Password Hashing and Preprocessing

Pooya Farshim[1] and Stefano Tessaro[2]

[1] University of York, York, UK
pooya.farshim@gmail.com
[2] University of Washington, Seattle, USA
tessaro@cs.washington.edu

Abstract. How does the cryptanalytic effort needed to compromise t out of m instances of hashed passwords scale with the number of users when arbitrary preprocessing information on the hash function is available? We provide a formal treatment of this problem in the *multi-instance setting with auxiliary information*. A central contribution of our work is an (arguably simple) transcript-counting argument that allows us to resolve a fundamental question left open by Bellare, Ristenpart, and Tessaro (BRT; CRYPTO 2012) in multi-instance security. We leverage this proof technique to formally justify unrecoverability of hashed salted passwords in the presence of auxiliary information in the random-oracle model. To this end we utilize the recent pre-sampling techniques for dealing with auxiliary information developed by Coretti et al. (CRYPTO 2018). Our bounds closely match those commonly assumed in practice.

Besides hashing of passwords through a monolithic random oracle, we consider the effect of iteration, a technique that is used in classical mechanisms, such as bcrypt and PBKDF2, to slow down the rate of guessing. Building on the work of BRT, we formulate a notion of KDF security, also in the presence of auxiliary information, and prove an appropriate composition theorem for it.

Keywords. Password hashing, multi-instance security, preprocessing, KDF security.

1 Introduction

Password hashing plays a central role in the design of secure systems. We store a password hash $H(pw)$ in lieu of a password pw for authentication purposes. Moreover, whenever key-management is too complex (e.g., in hard-drive encryption), one typically uses $H(pw)$ as a *secret key*. Generally, one assumes that the hash function is by itself secure in a standard cryptographic sense, and the real threat are attacks which exploit the limited entropy of humanly generated passwords and only evaluate the hash function in the *forward* direction on a sequence of password guesses. Several approaches have been adopted to make this task as hard as possible – these typically consist of making the computation of H as expensive as acceptable (e.g., via *iteration*, as in PKCS#5 [Kal00] or bcrypt [PM99], or by making the computation memory hard as in e.g., [PJ16, AS15]).

Our contributions, in a nutshell. This paper focuses on a crucial aspect of password-cracking attacks, namely the role of *pre-processing*. For example, *rainbow tables* [Oec03] are a well-known type of data structures that help speed up password-cracking attacks. The common wisdom is that *salting* defeats such pre-processing – one uses $H(sa, pw)$ instead of $H(pw)$, for a fresh salt sa. Indeed, recent works (cf. e.g. [Unr07, DGK17, CDGS18, CDG18]) analyze the security of random oracles under auxiliary information, and partially validate the benefits of salting.

Still, these results do not consider important aspects which are specific to password hashing. First, they focus on protecting a *single* password. Bellare, Ristenpart, and Tessaro (BRT) [BRT12] however point out that the security study of password hashing must consider *multi-instance security metrics*, to ensure that the hardness of password cracking grows with the number of passwords under attack. Second, existing results focus on monolithic random oracles, as opposed to constructions using them (e.g., by iterating them). Third, they focus on cryptographic hardness for uniformly chosen secrets, as opposed to using arbitrary distributions, with correlations across instances.

In this paper, we address all of the above, and extend the provable-security treatment of password-hashing (following BRT) to the pre-processing setting. On the way, of independent interest, we resolve open problems in the characterization of the hardness of password distributions via guessing games. We elaborate on our contributions next.

1.1 Guessing games

The first set of contributions are independent of pre-processing, and revisit password-recovery hardness *metrics* in the multi-instance setting. For example, consider a vector $\mathbf{pw} = (\mathbf{pw}[1], \ldots, \mathbf{pw}[m])$ of m passwords sampled from a distribution $\mathcal{P}$, and our aim is to guess *all* of them. Then, the optimal guess is the *most likely* vector $\mathbf{pw}^*$ output by $\mathcal{P}$, and the success probability is captured *exactly* by the *min-entropy* $\mathbf{H}_\infty(\mathcal{P})$.

It is not immediately clear whether min-entropy, however, is a good metric in the password-hashing setting – there, one is additionally given m hashes

$$H(\mathbf{sa}[1], \mathbf{pw}[1]), \ldots, H(\mathbf{sa}[m], \mathbf{pw}[m]) \ , \tag{1}$$

where $\mathbf{sa}$ is a public vector of salts – which we assume to be distinct and sufficiently long for this discussion – and is asked to recover *the entire vector* $\mathbf{pw}$. The availability of the hashes themselves allows for verification of individual password guesses. Following [BRT12], this can be abstracted as an *interactive password guessing game* which initially samples $\mathbf{pw} \twoheadleftarrow \mathcal{P}$, and the adversary can issue user-specific queries $\textsc{Test}(i, pw)$, and learn whether or not $\mathbf{pw}[i] = pw$. The adversary wins if a query $\textsc{Test}(i, \mathbf{pw}[i])$ is made for *every* $i \in [m]$.

