salted" constructions of one-way functions, collision-resistant hash functions (CRHFs), PRGs, PRFs and MACs. Note that the "unsalted" results for one-way functions, PRGs, PRFs, and MACs mentioned above are simply special cases of our results when the cardinality of the salt space is $K = 1$.

Our results are summarized in Table 1, where they are compared to the best known attacks using preprocessing. In the case of CRHFs, our bound is essentially tight and matches the trivial attack of storing explicit collisions for S distinct salts. In the remaining cases, although our bounds are not tight, it is interesting to note that (assuming $N \geq T \geq S$) our results show that setting the length of the salt equal to the length of the secret (i.e., setting $K = N$) yields the same security bound $O(T/N)$ that is achieved for constructions in the standard random-oracle model *without* preprocessing. Summarizing a bit informally: using an n-bit salt and an n-bit secret gives n-bit security *even in the presence of preprocessing.* Thus, salting provably defeats preprocessing in these settings. To phrase this differently, when salting is feasible, one gets the (essentially) the same security in the ROM-AI as in the traditional ROM; which means that the important concern expressed by the work of Bernstein and Lange [5] — regarding the use of (potentially unrealistic) non-uniform attackers in definitions of security — becomes a moot point when salting is possible.

All our bounds are proven using the "compression paradigm" introduced by Gennaro and Trevisan [11, 10]. The main idea is to argue that if some attacker succeeds with "high" probability, then that attacker can be used to reversibly encode (i.e., compress) a random oracle beyond what is possible from an information-theoretic point of view. Since we are considering attackers who perform preprocessing, our encoding must include the S-bit auxiliary information produced by the attacker. Thus, the main technical challenge we face is to ensure that our encoding compresses by (significantly) more than S bits.

Outlook. In this work we revisit the random-oracle model with auxiliary input, as we believe it has received insufficient attention from the cryptographic community despite being important for a variety of reasons. We hope our work will motivate researchers to further explore the security of other constructions in the random-oracle model in the presence of auxiliary information, as well as the effect of preprocessing on other ideal primitives (e.g., ideal ciphers).

2 Limits On the Power of Pre-Sampling

For two distributions D_1, D_2 over universe Ω, we use $\Delta(D_1, D_2)$ to denote their statistical distance $\frac{1}{2} \cdot \sum_{y \in \Omega} |\Pr[D_1 = y] - \Pr[D_2 = y]|$. Let $\mathsf{Func}(A, B)$ denote the set of all functions from A to B.

In this section, we revisit the result of Unruh [19] that allows one to replace arbitrary (bounded-length) auxiliary information about a random oracle $\mathcal{O}$ with a (bounded-size) set fixing the value of the random oracle on some fraction of points. For a set of tuples $Z = \{(x_1, y_1), \ldots\}$, we let $\mathcal{O}'[Z]$ denote a random oracle chosen uniformly subject to the constraints $\mathcal{O}'(x_i) = y_i$.

Theorem 1 ([19]). *Let $P, S, T \geq 1$ be integers, and let A_0 be an oracle algorithm that outputs state of length at most S bits. Then there is an oracle algorithm* Pre *outputting a set containing at most P tuples such that for any oracle algorithm A_1 that makes at most T oracle queries,*

$$\Delta(A_1^{\mathcal{O}}(A_0^{\mathcal{O}}),\ A_1^{\mathcal{O}'[\mathsf{Pre}^{\mathcal{O}}]}(A_0^{\mathcal{O}})) \leq \sqrt{\frac{ST}{2P}}\,.$$

This theorem enables proving various results in the random-oracle model even in the presence of auxiliary input by first replacing the auxiliary input with a fixed set of input/output pairs and then using standard lazy-sampling techniques for the value of the random oracle at other points. However, applying this theorem incurs a cost of $\sqrt{ST/2P}$, and so super-polynomial P is required in order to obtain negligible advantage overall. It was open whether the bound in Theorem 1 can be improved; Unruh conjectured [19, Conjecture 14] that for all polynomials S, T there is a polynomial P such that the statistical difference above is negligible. We disprove this conjecture by showing that the bound in the theorem cannot be improved (in general) below $O(ST/P)$. That is,

Theorem 2. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([N], \{0,1\})$, and let $S, T, P \geq 1$ be integers with $4P^2/ST + ST \leq N$. Then there is an oracle algorithm A_0 that outputs S-bit state and an oracle algorithm A_1 that makes T oracle queries such that for any oracle algorithm* Pre *outputting a set containing at most P tuples,*

$$\Delta(A_1^{\mathcal{O}}(A_0^{\mathcal{O}}),\ A_1^{\mathcal{O}'[\mathsf{Pre}^{\mathcal{O}}]}(A_0^{\mathcal{O}})) \geq \frac{ST}{24P}\,.$$

Proof. Pick S disjoint sets $X_1, \ldots, X_S \subset [N]$, where each set is of size $t = T \cdot (4(P/ST)^2 + 1)$. Partition each set X_i into $t/T = 4(P/ST)^2 + 1$ disjoint blocks $X_{i,1}, \ldots, X_{i,t/T}$, each of size T. Algorithm $A_1^{\mathcal{O}}$ outputs an S-bit state where the ith bit is equal to $\mathrm{maj}(\oplus_{x \in X_{i,1}} \mathcal{O}(x), \ldots, \oplus_{x \in X_{i,t/T}} \mathcal{O}(x))$ where maj is the majority function. Algorithm $A_1^{\mathcal{O}}(b_1, \ldots, b_S)$ chooses a uniform block $X_{i,j}$ and outputs 1 iff $\oplus_{x \in X_{i,j}} \mathcal{O}(x) = b_i$.

We have

$$\begin{aligned}
&\Pr[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}) = 1] \\
&= \Pr_{\mathcal{O},i,j}\left[\mathrm{maj}(\oplus_{x\in X_{i,1}}\mathcal{O}(x), \ldots, \oplus_{x\in X_{i,t/T}}\mathcal{O}(x)) = \oplus_{x\in X_{i,j}}\mathcal{O}(x)\right] \\
&= \Pr_{z_1,\ldots,z_{t/T}\leftarrow\{0,1\}, j\leftarrow[t/T]}[\mathrm{maj}(z_1, \ldots, z_{t/T}) = z_j] \\
&= \mathbb{E}_j\left[\Pr\left[\sum_{i\neq j} z_i = \frac{t/T-1}{2}\right] + \frac{1}{2}\cdot\Pr\left[\sum_{i\neq j} z_i \neq \frac{t/T-1}{2}\right]\right] \\
&= \frac{1}{2} + \frac{1}{2}\cdot\Pr\left[\sum_{i>1} z_i = \frac{t/T-1}{2}\right] \\
&= \frac{1}{2} + \binom{t/T-1}{\frac{t/T-1}{2}}\cdot 2^{-t/T} \\
&\geq \frac{1}{2} + \frac{1}{3\sqrt{t/T-1}} \;=\; \frac{1}{2} + \frac{ST}{6P},
\end{aligned}$$

where the inequality above uses $\sqrt{2\pi n}\,(n/e)^n \leq n! \leq e\sqrt{n}\,(n/e)^n$ so that

$$\binom{n}{n/2} \geq \frac{\sqrt{2\pi n}\,(n/e)^n}{(e\sqrt{n/2}\,(n/2e)^{n/2})^2} = \frac{2\sqrt{2\pi}}{e^2\sqrt{n}}\cdot 2^n \geq \frac{2}{3}\cdot\frac{2^n}{\sqrt{n}}\,.$$

On the other hand, for any algorithm Pre we have

$$\begin{aligned}
&\Pr[A_1^{\mathcal{O}'[\mathsf{Pre}^{\mathcal{O}}]}(A_0^{\mathcal{O}}) = 1] \\
&= \Pr_{i,j,\mathcal{O},\mathcal{O}'}[\mathrm{maj}(\oplus_{x\in X_{i,1}}\mathcal{O}(x), \ldots, \oplus_{x\in X_{i,t/T}}\mathcal{O}(x)) = \oplus_{x\in X_{i,j}}\mathcal{O}'(x)] \\
&\leq \frac{P/T}{St/T} + \frac{1}{2}\cdot\left(1 - \frac{P/T}{St/T}\right) \\
&= \frac{1}{2} + \frac{P}{2St} \;\leq\; \frac{1}{2} + \frac{ST}{8P}.
\end{aligned}$$

The first inequality above holds since, for any fixed $i, j, \mathcal{O}$,

$$\Pr_{\mathcal{O}'}\left[\mathrm{maj}(\oplus_{x\in X_{i,1}}\mathcal{O}(x), \ldots, \oplus_{x\in X_{i,t/T}}\mathcal{O}(x)) = \oplus_{x\in X_{i,j}}\mathcal{O}'(x)\right] = 1/2$$

unless the value of $\mathcal{O}'$ is fixed by $\mathsf{Pre}^{\mathcal{O}}$ at every point in $X_{i,j}$. But $\mathsf{Pre}^{\mathcal{O}}$ can ensure that the value of $\mathcal{O}'$ is fixed in that way for at most P/T out of the St/T blocks defined by i, j. This concludes the proof. ■

3 Function Inversion

In this section, we prove bounds on the hardness of inverting "salted" random oracles in the presence of preprocessing. For natural number n, let $[n] = \{1, \ldots, n\}$.

Consider choosing a random function $\mathcal{O} : [K] \times [N] \to [M]$ and then allowing an attacker A_0 (with oracle access to $\mathcal{O}$) to perform arbitrary preprocessing to generate an S-bit state $\mathsf{st}_{\mathcal{O}}$. We then look at the hardness of inverting $\mathcal{O}(a, x)$, given $\mathsf{st}_{\mathcal{O}}$ and a, for algorithms A_1 making up to T oracle queries, where $a \in [K]$ and $x \in [N]$ are uniform. We consider two notions of inversion: finding x itself, or the weaker goal of finding any x' with $\mathcal{O}(a, x') = \mathcal{O}(a, x)$. Assuming $N = M$ for simplicity in the present discussion, we show that in either case the probability of successful inversion is $O(\frac{ST}{KN} + \frac{T \log N}{N})$. We remark that the best bound one could hope to prove via a generic approach (i.e., using Theorem 1 with best-possible bound $O(ST/P)$) is[3] $O(\sqrt{ST/KN} + T/N)$.

By way of comparison, rainbow tables [12, 9, 17, 2, 7] address the case $K = 1$ (i.e., no salt), and give success probability $O(\min\{ST/N, (S^2T/N^2)^{1/3}\} + T/N)$. One way to adapt rainbow tables to handle salt is to compute K independent rainbow tables, each using space S/K, for the K reduced functions $\mathcal{O}(a, \cdot)$. This approach gives success probability $O(\min\{ST/KN, (S^2T/K^2N^2)^{1/3}\} + T/N)$, showing that our bound is tight when $ST^2 < KN$.

We begin with some preliminary lemmas that we will use in this and the following sections.

Lemma 1. *Say there exist deterministic procedures* $(\mathsf{Enc}, \mathsf{Dec})$ *such that for all* $m \in M$ *we have* $\mathsf{Dec}(\mathsf{Enc}(m)) = m$. *Then* $\mathbb{E}_m[\,|\mathsf{Enc}(m)|\,] \geq \log |M|$.

Proof. For $m \in M$, let $s_m = |\mathsf{Enc}(m)|$. Define $C = \sum_m 2^{-s_m}$, and for $m \in M$ let $q_m = 2^{-s_m}/C$. Then $\mathbb{E}_m[\,|\mathsf{Enc}(m)|\,] = -\mathbb{E}_m[\log q_m] - \log C$. By Jensen's inequality, $\mathbb{E}_m[\log q_m] \leq \log \mathbb{E}_m[q_m] = -\log |M|$, and by Kraft's inequality $C \leq 1$. The lemma follows. ■

Following De et al. [7], we also consider randomized encodings $(\mathsf{Enc}, \mathsf{Dec})$ for a set M. We say such an encoding has *recovery probability* δ if for all $m \in M$,

$$\Pr_r[\mathsf{Dec}(\mathsf{Enc}(m, r), r) = m] \geq \delta.$$

(Note that Dec is given the randomness used by Enc.) The *encoding length* of $(\mathsf{Enc}, \mathsf{Dec})$ is defined to be $\max_{m,r}\{\,|\mathsf{Enc}(m, r)|\}$.

Lemma 2 ([7]). *Suppose there exist randomized encoding and decoding procedures* $(\mathsf{Enc}, \mathsf{Dec})$ *for a set* M *with recovery probability* δ. *Then the encoding length of* $(\mathsf{Enc}, \mathsf{Dec})$ *is at least* $\log |M| - \log 1/\delta$.

Proof. By a standard averaging argument, there exists an r and a set $M' \subseteq M$ with $|M'| \geq \delta \cdot |M|$ such that $\mathsf{Dec}(\mathsf{Enc}(m, r), r) = m$ for all $m \in M'$. Let $\mathsf{Enc}', \mathsf{Dec}'$ be the deterministic algorithms obtained by fixing the randomness to r. By Lemma 1, $\mathbb{E}_{m'}[\,|\mathsf{Enc}'(m')|\,] \geq |M'| \geq |M| - \log 1/\delta$, and hence there exists an m' with $|\mathsf{Enc}'(m')| \geq |M| - \log 1/\delta$. ■

[3] Any such bound would take the form $O(ST/P + P/KN + T/N)$, where the first term is from application of the theorem, the second is the probability that the input to $\mathcal{A}_1$ is from the set of fixed points, and the third is the success probability of a trivial brute-force search. Setting $P = \sqrt{ST/KN}$ optimizes this bound.

We now state and prove the main results of this section.

Theorem 3. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K] \times [N],\ [M])$. For any oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T oracle queries,*

$$\Pr_{\mathcal{O},a,x}[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, \mathcal{O}(a,x)) = x] = O\left(\frac{ST}{KN} + \frac{T \log N}{N}\right).$$

Theorem 4. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K] \times [N],\ [M])$. Fix oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T oracle queries, and such that*

$$\Pr_{\mathcal{O},a,x}[x' := A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, \mathcal{O}(a,x)) : \mathcal{O}(a,x) = \mathcal{O}(a,x')] = \varepsilon\,.$$

Then either $\varepsilon < \frac{\log MN}{N}$ or else $\varepsilon = O\left(\frac{ST}{K\cdot\alpha} + \frac{T\log N}{\alpha}\right)$, where we let $\alpha = \min\{N/\log M, M\}$.