BRT suggest that the best probability of winning this game with a given budget T of Test queries—which we denote as $\mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T)$—is by itself a

good metric of hardness for password distributions. However, it is very hard to evaluate. Our first contribution is a bound of the form

$$\mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T) \leq \left(\frac{eT}{m}\right)^m \cdot 2^{-\mathbf{H}_\infty(\mathcal{P})} \ , \tag{2}$$

which only depends on the min-entropy of the distribution $\mathcal{P}$. This resolves the main open question of [BRT12], which only gave a bound for the case where (1) the passwords are *independent* and (2) we know separate and a-priori fixed bounds T_i on the number of $\textsc{Test}(i, \cdot)$ queries for each $i \in [m]$. We note that (2) is a strong assumption, since an optimal attacker generally stops using queries for a particular password when successful. For the case where the passwords are drawn independently from a set of size N, for example, our bound is $\left(\frac{eT}{mN}\right)^m$, which is clearly tight (the optimal strategy makes T/m distinct guesses for each $i \in [m]$).

In fact, our framework studies a more general metric $\mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\mathsf{Gen}}(T, c)$ which considers a general salt generator Gen (which may generate colliding salts, or salts with low entropy), allows for a password to be re-used across ℓ salts, and enables the adversary to learn c passwords via *corruption queries*. On special case of interest is that of *no salts*, i.e., $\mathsf{Gen} = \perp$ outputs m empty strings as salts (and thus $\ell = 1$ without loss of generality). Here, we prove that

$$\mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},1,\perp}(T) \leq T^m \cdot 2^{-\mathbf{H}_\infty(\mathcal{P})} \ . \tag{3}$$

While this bound appears natural, the main technical challenge in the proof (which exploits a combinatorial counting argument) is to deal with distributions which yield collisions across passwords. It is worth noticing here that the effort needed to compromise all passwords, for example, increases by a factor of m – this is because every individual query can be helpful to guess any of the passwords.

1.2 Unrecoverability bounds

We then turn to our first contribution in the pre-processing model. Here, we give tight bounds on the success probability of recovering all of $\mathbf{pw}$ in the setting of (1), when H is a random oracle with n-bit outputs (in the following, we let $N = 2^n$) to which the adversary can issue T queries, and where additionally the adversary is given S bits of pre-processed information about the random oracle H. This model is often referred to as the *random-oracle model with auxiliary information*, or AI-ROM for short. This generalizes in particular prior works on studying one-wayness in the AI-ROM [GT00, Wee05, Unr07, DTT10, DGK17, CDGS18] in that we consider both general pre-image distributions as well as (most importantly) *multi-instance* security.

Our analyses rely on a reduction to the bounds for guessing games discussed above, combined with the *bit-fixing random oracle model* (BF-ROM) of [Unr07, CDGS18]. In the BF-ROM, one analyzes unrecoverability in a setting where P input-output pairs of the random oracle are chosen arbitrarily (the rest of the random oracle is truly random) – here, P is a parameter. In [CDGS18] it is shown

that replacing P with (roughly) ST, and multiplying the recovering probability by 2, gives a corresponding AI-ROM bound. We will slightly relax this paradigm, and realize that setting $P = ST/m$, while multiplying the probability by 2^m, allows us to obtain the right bound.

Unrecoverability in the unsalted case. For example, for the case where passwords are chosen uniformly from a set of size N (which is equal to the output size of the random oracle), we show that if no salts are used, then the probability of recovering all passwords is of order (assuming S is large enough)

$$\left(\frac{ST}{mN}\right)^m .$$

We also obtain a corresponding bound for arbitrary distributions. This is interesting, because it means that if we want to recover all passwords with probability *one*, then we need to invest time $T = mN/S$, in other words, the complexity of recovering multiple passwords *without* salts, even given a rainbow tables, grows linearly in the number of passwords. This is in contrast to the setting *without* auxiliary information, where in time $T = N$ we can recover essentially any number of passwords. In Appendix A, we give a self-contained and straightforward extension of Hellman's space-time trade-off for multi-instance security, where one can exactly see that computation needed to break one password cannot be recycled for another one.

Unrecoverability in the salted case. For the case with salts, in contrast, the expectation is that the S bits of pre-processing is not helpful. We confirm this via a bound, which depends on the salt size K, and prove that the probability of recovering all m passwords is roughly

$$\left(\frac{T}{N}\right)^m + \frac{m\ell}{K}\left(\frac{eST}{m^2N}\right)^m + \frac{m^2}{K}$$

for the special case of passwords drawn independently from an N-element set. (Again, our final bound is more general.) In other words, the first term becomes the leading one if K is sufficiently large.

1.3 AI-KDF security of iteration

Finally, we consider a simulation-based notion of KDF security which extends the notion introduced by BRT [BRT12] to the AI-ROM setting. The basic idea of KDF security is to see the functionality provided by a key-derivation function as providing m keys to honest users and giving an attacker the ability to test passwords guess for correctness. Thus, this notion requires that for any (real-world) attacker against a KDF function with salted passwords and auxiliary information there is a simulator against the ideal functionality that has essentially the same advantage. This notion in particular justified the use of password-based KDFs to replace uniform keys with those derived from (salted) passwords.

BRT's notion was inspired by the indifferentiability framework [MRH04, CDMP05], and restricts it to a particular ideal KDF functionality. Directly using indifferentiability as the target security notion, however, suffers from two drawbacks: the indifferentiability of iterated constructions (from RO) in general, remain unclear. Second, concrete security bounds for indifferentiability, due to the attack surface exposed, often fall short of providing the guarantees that are needed in practice to provably set salt sizes.

To prove our result, we first formulate a notion of KDF security in the bit-fixing random-oracle model. In this model we can present a simulator that simulates the primitive oracle by looking for about to complete chains of length $r-1$, where r in the iteration count, and if so, using the Test oracle provided in the ideal game to handle the query. As with BRT, we require that when the number of queries made by the adversary to the primitive oracle is T, the number of queries made by the simulator to its Test oracle is only T/r. This restriction will allow us to conclude that in applications an adversary needs to place r queries to the random oracle to check whether or not a candidate guess for a password was correct. Finally, we lift this BF-ROM result to the AI-RO model using [CDGS18, Theorem 5].