To prove Theorem 3, we first prove the following lemma:

Lemma 3. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K] \times [N],\ [M])$. Assume there exist oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T oracle queries, and such that*

$$\Pr_{\mathcal{O},a,x}\left[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, \mathcal{O}(a,x)) \text{ ever queries } (a,x) \text{ to } \mathcal{O}\right] = \epsilon.$$

Then there exists a randomized encoding for a set $\mathcal{F} \subseteq \mathsf{Func}([K] \times [N],\ [M])$ of size at least $\frac{\varepsilon}{2} \cdot M^{KN}$, with recovery probability at least 0.9 and encoding length (in bits) at most

$$KN\log M + S + K\log N - \frac{\varepsilon KN}{100T}\log\left(\frac{\varepsilon N}{100eT}\right).$$

Proof. By an averaging argument, there is a set $\mathcal{F} \subseteq \mathsf{Func}([K] \times [N],\ [M])$ of size at least $\varepsilon/2 \cdot |\mathsf{Func}([K] \times [N],\ [M])| = \frac{\varepsilon}{2} \cdot M^{KN}$ such that for all $\mathcal{O} \in \mathcal{F}$

$$\Pr_{a,x}[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, \mathcal{O}(a,x)) \text{ ever queries } (a,x) \text{ to } \mathcal{O}] \geq \epsilon/2.$$

Fix arbitrary $\mathcal{O} \in \mathcal{F}$. We encode $\mathcal{O}$ as follows. Let $\mathsf{st}_{\mathcal{O}}$ be the output of $A_0^{\mathcal{O}}$ and, for $a \in [K]$, let $U_a \subseteq [N]$ be the points x for which $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a, \mathcal{O}(a,x))$ ever queries (a,x) to $\mathcal{O}$. The high-level idea is that rather than encode the mapping $\{(x, \mathcal{O}(a,x))\}_{x\in U_a}$ explicitly, we will encode the set of points $\{\mathcal{O}(a,x)\}_{x\in U_a}$ and then use A_1 to recover the mapping. If we attempt this in the straightforward way, however, then it may happen that A_1 queries its oracle on a point for which the mapping is not yet known. To get around this issue, we use this approach for a random subset of U_a so that this only happens with small probability.

Specifically, the encoder uses randomness r to pick a set $R \subseteq [K] \times [N]$, where each $(a, x) \in [K] \times [N]$ is included in R with probability $1/10T$. For $a \in [K]$, let G_a be the set of $(a, x) \in R$ such that $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a, \mathcal{O}(a, x))$ at some point queries (a, x) to $\mathcal{O}$, but never queries $\mathcal{O}$ on any other $(a', x') \in R$. Let $G = \bigcup_a G_a$. Define $V_a = \{\mathcal{O}(a, x)\}_{x \in G_a}$, and note that $|V_a| = |G_a| \leq N$.

As in De et al. [7], with probability at least 0.9 the size of G is at least $\varepsilon KN/100T$. To see this, note that by a Chernoff bound, R has at least $\varepsilon KN/40T$ points with probability at least 0.95. The expected number of points $(a, x) \in R$ for which $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a, \mathcal{O}(a, x))$ at some point queries (a, x) to $\mathcal{O}$ but A_1 also queries $\mathcal{O}$ for some other point $(a', x') \in R$ (i.e., besides (a, x) itself) is at most $\frac{\varepsilon KN}{2} \cdot \frac{1}{10T} \cdot \left(1 - (1 - 1/10T)^T\right) \leq \frac{\varepsilon KN}{2000T}$. Thus, by Markov's inequality, with probability at least 0.95 the number of such points is at most $\frac{\varepsilon KN}{100T}$. So with probability at least 0.9, we have $|G| \geq \frac{3\varepsilon KN}{200T} \geq \frac{\varepsilon KN}{100T}$.

Assuming $|G| \geq \varepsilon KN/100T$, we encode $\mathcal{O}$ as follows:

1. Include $\mathsf{st}_{\mathcal{O}}$ and, for each $a \in [K]$, include $|V_a|$ and a description of V_a. This requires $S + K \log N + \sum_{a \in [K]} \log \binom{M}{|G_a|}$ bits.
2. For each a and $y \in V_a$ (in lexicographic order), run $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a, y)$ and include in the encoding the answers to all the oracle queries made by A_1 that have not been included in the encoding so far, except for any queries in R. (By definition of G_a, there will be at most one such query and, if so, it will be the query (a, x) such that $\mathcal{O}(a, x) = y$.)
3. For each $(a, x) \in ([K] \times [N]) \setminus G$ (in lexicographic order) for which $\mathcal{O}(a, x)$ has not been included in the encoding so far, add $\mathcal{O}(a, x)$ to the encoding.

Steps 2 and 3 explicitly include in the encoding the value of $\mathcal{O}(a, x)$ for each $(a, x) \in ([K] \times [N]) \setminus G$. Thus, the total number of bits added to the encoding by those steps is $(KN - \sum_a |G_a|) \cdot \log M$.

To decode, the decoder first uses r to recover the set R defined above. Then it does the following:

1. Recover $\mathsf{st}_{\mathcal{O}}$, $\{|V_a|\}_{a \in K}$, and $\{V_a\}_{a \in K}$.
2. For each a and $y \in V_a$ (in lexicographic order), run $A_1(\mathsf{st}_{\mathcal{O}}, a, y)$ while answering the oracle queries of A_1 using the values stored in the encoding. The only exception is if A_1 ever makes a query $(a, x) \in R$, in which case y itself is returned as the answer. If that ever occurs, then x is such that $\mathcal{O}(a, x) = y$.
3. For each $(a, x) \in [K] \times [N]$ (in lexicographic order) for which $\mathcal{O}(a, x)$ is not yet defined, recover the value of $\mathcal{O}(a, x)$ from the remainder of the encoding.

Assuming $|G| \geq \varepsilon KN/100T$, the decoding procedure recovers $\mathcal{O}$. The encoding length is

$$S + K \log N + \sum_{a \in [K]} \log \binom{M}{|G_a|} + \left(KN - \sum_{a \in K} |G_a|\right) \log M.$$

Because $\binom{M}{|G_a|} \le \left(\frac{eM}{|G_a|}\right)^{|G_a|}$, the encoding length is bounded by

$$\begin{aligned}
&S + K\log N + KN\log M - \sum_a |G_a| \log\left(\frac{|G_a|}{e}\right)\\
&\le S + K\log N + KN\log M - |G|\log\left(\frac{|G|}{eK}\right)\\
&\le S + K\log N + KN\log M - \frac{\varepsilon KN}{100T}\log\left(\frac{\varepsilon N}{100eT}\right),
\end{aligned}$$

where the second line uses concavity of the function $f(y) = -y\log(y/e)$, and the last line is because $|G| \ge \frac{\varepsilon KN}{100T}$. ■

Corollary 1. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K]\times[N],\ [M])$. Assume there exist oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T oracle queries, and such that*

$$\Pr_{\mathcal{O},a,x}\left[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, \mathcal{O}(a,x)) \text{ ever queries } (a,x) \text{ to } \mathcal{O}\right] = \epsilon.$$

Then $\varepsilon = O\left(\frac{ST}{KN} + \frac{T\log N}{N}\right)$.

Proof. Lemma 3 gives an encoding for a set of size $\frac{\varepsilon}{2}\cdot M^{KN}$ with recovery probability 0.9, and encoding length at most $NK\log M + S + K\log N - \frac{\varepsilon KN}{100T}\log\left(\frac{\varepsilon N}{100eT}\right)$ bits. But Lemma 2 shows that any such encoding must have encoding length at least $NK\log M - \log\frac{2}{\varepsilon} - \log\frac{10}{9}$ bits. We thus conclude that

$$S + K\log N + \log\frac{20}{9\varepsilon} \ge \frac{\varepsilon KN}{100T}\log\left(\frac{\varepsilon N}{100eT}\right). \tag{1}$$

Equation (1) implies the corollary, since either $\varepsilon < \frac{200eT}{N}$, or else it must be the case that $\varepsilon \le (\frac{100T}{KN})\cdot(S + K\log N + \log N)$. ■

Corollary 1 implies Theorem 3, since we may assume that if $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a, y)$ outputs x then it queries (a,x) to $\mathcal{O}$ (using one additional query, if necessary).

We now prove Theorem 4. For fixed $\mathcal{O}$ and $a \in [K]$, let $Y_{\mathcal{O},a} \subseteq [M]$ be the set of points A_1 successfully inverts, i.e.,

$$Y_{\mathcal{O},a} = \{y \ :\ \mathcal{O}(a, A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, y)) = y\}.$$

Let $X_{\mathcal{O},a} \subseteq [N]$ be the pre-images of the points in $Y_{\mathcal{O},a}$. That is,

$$X_{\mathcal{O},a} = \{x \ :\ \mathcal{O}(a,x) \in Y_{\mathcal{O},a}\}.$$

We show a deterministic encoding for $\mathsf{Func}([K]\times[N],[M])$. Given a function $\mathcal{O}$, we encode it by including for each $a \in [K]$ the following information:

1. The set $X_{\mathcal{O},a}$ (along with its size), using $\log N + \binom{N}{|X_{\mathcal{O},a}|}$ bits.

2. The set $Y_{\mathcal{O},a}$ (along with its size), using $\log M + \binom{M}{|Y_{\mathcal{O},a}|}$ bits.
3. For each $x \in X_{\mathcal{O},a}$, the value $\mathcal{O}(a,x) \in Y_{\mathcal{O},a}$ encoded using $\log |Y_{\mathcal{O},a}|$ bits.
4. For each $x \notin X_{\mathcal{O},a}$, the value $\mathcal{O}(a,x)$ encoded using $\log M$ bits.

Decoding is done in the obvious way.

The encoding length of $\mathcal{O}$ (in bits) is

$$K \log N + K \log M$$
$$+ \sum_{a \in [K]} \log \binom{N}{|X_{\mathcal{O},a}|} + \log \binom{M}{|Y_{\mathcal{O},a}|} + |X_{\mathcal{O},a}| \cdot \log |Y_{\mathcal{O},a}| + (N - |X_{\mathcal{O},a}|) \cdot \log M.$$

Using the inequality $\log \binom{A}{B} \leq B \cdot \log \frac{eA}{B}$ and the log-sum[4] inequality, the encoding length of $\mathcal{O}$ (in bits) is at most

$$K \log N + K \log M + \left(\sum_{a \in [K]} |X_{\mathcal{O},a}| \right) \cdot \log \frac{eN \sum_{a \in [K]} |Y_{\mathcal{O},a}|}{M \sum_{a \in [K]} |X_{\mathcal{O},a}|}$$
$$+ \left(\sum_{a \in [K]} |Y_{\mathcal{O},a}| \right) \cdot \log \frac{eKM}{\sum_{a \in [K]} |Y_{\mathcal{O},a}|} + KN \log M. \tag{2}$$

Let $\epsilon' \overset{\text{def}}{=} \Pr_{\mathcal{O},a,x}[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, \mathcal{O}(a,x)) = x]$, and note that $\mathbb{E}_{\mathcal{O}}[\sum_a |X_{\mathcal{O},a}|] = \epsilon NK$ and $\mathbb{E}_{\mathcal{O}}[\sum_a |Y_{\mathcal{O},a}|] = \epsilon' NK$. Applying the log-sum inequality again, we see that the expected encoding length (taken over choice of $\mathcal{O}$) is upper bounded by

$$K \log NM + (\varepsilon NK) \cdot \log \frac{eN\varepsilon' NK}{M\varepsilon NK} + (\varepsilon' NK) \cdot \log \frac{eKM}{\varepsilon' NK} + KN \log M.$$

By Lemma 1 the expected encoding length must be at least $KN \log M$ bits, and so we obtain

$$\frac{\log N + \log M}{N} + \varepsilon' \cdot \log \frac{eM}{\varepsilon' N} \geq \varepsilon \cdot \log \frac{M\varepsilon}{eN\varepsilon'}. \tag{3}$$

If $\varepsilon < \frac{\log MN}{N}$ we are done, so assume $\varepsilon \geq \frac{\log MN}{N}$. If $M\varepsilon/(eN\varepsilon') < 4$ then $\varepsilon < 4e\varepsilon' N/M$. If not, then this along with Equation (3) gives

$$\begin{aligned} \varepsilon' \log \frac{eM}{\varepsilon' N} &\geq \varepsilon \log \frac{M\varepsilon}{eN\varepsilon'} - \frac{\log NM}{N} \\ &\geq 2\varepsilon - \frac{\log NM}{N} \\ &\geq \varepsilon, \end{aligned}$$

which implies $\varepsilon = O(\varepsilon' \log M)$ (we may assume without loss of generality that $\varepsilon' \geq 1/N$). Overall we conclude that $\varepsilon = O(\varepsilon' \cdot \max\{\log M, N/M\})$. Using the bound on ε' from Theorem 3, we obtain the claimed bound on ε.

[4] The log-sum inequality states that for nonnegative $t_1, \ldots, t_n$ and $w_1, \ldots, w_n$, it holds that $\sum_{i=1}^n t_i \log(w_i/t_i) \leq \left(\sum_{i=1}^n t_i\right) \cdot \log(\sum_{i=1}^n w_i / \sum_{i=1}^n t_i)$. It also implies the average of $t_1 \log(w_1/t_1), \ldots, t_n \log(w_n/t_n)$ is less that $\overline{t} \log(\overline{w}/\overline{t})$ where $\overline{t}$ is the average of $t_1, \ldots, t_n$ and $\overline{w}$ is the average of $w_1, \ldots, w_n$.

4 Collision-Resistant Hash Functions

In this section, we prove the following theorem.

Theorem 5. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K] \times [N],\ [M])$. For any oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T oracle queries,*

$$\Pr_{\mathcal{O},a}[(x,x') := A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a) : x \neq x' \wedge \mathcal{O}(a,x) = \mathcal{O}(a,x')] = O\left(\frac{S + \log K}{K} + \frac{T^2}{M}\right).$$

The bound in the above theorem roughly matches the parameters achieved by the following: A_0 outputs collisions in $\mathcal{O}(a_i, \cdot)$ for each of $a_1, \ldots, a_S \in [K]$. Then A_1 outputs the appropriate collision if $a = a_i$ for some i, and otherwise performs a birthday attack in an attempt to find a collision.