Applications. We prove a composition theorem for AI-KDF security for a range of games beyond IND-CPA security as considered by BRT and extended to encompass a preprocessing phase. This result shows that uniform values used in a game can be replaced with those derived from *salted* passwords via an AI-KDF-secure function, even in the presence of preprocessing. This result can thus be seen as formal justification of "salting defeats preprocessing in password-based cryptography." As with BRT, our simulation-based notion allows us to reduce security to the full difficulty of the multi-instance password-guessing game.

1.4 Structure of the paper

We start by recalling the necessary preliminaries in Section 2. We define our basic measures of unguessability of passwords in Section 3, where we establish our basic bounds. In Section 4 we define unrecoverability of hashed passwords and relate them to unguessability, both in the salted and unsalted settings. In Section 5 we study iterated hashing under KDF security in the presence of auxiliary information and show how to apply this result (and in general KDF security) to securely replace uniform keys with password-derived ones in various applications in the presence of preprocessing.

2 Preliminaries

Notation. Throughout the paper $\mathbb{N}$ denotes the set of nonnegative integers including zero, $\{0,1\}^n$ the set of all bit strings of length n, and $\{0,1\}^*$ denotes the set of all finite-length bit strings, and ε the empty string. For two bit-strings X

and Y, $X|Y$ denotes string concatenation and (X, Y) denotes a uniquely decodable encoding of X and Y. The length of a string X is denoted by $|X|$. For a finite, non-empty set S we write $s \twoheadleftarrow S$ to mean that s is sampled uniformly at random from S. Overloading the notion, for a randomized algorithm A with input(s) x we write $y \twoheadleftarrow A(x)$ to mean that y is sampled from the outputs of A according to the distribution induced by running A on uniform random coins. We denote adversarial procedures, which may be randomized and/or stateful, by $\mathcal{A}$, honest stateless procedures with C, and honest stateful procedures with $\mathcal{S}$.

FACTORIALS AND FRIENDS. Recall that $(n/3)^n \leq n! \leq \mathrm{e} \cdot (n/2)^n$ and that $n! \sim \sqrt{2\pi n}(n/\mathrm{e})^n$. Further, for all $1 \leq m \leq T$,

$$\binom{T}{m} \leq \frac{T^m}{m!} < \left(\frac{\mathrm{e}T}{m}\right)^m \ .$$

We let $(T)_m := \binom{T}{m} m!$ denote the falling factorial. The Stirling numbers of the second kind $\left\{ {m \atop k} \right\}$ count the number of partitions of a set of size m into k non-empty sets. For these numbers we have,

$$\sum_{k=0}^{m} \left\{ {m \atop k} \right\} (T)_k = T^m \qquad \text{and} \qquad \left\{ {m \atop k} \right\} \leq \binom{m}{k} k^{m-k} \ .$$

We also have that $\left\{ {m \atop 0} \right\} = 0$ for $m \geq 1$.

THE RO MODEL. We denote the set of all functions from a domain D to a finite range R by $\mathrm{Fun}(D, R)$. The random-oracle model $\mathsf{RO}(D, R)$ is a model of computation where parties are given oracle access to a uniformly random function $\mathsf{H} \twoheadleftarrow \mathrm{Fun}(D, R)$.[3] We denote an adversary $\mathcal{A}$ with access to H by $\mathcal{A}^{\mathsf{H}}()$.

THE BF-RO MODEL. An assignment (or pre-set) list L is a list of pairs of points $(x, y) \in D \times R$ that respects the property of defining a function, i.e., for each x there is at most one y such that $(x, y) \in L$. For $P \in \mathbb{N}$, the *bit-fixing* random-oracle model $\mathsf{BF\text{-}RO}(P, D, R)$ grants oracle access to a uniformly chosen random function $\mathsf{H} \twoheadleftarrow \mathrm{Fun}(N, M)$ compatible with L, where $(\sigma, L) \twoheadleftarrow \mathcal{A}_0()$ is chosen by an initial (aka. offline or preprocessing) adversary $\mathcal{A}_0()$ and has size at most P. Note that $\mathcal{A}_0$ does *not* get access to H. We denote the online phase of the attack by $\mathcal{A}_1^{\mathsf{H}}(\sigma)$, which gets access to H and σ, the information passed from $\mathcal{A}_0$. Thus, σ does not depend on H. We use $\mathsf{H}[L]$ for a random oracle conditioned on L. Note that in the BF-RO model, there is no upper bound on the size of σ.

THE AI-RO MODEL. For $P \in \mathbb{N}$, the *auxiliary-input* (AI) random-oracle model $\mathsf{AI\text{-}RO}(S, D, R)$ grants oracle access to a random function $\mathsf{H} \twoheadleftarrow \mathrm{Fun}(D, R)$ together with oracle-dependent auxiliary information $\sigma \twoheadleftarrow \mathcal{A}_0(\mathsf{H})$, where $|\sigma| \leq P$. We denote the online phase by $A_1^{\mathsf{H}}(\sigma)$.

[3] Note the distribution is well-defined when $D = \{0,1\}^*$ and can be formalized in the language of measure theory.