To prove Theorem 5, we first prove the following lemma:

Lemma 4. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K] \times [N],\ [M])$. Assume there exist oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T oracle queries, and such that*

$$\Pr_{\mathcal{O},a}[(x,x') := A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a) : x \neq x' \wedge \mathcal{O}(a,x) = \mathcal{O}(a,x')] = \epsilon.$$

Then there exists a deterministic encoding for the set $\mathsf{Func}([K] \times [N],\ [M])$ with expected encoding length (in bits) at most

$$S + KN \log M + \log K - \frac{\varepsilon K}{2} \log\left(\frac{\varepsilon M}{8eT^2}\right).$$

Proof. Fix $\mathcal{O} \colon [K] \times [N] \to [M]$, and let $\mathsf{st}_{\mathcal{O}} = A_0^{\mathcal{O}}$. Let $G_{\mathcal{O}}$ be the set of $a \in [K]$ such that $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a)$ outputs a collision in $\mathcal{O}(a, \cdot)$. We assume, without loss of generality, that if $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a)$ outputs x, x', then it must have queried $\mathcal{O}(a,x)$ and $\mathcal{O}(a,x')$ at some point in its execution. The basic observation is that we can use this to compress $\mathcal{O}(a, \cdot)$ for $a \in G_{\mathcal{O}}$. Specifically, rather than store both $\mathcal{O}(a,x)$ and $\mathcal{O}(a,x')$ (using $2 \log M$ bits), where x, x' is the collision in $\mathcal{O}(a, \cdot)$ output by A_1, we instead store the value $\mathcal{O}(a,x) = \mathcal{O}(a,x')$ *once*, along with the indices i, j of the oracle queries $\mathcal{O}(a,x)$ and $\mathcal{O}(a,x')$ made by A_1 (using a total of $\log M + 2 \log T$ bits). This is a net savings if $2 \log T < \log M$. Details follow.

A simple case. To illustrate the main idea, we first consider a simple case where $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a)$ never makes oracle queries $\mathcal{O}(a', x)$ with $a' \neq a$. Under this assumption, we encode $\mathcal{O}$ as follows:

1. Encode $\mathsf{st}_{\mathcal{O}}$, $|G_{\mathcal{O}}|$, and $G_{\mathcal{O}}$. This requires $S + \log K + \log \binom{K}{|G_{\mathcal{O}}|}$ bits.
2. For each $a \in G_{\mathcal{O}}$ (in lexicographic order), run $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a)$ and let the second components of the oracle queries of A_1 be $x_1, \ldots, x_T$. (We assume without loss of generality these are all distinct.) If x, x' are the output of A_1, let $i < j$ be such that $\{x, x'\} = \{x_i, x_j\}$. Encode i and j, along with the answers to each of A_1's oracle queries (in order) except for the jth. Furthermore, encode $\mathcal{O}(a,x)$ for all $x \in [N] \setminus \{x_1, \ldots, x_T\}$ (in lexicographic order). This requires $(N-1) \cdot \log M + 2 \log T$ bits for each $a \in G_{\mathcal{O}}$.

3. For each $a \notin G_{\mathcal{O}}$ and $x \in [N]$ (in lexicographic order), store $\mathcal{O}(a, x)$. This uses $N \log M$ bits for each $a \notin G_{\mathcal{O}}$.

Decoding is done in the obvious way.

The encoding length of $\mathcal{O}$ (in bits) is

$$S + \log K + \log \binom{K}{|G_{\mathcal{O}}|} + KN \log M - |G_{\mathcal{O}}| \cdot (\log M - 2 \log T).$$

Using the inequality $\binom{K}{|G_f|} \leq (\frac{eK}{|G_f|})^{|G_f|}$, the expected encoding length (in bits) is thus

$$\begin{aligned}
&S + \log K + \mathbb{E}_{\mathcal{O}} \left[|G_{\mathcal{O}}| \cdot \log \frac{eK}{|G_{\mathcal{O}}|} \right] \\
&\quad + KN \log M - \mathbb{E}_{\mathcal{O}}[|G_{\mathcal{O}}|] \cdot (\log M - 2 \log T) \\
&\leq S + \log K + \mathbb{E}_{\mathcal{O}}[|G_{\mathcal{O}}|] \cdot \log \frac{eK}{\mathbb{E}_{\mathcal{O}}[|G_{\mathcal{O}}|]} \\
&\quad + KN \log M - \mathbb{E}_{\mathcal{O}}[|G_{\mathcal{O}}|] \cdot (\log M - 2 \log T) \\
&= S + \log K + KN \log M - \varepsilon K \log \left(\frac{\varepsilon M}{eT^2} \right),
\end{aligned}$$

where the inequality uses concavity of the function $y \cdot \log 1/y$, and the third line uses $\mathbb{E}_{\mathcal{O}}[|G_{\mathcal{O}}|] = \varepsilon K$.

The general case. In the general case, we need to take into account the fact that A_1 may make arbitrary queries to $\mathcal{O}$. This affects the previous approach because $A_1(\mathsf{st}_{\mathcal{O}}, a)$ may query $\mathcal{O}(a', x)$ for a value x that is output as part of a collision by $A_1(\mathsf{st}_{\mathcal{O}}, a')$.

To deal with this, consider running $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a)$ for all $a \in G_{\mathcal{O}}$. There are at most $T \cdot |G_{\mathcal{O}}|$ distinct oracle queries made overall. Although several of them may share the same prefix $a \in [K]$, there are at most $|G_{\mathcal{O}}|/2$ values of a that are used as a prefix in more than $2T$ queries. In other words, there is a set $G'_{\mathcal{O}} \subseteq G_{\mathcal{O}}$ of size at least $|G_{\mathcal{O}}|/2$ such that each $a \in G'_{\mathcal{O}}$ is used in at most $2T$ queries when running $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a)$ for all $a \in G'_{\mathcal{O}}$.

To encode $\mathcal{O}$ we now proceed in a manner similar to before, but using $G'_{\mathcal{O}}$ in place of $G_{\mathcal{O}}$. Moreover, we run $A_1^{\mathcal{O}}(\mathsf{st}_{\mathcal{O}}, a)$ for all $a \in G'_{\mathcal{O}}$ (in lexicographic order) and consider all the distinct oracle queries made. For each $a \in G'_{\mathcal{O}}$, let $i_a < j_a \leq 2T$ be such that the i_ath and j_ath oracle queries that use prefix a are distinct but yield the same output. (There must exist such indices by assumption on A_1.) We encode (i_a, j_a) for all $a \in G'_{\mathcal{O}}$, along with the answers to all the (distinct) oracle queries made with the exception of the j_ath oracle query made using prefix a for all $a \in G'_{\mathcal{O}}$. The remainder of $\mathcal{O}(\cdot, \cdot)$ is then encoded in the trivial way as before. Decoding is done in the natural way.

Arguing as before, but with ϵK replaced by $\epsilon K/2$ and T replaced by $2T$, we see that the expected encoding length (in bits) is now at most

$$S + \log K + KN \log M - \frac{\varepsilon K}{2} \log \left(\frac{\varepsilon M}{8eT^2} \right),$$

as claimed. ∎

Lemma 4 gives an encoding for $\mathsf{Func}([K] \times [N],\ [M])$ with expected length at most

$$S + \log K + KN \log M - \frac{\varepsilon K}{2} \log\left(\frac{\varepsilon M}{8eT^2}\right)$$

bits. But Lemma 1 shows that any such encoding must have expected length at least $NK \log M$ bits. We thus conclude that

$$S + \log K \geq \frac{\varepsilon K}{2} \log\left(\frac{\varepsilon M}{8eT^2}\right).$$

This implies Theorem 5 since either $\varepsilon \leq \frac{16eT^2}{M}$ or else $\varepsilon \leq \frac{2S+2\log K}{K}$.

5 Pseudorandom Generators and Functions

In this section, we prove the following theorems.

Theorem 6. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K] \times [N],\ [M])$. For any oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state (with $S > \log N$) and A_1 makes at most T oracle queries,*

$$\left|\Pr_{\mathcal{O},a,x}[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, \mathcal{O}(a,x)) = 1] - \Pr_{\mathcal{O},a,y}[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, y) = 1]\right|$$
$$= O\left(\log M \cdot \left(\sqrt{\frac{ST}{KN}} + \frac{T \log N}{N}\right)\right).$$

Theorem 7. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K] \times [N] \times [L],\ \{0,1\})$. For any oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T oracle queries to $\mathcal{O}$ and at most q queries to its other oracle,*

$$\left|\Pr_{\mathcal{O},a,k}[A_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(A_0^{\mathcal{O}}, a) = 1] - \Pr_{\mathcal{O},a,f}[A_1^{\mathcal{O},f}(A_0^{\mathcal{O}}, a) = 1]\right|$$
$$= O\left(q \cdot \left(\sqrt{\frac{ST}{KN}} + \frac{T \log N}{N}\right)\right),$$

where f is uniform in $\mathsf{Func}([L], \{0,1\})$.

Note that in both cases, an exhaustive-search attack (with $S = 0$) achieves distinguishing advantage $\Theta(T/N)$. With regard to pseudorandom generators (Theorem 6), De et al. [7] show an attack with $T = 0$ that achieves distinguishing advantage $\Omega(\sqrt{\frac{S}{KN}})$ when $M > N$. Their attack can be extended to the case of pseudorandom functions (assuming $L \geq q > \log N$) to obtain distinguishing advantage $\Omega(\sqrt{\frac{S}{KN}})$ in that case as well.

In proving the above, we rely on the following [7, Lemma 8.4]:

Lemma 5. *Fix a parameter ϵ, and oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T queries to $\mathcal{O}$ but may not query its input. Let $\mathcal{F} \subseteq \mathsf{Func}\left([K] \times [N],\ \{0,1\}\right)$ be such that if $\mathcal{O} \in \mathcal{F}$ then*

$$\Pr_{a,x}[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, x) = \mathcal{O}(a,x)] \geq \frac{1}{2} + \varepsilon.$$

Then there is a randomized encoding for $\mathcal{F}$ with recovery probability $\Omega(\varepsilon/T)$ and encoding length (in bits) at most $KN + S - \Omega\left(\frac{\varepsilon^2 KN}{T}\right) + O(1)$.

We now prove Theorem 6.

Proof. Let

$$\epsilon = \left| \Pr_{\mathcal{O},a,x}[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, \mathcal{O}(a,x)) = 1] - \Pr_{\mathcal{O},a,y}[A_1^{\mathcal{O}}(A_0^{\mathcal{O}}, a, y) = 1] \right|.$$

We assume for simplicity that M is a power of 2. By Yao's equivalence of distinguishability and predictability [20], there exist $i \in [\log M]$ and oracle algorithms (B_0, B_1) such that B_0 outputs at most $S+1$ bits and B_1 makes at most T oracle queries, and such that

$$\Pr_{\mathcal{O},a,x}[B_1^{\mathcal{O}}(B_0^{\mathcal{O}}, a, \mathcal{O}_1(a,x), \ldots, \mathcal{O}_{i-1}(a,x)) = \mathcal{O}_i(a,x)] \geq 1/2 + \varepsilon/\log M,$$

where $\mathcal{O}_i(a,x)$ denotes the ith bit of $\mathcal{O}(a,x)$. If B_1 queries (a,x) with probability at least $\varepsilon/2\log M$, then using Corollary 1 we have

$$\varepsilon = O\left(\log M \cdot \left(\frac{ST}{KN} + \frac{T \log N}{N}\right)\right). \tag{4}$$

Otherwise, we may construct algorithms (C_0, C_1) such that

- C_0 runs B_0 and also outputs as part of its state the truth table of a function mapping $[K] \times [N]$ to outputs of length at most $(\log M - 1)$ bits;
- C_1 makes at most T oracle queries, and never queries its own input;

and such that

$$\Pr_{\mathcal{O}_i,a,x}[C_1^{\mathcal{O}_i}(C_0^{\mathcal{O}_i}, a, x) = \mathcal{O}_i(a,x)] \geq 1/2 + \varepsilon/2\log M.$$

So for at least an $(\varepsilon/4\log M)$-fraction of $\mathcal{O}_i \in \mathsf{Func}\left([K] \times [N], \{0,1\}\right)$ we have

$$\Pr_{a,x}[C_1^{\mathcal{O}_i}(C_0^{\mathcal{O}_i}, a, x) = \mathcal{O}_i(a,x)] \geq 1/2 + \varepsilon/4\log M.$$

Using Lemma 5 and the specific construction of C_0, this means there is a randomized encoding of at least a $(\varepsilon/4\log M)$-fraction of $\mathcal{O} \in \mathsf{Func}\left([K] \times [N], [M]\right)$ with encoding length (in bits) at most

$$KN + KN \cdot (\log M - 1) + S - \Omega\left(\frac{(\varepsilon/\log M)^2 KN}{T}\right) + O(1),$$

and recovery probability at least $\varepsilon/4T\log M$. Lemma 2 then implies

$$KN\log M + S - \Omega\left(\frac{(\varepsilon/\log M)^2KN}{T}\right) \geq \log\left(\frac{\varepsilon M^{KN}}{4\log M}\right) - \log\left(\frac{4T\log M}{\varepsilon}\right).$$

Now, if $\varepsilon < \frac{4T\log M}{N}$ we are done. Otherwise, the above equation gives

$$\begin{aligned}\Omega\left(\frac{(\varepsilon/\log M)^2KN}{T}\right) &\leq S + \log\left(\frac{4\log M}{\varepsilon}\right) + \log\left(\frac{4T\log M}{\varepsilon}\right),\\ &\leq S + O(\log N) = O(S)\end{aligned}$$

which in turn means that $\varepsilon = O\left(\sqrt{\frac{ST}{KN}}\cdot\log M\right)$. This, with Equation (4), implies the theorem. ■

As intuition for the proof of Theorem 7, note that we may view a pseudorandom function as a pseudorandom generator mapping a key to the truth table for a function, with the main difference being that the distinguisher is not given the entire truth table as input but instead may only access parts of the truth table via queries it makes. We may thus apply the same idea as in the proof of Theorem 6, with the output length (i.e., $\log M$) replaced by the number of queries the distinguisher makes. However in this case, Lemma 5 cannot be directly applied and a slightly more involved compression argument is required.