GAMES. A game is a randomized stateful algorithm $(y, st) \leftarrow \mathrm{G}(x; r; st)$, where x is the input, r the randomness, and st the state, which is initialized to ε. The output y is returned to a stateful adversary $(x, st') \twoheadleftarrow \mathcal{A}_1(y, st')$, which then calls the game x and so on. This interaction terminates by the game returning a flag win indicating win/loss. For indistinguishability games, the advantage metric takes the form $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G}}(\mathcal{A}_1) := 2 \cdot \Pr[\mathsf{win}] - 1$ and for unpredictability games it takes the form $\mathbf{Adv}^{\mathrm{pred}}_{\mathrm{G}}(\mathcal{A}_1) := \Pr[\mathsf{win}]$. We can lift this definition to ideal models of computation by sampling a random oracle $\mathsf{H} \twoheadleftarrow \mathrm{Fun}(D, R)$, which both G and $\mathcal{A}_1$ can access, and passing auxiliary information computed by $\mathcal{A}_0$, as described above, to $\mathcal{A}_1$. We define $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G},\mathsf{AI\text{-}RO}}(S, T)$ by taking the maximum of $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G}}(\mathcal{A}_0, \mathcal{A}_1)$ over all $(\mathcal{A}_0, \mathcal{A}_1)$ that output at most S bits of auxiliary information and place at most T queries to the random oracle. Unpredictability advantages and advantages in the BF-RO model are defined analogously.

Coretti, Dodis, Guo, and Steinberger [CDGS18, Theorems 5 and 6] prove the following result, which bounds adversarial advantage in the AI-RO model in terms of that in the BF-RO model.

Theorem 1 ([CDGS18, Theorems 5 and 6]). *For any $P \in \mathbb{N}$ and any $\gamma > 0$ and any game* G *in the AI-RO model,*

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G},\mathsf{AI\text{-}RO}}(S, T) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G},\mathsf{BF\text{-}RO}}(P, T_{\mathrm{G}} + T) + \frac{(S + \log \gamma^{-1}) \cdot (T_{\mathrm{G}} + T)}{P} + \gamma \ ,$$

where T_{G} is the query complexity of G*. Furthermore, for any unpredictability game* G*,*

$$\mathbf{Adv}^{\mathrm{pred}}_{\mathrm{G},\mathsf{AI\text{-}RO}}(S, T) \leq 2^{(S + \log \gamma^{-1}) \cdot (T_{\mathrm{G}} + T)/P} \cdot \mathbf{Adv}^{\mathrm{pred}}_{\mathrm{G},\mathsf{BF\text{-}RO}}(P, T_{\mathrm{G}} + T) + \gamma \ .$$

3 Unguessability

PASSWORD SAMPLERS. A password sampler is a randomized algorithm $\mathcal{P}$ that takes no input and outputs a vector of passwords $\mathbf{pw} = (\mathbf{pw}[1], \ldots, \mathbf{pw}[m])$ and some leakage z on the passwords.[4] We assume that $\mathcal{P}$ always outputs the same number of passwords m. To make this explicit we call the sampler an m-sampler. We note that password samplers in our work do not get access to the random oracle.[5]

A basic measure of the unguessability of a password sampler is its min-entropy. We consider an average-case notion over leakage z, as this will not be under the control of the adversary:

$$\tilde{\mathbf{H}}_{\infty}(\mathcal{P} \mid \mathcal{Z}) := -\log \mathbb{E}_{\mathcal{Z}}(2^{-\mathbf{H}_{\infty}(\mathcal{P} \mid \mathcal{Z} = z)}) \ ,$$

[4] This is *not* the preprocessing information, only some partial information related to passwords.

[5] In this work we do not consider password samplers that have oracle access to an ideal primitive.

Here $\mathcal{Z}$ denotes the random variable corresponding to the leakage.

UNGUESSABILITY. Following [BRT12], we consider a guessability game which allows for testing and adaptive corruptions of guessed passwords. The goal of the adversary is to guess *all* passwords. More precisely, for an adversary $\mathcal{A}$ we define

$$\mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T, c) := \max_{\mathcal{A}} \Pr\left[\textsc{Guess}^{\mathcal{A}}_{\mathcal{P}}\right] \ ,$$

where game $\textsc{Guess}^{\mathcal{A}}_{\mathcal{P}}$ is defined in Fig. 1 and the maximum is taken over all $\mathcal{A}$ that place at most T queries to TEST, and at most c queries to COR. Observe that the test oracle takes an index i. This signifies the fact that guesses are *user-specific* and thus cannot be amortized over all users. Our definition also includes some side information z for generality, which was not present in BRT.

Game $\textsc{Guess}^{\mathcal{A}}_{\mathcal{P}}$:	Proc. $\textsc{Test}(i, pw)$:	Proc. $\textsc{Cor}(i)$:
$(\mathbf{pw}, z) \twoheadleftarrow \mathcal{P}$	$\mathsf{win}_i \leftarrow (pw = \mathbf{pw}[i])$	$\mathsf{win}_i \leftarrow \mathsf{true}$
$y \twoheadleftarrow \mathcal{A}^{\textsc{Test},\textsc{Cor}}(z)$	return win_i	return $\mathbf{pw}[i]$
return $\bigwedge_{i=1}^{m} \mathsf{win}_i$		

Fig. 1. The guessing game.