With this in mind, we turn to the proof of Theorem 7:

Proof. Let

$$\epsilon = \left|\Pr_{\mathcal{O},a,k}[A_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(A_0^{\mathcal{O}},a)=1] - \Pr_{\mathcal{O},a,f}[A_1^{\mathcal{O},f}(A_0^{\mathcal{O}},a)=1]\right|.$$

By Yao's equivalence of distinguishability and predictability [20], there exist $i \in [q]$ and oracle algorithms (B_0, B_1) such that B_0 outputs at most $S+1$ bits and B_1 makes at most T oracle queries to $\mathcal{O}$ and $i < q$ distinct queries to the second oracle, and such that

$$\Pr_{\mathcal{O},a,k}[B_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(B_0^{\mathcal{O}},a) \text{ outputs } (x,b) \text{ s.t. } \mathcal{O}(a,k,x)=b] \geq \frac{1}{2} + \varepsilon/q,$$

where B_1 does not query x to its second oracle. If with probability at least $\varepsilon/2q$ it holds that B_1 queries $\mathcal{O}$ on some point of the form $(a,k,\star)$, then we can apply Corollary 1 to $\mathcal{O}' \in \mathsf{Func}([K]\times[N],\{0,1\}^L)$ (viewing $\mathcal{O}'(a,k)$ as the truth table for $\mathcal{O}(a,k,\cdot)$) to conclude that

$$\varepsilon = O\left(q\cdot\left(\frac{ST}{KN} + \frac{T\log N}{N}\right)\right). \tag{5}$$

Otherwise, we may construct an algorithm C_1 that behaves as B_1 except that it never queries $\mathcal{O}$ on any query with prefix (a,k), and such that

$$\Pr_{\mathcal{O},a,k}[C_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(B_0^{\mathcal{O}},a,k) \text{ outputs } (x,b) \text{ s.t. } \mathcal{O}(a,k,x)=b] \geq 1/2 + \varepsilon/2q.$$

(Note that C_1, like B_1, does not query x to its second oracle.) This means that for at least an $(\varepsilon/4q)$-fraction of $\mathcal{O} \in \mathsf{Func}\left([K] \times [N], \{0,1\}^L\right)$, it holds that

$$\Pr_{a,k}[C_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(B_0^{\mathcal{O}}, a, k) \text{ outputs } (x,b) \text{ s.t. } \mathcal{O}(a,k,x) = b] \geq 1/2 + \varepsilon/4q.$$

We encode this set of functions using a randomized encoding. The encoder uses randomness r to pick a set $R \subseteq [K]\times[N]$, where each $(a,k) \in [K]\times[N]$ is included in R with probability $1/10T$. Let $G \subseteq R$ be the set of $(a,k) \in R$ such that $C_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(B_0^{\mathcal{O}}, a, k)$ does not query $\mathcal{O}$ on any point with prefix $(a', k') \in R$. Let G_0 be the subset of G such that the output of C_1 is correct, and let $G_1 = G \setminus G_0$. As in De et al. [7], with probability at least $\varepsilon/160qT$, it holds that $|G| \geq \frac{KN}{40T}$ and $|G_0| - |G_1| \geq \frac{\varepsilon KN}{80qT}$. Conditioned on these holding, we encode $\mathcal{O}$ as follows:

1. Include $B_0^{\mathcal{O}}$. This uses at most $S+1$ bits.
2. For each $(a,k) \in ([K] \times [N]) \setminus R$ (in lexicographic order), include the truth table of $\mathcal{O}(a,k,\cdot)$. Then for each $(a,k) \in R \setminus G$ (in lexicographic order), include the truth table of $\mathcal{O}(a,k,\cdot)$. This uses a total of $(KN - |G|) \cdot L$ bits.
3. Include a description of G_0. This uses $\log\binom{|G|}{|G_0|}$ bits.
4. For each $(a,k) \in G$ (in lexicographic order), include in the encoding the answers to the oracle queries made by C_1 to its second oracle $\mathcal{O}(a,k,\cdot)$ in the order they are made; denote the set of such queries by X, and let (x,b) be the output of C_1. Also include in the encoding $\mathcal{O}(a,k,x')$ for all $x' \notin X \cup \{x\}$. This requires $|G| \cdot (L-1)$ bits.

To decode, the decoder first uses r to recover the set R defined above. Then it does the following:

1. Recover $B_0^{\mathcal{O}}$.
2. For each $(a,k) \in ([K]\times[N]) \setminus R$, recover the truth table of $\mathcal{O}(a,k,\cdot)$. Identify the set G by running $C_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(B_0^{\mathcal{O}}, a, k)$ for $(a,k) \in R$ by checking whether C_1 makes any queries to its first oracle whose prefix is in R. Then, for all $(a,k) \in R \setminus G$, recover the truth table of $\mathcal{O}(a,k,\cdot)$.
3. Recover G_0.
4. For each $(a,k) \in G$, run $C_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(B_0^{\mathcal{O}}, a, k)$ using values recovered already to answer queries to the first oracle, and values from the encoding to answer queries to the second oracle. When C_1 outputs (x,b), set $\mathcal{O}(a,k,x) = b$ if $(a,k) \in G_0$, and $\mathcal{O}(a,k,x) = 1-b$ otherwise. Recover the rest of the function table for $\mathcal{O}(a,k,\cdot)$ from the rest of the encoding.

Note that the above decoding procedure always succeeds when our assumptions about $|G|, |G_0|$, and $|G_1|$ hold. Moreover, given these assumptions, the length of the above encoding (in bits) is at most

$$S + 1 + KNL + \log\binom{|G|}{|G_0|} - |G| \leq S + 1 + KNL - \Omega\left(\frac{\varepsilon^2 KN}{q^2 T}\right).$$

Lemma 2 thus implies $\varepsilon = O(q \cdot \sqrt{\frac{ST}{KN}})$. Combined with Equation (5), this gives the result of the theorem. ■

6 Message Authentication Codes (MACs)

In this section, we prove the following theorem.

Theorem 8. *Consider random oracles $\mathcal{O} \in \mathsf{Func}([K] \times [N] \times [L],\ [M])$. For any oracle algorithms (A_0, A_1) such that A_0 outputs S-bit state and A_1 makes at most T queries to $\mathcal{O}$,*

$$\Pr_{\mathcal{O},a,k}\left[(m,t) := A_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(A_0^{\mathcal{O}}, a) \,:\, \mathcal{O}(a,k,m) = t\right]$$
$$= O\left(\frac{ST}{KN} + \frac{T}{M} + \frac{T\log N}{N}\right),$$

where it is required that A_1 not query m to its second oracle.

Note that any generic inversion attack can be used to attack the above construction of a MAC by fixing some $m \in [L]$ and then inverting the function $\mathcal{O}(a,\cdot,m)$ given a; in this sense, it is to be expected that the bound above contains terms $\mathcal{O}\left(\frac{ST}{KN} + \frac{T\log N}{N}\right)$ as in Theorem 3. There is, of course, also a trivial guessing attack that achieves advantage $1/M$.

Proof. Let

$$\varepsilon = \Pr_{\mathcal{O},a,k}\left[(m,t) := A_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(A_0^{\mathcal{O}}, a) \,:\, \mathcal{O}(a,k,m) = t\right].$$

If A_1 queries $\mathcal{O}$ on some point of the form $(a,k,\star)$ with probability at least $\varepsilon/2$, then we can apply Corollary 1 to $\mathcal{O}' \in \mathsf{Func}([K]\times[N],[M]^L)$ (viewing $\mathcal{O}'(a,k)$ as the function table of $\mathcal{O}(a,k,\cdot)$) to conclude that $\varepsilon = O\left(\frac{ST}{KN} + \frac{T\log N}{N}\right)$. Otherwise, we may construct an algorithm B_1 that behaves as A_1 except that it never queries $\mathcal{O}$ on any point of the form $(a,k,\star)$, and such that

$$\Pr_{\mathcal{O},a,k}[B_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(A_0^{\mathcal{O}}, a, k) \text{ outputs } (m,t) \text{ s.t. } \mathcal{O}(a,k,m) = t] \geq \varepsilon/2,$$

where B_1 (just like A_1) does not query m to its second oracle.