SIMPLIFYING ASSUMPTIONS. We assume, wlog, that $\mathcal{A}$ does not call $\textsc{Cor}(i)$ on any index i for which TEST returned true. Similarly, we assume that $\mathcal{A}$ does not call TEST with an index i which was queried to $\textsc{Cor}(i)$. Moreover, for $c' \leq c \leq m$, any adversary $\mathcal{A}$ that places c' queries to COR can be transformed to an adversary $\mathcal{B}$ that places $c \geq c'$ queries to COR without any loss in advantage. [6] Thus, throughout the paper we may assume, wlog, that adversaries that place at most c corrupt queries place exactly c corrupt queries. Furthermore, the set of corrupted indices and those for which TEST was successful are disjoint.

A BASIC MEASURE OF UNGUESSABILITY. It follows from the definitions of unguessability and average-case min-entropy that

$$\mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m, 0) = 2^{-\tilde{\mathbf{H}}_\infty(\mathcal{P}|\mathcal{Z})} \ .$$

To see this, note that given any $\mathcal{P}$ and any $\mathcal{A}$ in GUESS we can build an adversary $\mathcal{B}$ with *no* oracle access as follows. $\mathcal{B}$ runs $\mathcal{A}$ on the leakage that it receives and

[6] Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and forwards its TEST and COR queries to its own respective oracles. It stops $\mathcal{A}$ from making further queries to TEST once $m - c$ queries to TEST return true. $\mathcal{B}$ corrupts the remaining indices. By the previous assumption, the passwords for these indices have not been found due to a corrupt or a test query so far. Clearly, the number of TEST queries of $\mathcal{B}$ is less than those of $\mathcal{A}$. Further $\mathcal{B}$ places in total at most c queries to COR. To see this let c'_0 be the number of corrupt queries by $\mathcal{A}$ when it was stopped. At this point $c'_0 + m - c$ passwords were found. Hence there are $m - (c'_0 + m - c) = c - c'_0$ passwords that were not found at that stage. Thus, $\mathcal{B}$ places $c'_0 + (c - c'_0) = c$ queries to its COR oracle after $\mathcal{A}$ was stopped.

answers all its TEST queries with true, and outputs the set of passwords queried to TEST sorted according to their index. For all password vectors where $\mathcal{A}$ is successful, these queries are answered correctly. Thus, $\mathcal{B}$ runs $\mathcal{A}$ perfectly in the environment that it expects. Thus, $\mathcal{B}$ is successful whenever $\mathcal{A}$ is. Inequality in the opposite direction is trivial and we obtain the result. We note that in the presence of adaptive corruptions, it is unclear how to relate the above game-based notion of unguessability to a standard information-theoretic notion. Thus,

$$\mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m-c,c)$$

will form our basic measure of unguessability of passwords, which we call c-unguessability.

THRESHOLD SECURITY. Consider a *threshold* notion of unguessability whereby the adversary's goal is to guess t out of m of the passwords, while having access to TEST oracle, but no longer a COR oracle. The advantage of any such adversary is easily upper bounded by the advantage of an adversary that before termination simply corrupts the $m-t$ users for which recovery was not attempted. Thus,

$$\mathbf{Adv}^{t\text{-guess}}_{\mathcal{P}}(T) \leq \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T, m-t) \ .$$

In the reverse direction, however, an inequality does not hold in general.[7] However, when the password sampler is a *product sampler* in the sense that for some $\mathcal{P}_i$ it computes $(\mathbf{pw}[i], z_i) \twoheadleftarrow \mathcal{P}_i$ (run on independent coins) and returns the vector $((\mathbf{pw}[1], \ldots, \mathbf{pw}[m]), z)$ where $z := (z_1, \ldots, z_m)$, we have

$$\mathbf{Adv}^{\text{guess}}_{\otimes \mathcal{P}_i}(T, c) \leq \mathbf{Adv}^{(m-c)\text{-guess}}_{\otimes \mathcal{P}_i}(T) \ .$$

Using an argument similar to [BRT12, Lemma F.1] the independence of the passwords allows for a perfect simulation of COR(i). Run the corrupting adversary and answer its test oracle using its own test oracle. When a COR(i) query is placed, return a value distributed according to $\mathcal{P}_i$, conditioned on auxiliary information z_i and output not matching any pw for which TEST(i, pw) previously responded with false.

Thus for product samplers, recovering t out of m passwords without corruptions is equivalent to recovering t passwords while corrupting $m-t$ of them. Since for non-product distributions the corrupting adversaries are stronger, we work with such adversaries.[8]

Our first result relates the unguessability of passwords to c-unguessability.

Theorem 2 (Unguessability). *For any m-sampler $\mathcal{P}$ and any $T, c \in \mathbb{N}$,*

$$\mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T, c) \leq \binom{T}{m-c} \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m-c, c) \ .$$

[7] Consider m identical passwords, where the common password is uniformly distributed within a large set. With corruptions, guessing the common value is trivial; without, it can be done with small probability.

[8] Note however that for correlated passwords, this may imply that there is no security, whereas in practice we would like to argue that there is some security. We leave the treatment of this intermediate notion to future work.

Proof. By our simplifying assumptions, $\mathcal{A}$ makes exactly c corrupt queries and of the T test queries, exactly $m - c$ are successful. Consider an adversary $\mathcal{B}$ against GUESS that runs $\mathcal{A}$ as follows. At the onset, $\mathcal{B}$ guesses which of the $m - c$ queries among the T queries will result in true. There are $\binom{T}{m-c}$ such choices. It then runs $\mathcal{A}$ and answers all TEST queried for the guessed indices with true. The corruption queries are relayed. If the set of indices guessed is correct, the algorithm $\mathcal{B}$ runs $\mathcal{A}$ perfectly. We obtain the claim inequality by maximizing over $\mathcal{A}$. □

Despite its simplicity, this argument is novel and in particular resolves a problem left open by BRT on upper bounds for guessability with only a *global* bound on the total number of TEST queries (and not a priori bounds on the number of user-specific guesses, as treated by BRT).