We show a deterministic encoding for $\mathsf{Func}([K]\times[N]\times[L],[M])$. Fix $\mathcal{O}$ in that set, and let $U_{\mathcal{O}}$ be the set of (a,k) for which B_1 succeeds, i.e.,

$$U_{\mathcal{O}} = \{(a,k) \,:\, B_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(A_0^{\mathcal{O}}, a, k) \text{ outputs } (m,t) \text{ s.t. } \mathcal{O}(a,k,m) = t\}.$$

Let $G_{\mathcal{O}}$ be a subset of $U_{\mathcal{O}}$ such that for every $(a,k) \in G_{\mathcal{O}}$ it holds that $B_1^{\mathcal{O},\mathcal{O}(a,k,\cdot)}(A_0^{\mathcal{O}}, a, k)$ does not query its first oracle on any point with prefix $(a',k') \in G_{\mathcal{O}}$. Because B_1 makes at most T queries, there exists such a set $G_{\mathcal{O}}$ with size at least $|U_{\mathcal{O}}|/(T+1)$. We encode $\mathcal{O}$ as follows.

1. Include $A_0^{\mathcal{O}}$, $|G_{\mathcal{O}}|$, and a description of $G_{\mathcal{O}}$ using $S + \log KN + \log\binom{KN}{|G_{\mathcal{O}}|}$ bits.

2. For each $(a,k) \in ([K] \times [N]) \setminus G_{\mathcal{O}}$ (in lexicographic order), include the truth table of $\mathcal{O}(a,k,\cdot)$. This uses a total of $(KN - |G_{\mathcal{O}}|) \cdot L \log M$ bits.
3. For each $(a,k) \in G_{\mathcal{O}}$ (in lexicographic order), include in the encoding the answers to all the oracle queries made by B_1 to its second oracle in the order they are made; denote the set of such queries by X, and let (m,t) be the output of B_1. Also include in the encoding $\mathcal{O}(a,k,m')$ for all $m' \notin X \cup \{m\}$. This requires $|G_{\mathcal{O}}| \cdot (L-1) \log M$ bits.

Decoding is done in the obvious way. The encoding length (in bits) is at most

$$\begin{aligned} & S + \log KN + \log \binom{KN}{|G_{\mathcal{O}}|} + KNL \log M - |G_{\mathcal{O}}| \log M \\ & \le S + \log KN + |G_{\mathcal{O}}| \cdot \log \left(\frac{eKN}{|G_{\mathcal{O}}|} \right) + KNL \log M - |G_{\mathcal{O}}| \log M. \end{aligned}$$

The expected encoding length (taken over $\mathcal{O} \in \mathsf{Func}([K] \times [N] \times [L], [M])$) is thus at most

$$S + \log KN + \mathbb{E}[\,|G_{\mathcal{O}}|\,] \cdot \log \left(\frac{eKN}{M \cdot \mathbb{E}[\,|G_{\mathcal{O}}|\,]} \right) + KNL \log M,$$

using the log-sum inequality. But Lemma 1 shows that any such encoding must have expected length at least $KNL \log M$ bits. We thus conclude that

$$S + \log KN \ge \mathbb{E}[\,|G_{\mathcal{O}}|\,] \cdot \log \left(\frac{M \cdot \mathbb{E}[\,|G_{\mathcal{O}}|\,]}{eKN} \right) \ge \frac{\varepsilon KN}{2(T+1)} \cdot \log \left(\frac{M\varepsilon}{2e(T+1)} \right),$$

where the second inequality uses the facts that $y \log y$ is an increasing function for $y \ge 1$ and

$$\mathbb{E}[\,|G_{\mathcal{O}}|\,] \ge \mathbb{E}\left[\frac{|U_{\mathcal{O}}|}{T+1} \right] \ge \frac{\varepsilon KN}{2(T+1)}.$$

This implies Theorem 8 since either $\varepsilon \le \frac{4e(T+1)}{M}$, or else it must be the case that $\varepsilon \le \frac{2(S+\log KN)(T+1)}{KN} = O(\frac{ST}{KN} + \frac{T \log N}{N})$. ■

Acknowledgments

Jonathan Katz thanks Christine Evangelista, Aaron Lowe, Jordan Schneider, Lynesia Taylor, Aishwarya Thiruvengadam, and Ellen Vitercik, who explored problems related to salting and rainbow tables as part of an NSF-REU program in the summer of 2014.

References

1. Noga Alon, Oded Goldreich, Johan Håstad, and Rene Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

2. Elad Barkan, Eli Biham, and Adi Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In *Advances in Cryptology—Crypto 2006*, volume 4117 of *LNCS*, pages 1–21. Springer, 2006.
3. Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro. Multi-instance security and its application to password-based cryptography. In *Advances in Cryptology—Crypto 2012*, volume 7417 of *LNCS*, pages 312–329. Springer, 2012.
4. Mihir Bellare and Phil Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conf. on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
5. Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In *Advances in Cryptology—Asiacrypt 2013, Part II*, volume 8270 of *LNCS*, pages 321–340. Springer, 2013.
6. Kai-Min Chung, Huijia Lin, Mohammad Mahmoody, and Rafael Pass. On the power of nonuniformity in proofs of security. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 389–400. ACM, 2013.
7. Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In *Advances in Cryptology—Crypto 2010*, volume 6223 of *LNCS*, pages 649–665. Springer, 2010. Version dated February 17, 2010 available at `http://ttic.uchicago.edu/~madhurt/Papers/dtt-new.pdf`.
8. Yevgeniy Dodis and John P. Steinberger. Message authentication codes from unpredictable block ciphers. In *Advances in Cryptology—Crypto 2009*, volume 5677 of *LNCS*, pages 267–285. Springer, 2009.
9. Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM Journal on Computing*, 29(3):790–803, 1999.
10. Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Computing*, 35(1):217–246, 2005.
11. Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 305–313. IEEE, 2000.
12. Martin Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Information Theory*, 26(4):401–406, 1980.
13. Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *Proc. 26th Annual IEEE Conference on Computational Complexity*, pages 104–114. IEEE Computer Society, 2011.
14. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, 2nd edition*. Chapman & Hall/CRC Press, 2014.
15. Mohammad Mahmoody and Ameer Mohammed. On the power of hierarchical identity-based encryption. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 243–272. Springer, 2016.
16. Robert Morris and Ken Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.
17. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Advances in Cryptology—Crypto 2003*, volume 2729 of *LNCS*, pages 617–630. Springer, 2003.
18. Phillip Rogaway. Formalizing human ignorance. In *Progress in Cryptology—Vietcrypt 2006*, volume 4341 of *LNCS*, pages 211–228. Springer, 2006.

19. Dominique Unruh. Random oracles and auxiliary input. In *Advances in Cryptology—Crypto 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, 2007.
20. Andrew C.-C. Yao. Theory and applications of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 80–91. IEEE, 1982.
21. Andrew C.-C. Yao. Coherent functions and program checkers. In *22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 84–94. ACM Press, 1990.