SALTED GUESSABILITY. Following BRT, we consider an extension of GUESS that incorporates *salts*. We allow for multiple salts per password (as required in applications such as password-based encryption) and consider a TEST oracle which is *salt-specific* rather than user-specific. This test procedure thus *amortizes guessing* over all users who share a salt value. The rationale for this choice is that given salted *hashes* of passwords—whose security we will be ultimately analyzing—once a password is recovered, it is also recovered for all users for which password-salt pairs match. Crucially, a *single* query is needed to deduce this information.

Our formal definition, which is shown in Fig. 2, differs from that of BRT in a number of aspects. First, we allow for an arbitrary salt-generation algorithm Gen that takes a user index i and a counter j. (This choice allows for stateful generation of salts, which may be possible in certain contexts.) Second, under TEST we set win_i to true for *all* matching i, rather than only the first i for which a match is found as in BRT. This ensures that password-salt collisions do not result in an unwinnable game. We also release the set of all passwords indices for which password-salt matches the query (rather than the first such index). This choice more closely matches the setting of hashed passwords. Finally, we leak the *collision pattern* of the password-salt pairs. This is formalized via an algorithm $\mathsf{Colls}(\mathbf{pw}, \mathbf{sa})$ that takes a vectors of passwords $\mathbf{pw}$ of length m and an $m \times \ell$ matrix of salts and returns an $m\ell \times m\ell$ matrix whose $((i_1, j_1), (i_2, j_2))$ entry is set to 1 iff $(\mathbf{pw}[i_1], \mathbf{sa}[i_1, j_1]) = (\mathbf{pw}[i_2], \mathbf{sa}[i_2, j_2])$. We define the advantage of (T, c)-adversaries analogously to the GUESS game.

A direct reduction shows that for any password sampler $\mathcal{P}$, any salt-sampler Gen, and any number of salts per password ℓ,

$$\mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T, c) \leq \mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\mathsf{Gen}}(T, c) \ .$$

In order to prove a result in the opposite direction, for a salt sampler Gen, we define

$$\mathbf{Coll}_{\mathsf{Gen}}(m, \ell) := \Pr[\exists (i, j) \neq (i', j') \in [m] \times [\ell] : \mathsf{Gen}(i, j) = \mathsf{Gen}(i', j')]$$

Game SA-GUESS$^{\mathcal{A}}_{\mathcal{P},\ell,\mathsf{Gen}}$:	Proc. TEST(pw, sa):	Proc. COR(i):
$(\mathbf{pw}, z) \twoheadleftarrow \mathcal{P}$	$S \leftarrow \{i : \exists j (pw, sa) =$	$\mathsf{win}_i \leftarrow \mathsf{true}$
for $(i,j) \in [m] \times [\ell]$ do	$= (\mathbf{pw}[i], \mathbf{sa}[i,j])\}$	return $\mathbf{pw}[i]$
$\quad \mathbf{sa}[i,j] \twoheadleftarrow \mathsf{Gen}(i,j)$	for $i \in S$ do	
$z_{\mathsf{coll}} \leftarrow \mathsf{Colls}(\mathbf{pw}, \mathbf{sa})$	$\quad \mathsf{win}_i \leftarrow \mathsf{true}$	
$y \twoheadleftarrow \mathcal{A}^{\text{TEST},\text{COR}}(\mathbf{sa}, z, z_{\mathsf{coll}})$	return S	
return $\left(\bigwedge_{i=1}^{m} \mathsf{win}_i\right)$		

Fig. 2. The password-guessing game with salts where the collision pattern of password-salt pairs is always leaked. win_i are initialized to false.

as the probability of obtaining $m\ell$ distinct salts. For uniform salts in $[K]$,

$$\mathbf{Coll}_{\mathsf{Gen}}(m,\ell) = 1 - \frac{K!}{K^{m\ell}(K-m\ell)!} \leq \frac{m^2\ell^2}{K} \ .$$

We note that in some settings, the distinctness of salts may be guaranteed. For instance, by appending (i,j) to salts, where i is a "user-id" and j is an application-specific "session-id," one can guarantee distinctness. As we shall see, to defeat preprocessing the salts must also be unpredictable. Hence (i,j,sa) for a random $sa \twoheadleftarrow [K]$ can be used in these settings.

The next theorem relates salted unguessability of passwords to their user-specific guessability.

Theorem 3. *For any m-sampler $\mathcal{P}$, any Gen, and any $\ell, T, c \in \mathbb{N}$,*

$$\mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\mathsf{Gen}}(T,c) \leq \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T,c) + \mathbf{Coll}_{\mathsf{Gen}}(m,\ell) \ .$$

Proof. Given a (T,c)-adversary $\mathcal{A}$ against SA-GUESS we build a (T,c)-adversary $\mathcal{B}$ against GUESS as follows. Algorithm $\mathcal{B}(z)$ picks $m\ell$ salts via $\mathsf{Gen}(i,j)$ and terminates if the salts are not distinct. Algorithm $\mathcal{B}$ sets z_{coll} to be the identity matrix (if the salts do not collide, certainly password-salt pairs will not), runs $\mathcal{A}(\mathbf{sa}, z, z_{\mathsf{coll}})$ and answers its corrupt queries using its own equivalent oracle. TEST(pw, sa) queries are handled by first checking if $sa = \mathbf{sa}[i,j]$ for some (i,j). If not, $\mathcal{B}$ returns $\perp$; else it finds the *unique* (i,j) such that $sa = \mathbf{sa}[i,j]$. Such an index pair is unique due to the distinctness of salts. Algorithm $\mathcal{B}$ then queries TEST(i, pw) and returns $S := \{i\}$ if it receives true, and the empty set otherwise. (We note that the loss is additive, rather than multiplicative, since $\mathcal{A}$ might be successful exactly when there is a collision among the salts.) □

THE UNSALTED SETTING. Unsalted hashing of passwords is interesting from both a historical and theoretical point of view. Unsalted unguessability is closely linked to *amplification of hardness*. Second, unguessability of passwords without salts constitutes a "worst-case" scenario and can be used to upper-bound unguessability with respect to *any* other salt generator.[9] When there are no salts, all

[9] The proof of this fact follows from the observation that the collision pattern of passwords and the collision pattern of salts (which is publicly available) are sufficient to infer the collision pattern of password-salt pairs.

passwords fall under a single (empty) salt, and in order to check if a candidate password matches any of the sampled passwords a reduction analogous to one given above would need to call $\textsc{Test}(i, pw)$ for *all* $i \in [m]$. This, however, results in a blow up in the number of test queries, which we aim to avoid in this work.

Let us, by a slight abuse of notation, denote the salt generator Gen that always returns ε by $\perp$. We directly prove an upper bound on $\mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\perp}(T, c)$. This extends [BRT12, Theorem 3.2] in two aspects: first, and as mentioned above, the number of queries to $\textsc{Test}(i, \cdot)$ for each index i are no longer a priori fixed. Second, $\mathcal{P}$ no longer comprises independent and identically distributed samples from some base single-password distribution. Proving such a result was left open by BRT.

Theorem 4. *For any m-sampler $\mathcal{P}$ and any $\ell, T \in \mathbb{N}$,*

$$\mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\perp}(T, 0) \leq T^m \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m, 0) \ .$$

Proof. Observe that ℓ does not affect unguessability and thus we may assume, wlog, that $\ell = 1$. We now fix a *deterministic* adversary $\mathcal{B}$ and count the number of vectors $(\mathbf{pw}[1], \ldots \mathbf{pw}[m])$ on which $\mathcal{B}$ *wins*. Call this number N. Then, the final bound will be $N \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m, 0)$.

Consider a vector $(\mathbf{pw}[1], \ldots \mathbf{pw}[m])$ on which $\mathcal{B}$ wins with T queries, and suppose the m passwords are distinct. These can be represented uniquely by a permutation giving the order in which the uncorrupted passwords appear. There are $(m - c)!$ such permutations. There are $\binom{T}{m-c}$ such indices and thus $N = (m - c)!\binom{T}{m-c} = (T)_{m-c}$.

In general, the collision pattern induces a partition of passwords into k groups. Suppose there are no corruptions. Then the number of password vectors $(\mathbf{pw}[1], \ldots \mathbf{pw}[m])$ on which $\mathcal{B}$ wins is at most

$$\left\{ {m \atop k} \right\} \cdot k! \cdot \binom{T}{k} \ ,$$

where $\left\{ {m \atop k} \right\}$ are the Stirling numbers of the second kind. Thus, the total number of representations is at most

$$N \leq \sum_{k=1}^{m} \left\{ {m \atop k} \right\} \cdot (T)_k \ = T^m \ ,$$

where the last equality is by an identity for Stirling numbers. □

We now deal with the general case with corruptions. We consider a non-adaptive guessability game NA-Guess, where corruptions are carried out non-adaptively at the beginning of the game in parallel. This potentially lowers unpredictability advantage and thus strengthens upper bounds using it. Unpredictability in the adaptive game can be bounded by that in the non-adaptive game by guessing at the onset the $\binom{m}{c}$ indices that will be corrupted. However, below we carry a direct reduction to NA-Guess to avoid multiple losses.

Theorem 5. *For any m-sampler $\mathcal{P}$ and $\ell, T, c \in \mathbb{N}$,*

$$\mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\perp}(T-c,c) \leq (T^{m-c} + \mathcal{O}(T^{m-c-1})) \cdot \mathbf{Adv}^{\text{na-guess}}_{\mathcal{P}}(m-c,c) \ .$$

Proof. Once again, wlog, $\ell = 1$. We prove the bound for a modified game where $\textsc{Cor}(i)$ leaks the set of all indices j for which $\mathbf{pw}[j] = \mathbf{pw}[i]$. In this game, wlog, we may assume that $\textsc{Test}$ and $\textsc{Cor}$ oracles return *disjoint* sets that form a partition of $[m]$. This game is equivalent to the unmodified game where the adversary after a corrupt query places a test query on the password just revealed to learn its equality pattern. This results in c additional $\textsc{Test}$ queries.

We now count the number of successful transcripts.

- The number of sets in the partition, $k \geq c$.
- A partition of $[m]$ into k sets: $\left\{ {m \atop k} \right\}$ choices.
- Which c of the k sets will be corrupted: $\binom{k}{c}$ choices. (No ordering of the guesses is needed, since the queried index will be contained in exactly one set.
- The order of the remaining $k-c$ sets that will be returned as responses to $\textsc{Test}$ queries: $(k-c)!$ choices.
- The T test queries which these $k-c$ sets will be responses for: $\binom{T}{k-c}$ choices. (The rest of the queries are answered $\emptyset$.)

Hence,

$$\mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},1,\perp}(T-c,c) \leq \sum_{k=c}^{m} \left\{ {m \atop k} \right\} \binom{k}{c} (k-c)! \binom{T}{k-c} \cdot \mathbf{Adv}^{\text{na-guess}}_{\mathcal{P}}(m-c,c) \ .$$

When $c = 0$, this bound matches that stated in Theorem 4. Using (computer) algebra we have that,

$$\sum_{k=c}^{m} \left\{ {m \atop k} \right\} \binom{k}{c} (k-c)! \binom{T}{k-c} = \sum_{k=0}^{m-c} C(m-c,k,c) \cdot T^{m-c-k} = T^{m-c} + \mathcal{O}(T^{m-c-1}) \ ,$$

where the coefficients $C(n,k,c)$ are defined recursively via

$$C(n,k,c) := \begin{cases} 1 & \text{if } k = 0 \ ; \\ \left\{ {n+c \atop c} \right\} & \text{if } k = n \ ; \\ c \cdot C(n-1,k-1,c) + C(n-1,k,c) & \text{otherwise.} \end{cases}$$

When there is a single corruption ($c = 1$), the sum bounding the advantage has the simple closed form $(T+1)^{m-1}$. □

4 Unrecoverability

We now define two notions of unrecoverability for *hashed* passwords in the AI-RO and the BF-RO models respectively. In the AI-RO model the adversary

Game AI-REC$^{\mathcal{A}_0,\mathcal{A}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}$:	Game BF-REC$^{\mathcal{A}_0,\mathcal{A}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}$:
$\mathsf{H} \twoheadleftarrow \mathrm{Fun}(D,R)$	$\mathsf{H} \twoheadleftarrow \mathrm{Fun}(D,R)$
$\sigma \twoheadleftarrow \mathcal{A}_0(\mathsf{H})$	$(\sigma, L) \twoheadleftarrow \mathcal{A}_0()$
$(\mathbf{pw}, z) \twoheadleftarrow \mathcal{P}$	$(\mathbf{pw}, z) \twoheadleftarrow \mathcal{P}$
for $(i,j) \in [m] \times [\ell]$ do	for $(i,j) \in [m] \times [\ell]$ do
$\quad \mathbf{sa}[i,j] \twoheadleftarrow \mathsf{Gen}(i,j)$	$\quad \mathbf{sa}[i,j] \twoheadleftarrow \mathsf{Gen}(i,j)$
$\quad \mathbf{k}[i,j] \leftarrow \mathsf{KD}^{\mathsf{H}}(\mathbf{pw}[i], \mathbf{sa}[i,j])$	$\quad \mathbf{k}[i,j] \leftarrow \mathsf{KD}^{\mathsf{H}[L]}(\mathbf{pw}[i], \mathbf{sa}[i,j])$
$\mathbf{pw}' \twoheadleftarrow \mathcal{A}_1^{\mathsf{H},\mathrm{COR}}(\mathbf{sa}, \mathbf{k}, \sigma, z)$	$\mathbf{pw}' \twoheadleftarrow \mathcal{A}_1^{\mathsf{H}[L],\mathrm{COR}}(\mathbf{sa}, \mathbf{k}, \sigma, z)$
return $(\mathbf{pw}' = \mathbf{pw})$	return $(\mathbf{pw}' = \mathbf{pw})$
Proc. COR(i):	Proc. COR(i):
return $\mathbf{pw}[i]$	return $\mathbf{pw}[i]$

Fig. 3. The password recoverability games in the AI-RO model (left) and the BF-RO model (right).

can carry out an initial stage of the attack and obtain arbitrary preprocessing information on the entire table of the random oracle. Formally, we define

$$\mathbf{Adv}^{\text{ai-rec}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}(S,T,c) := \max_{\mathcal{A}_0,\mathcal{A}_1} \Pr\left[\text{AI-REC}^{\mathcal{A}_0,\mathcal{A}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}\right] ,$$

where game AI-REC is defined in Fig. 3 (left) and the maximum is taken over all $\mathcal{A}_0$ that output at most S bits of auxiliary information, and all $\mathcal{A}_1$ that place at most T queries to the random oracle and at most c queries to the corrupt oracle.

Similarly, we define

$$\mathbf{Adv}^{\text{bf-rec}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}(P,T,c) := \max_{\mathcal{A}_0,\mathcal{A}_1} \Pr\left[\text{BF-REC}^{\mathcal{A}_0,\mathcal{A}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}\right] ,$$

where game BF-REC is defined in Fig. 3 (right) and the maximum is taken over all $\mathcal{A}_0$ that output a list L of size at most P and all $\mathcal{A}_1$ that place at most T queries to the random oracle and at most c queries to the corrupt oracle.

We start by showing that for any salt generator the BF-RO advantage can be upper bounded by that in the salted unguessability game. Here we will rely on the fact that the collision pattern z_{coll} is known in the SA-GUESS game.

Theorem 6. *Let* $\mathsf{KD}^{\mathsf{H}}(pw, sa) := \mathsf{H}(pw|sa)$ *for random oracle* H*. Then for any* m*-sampler* $\mathcal{P}$*, any salt generator* Gen*, and any* $\ell, P, T, c \in \mathbb{N}$*,*

$$\mathbf{Adv}^{\text{bf-rec}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{H}}(P,T,c) \le \mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\mathsf{Gen}}(T+P,c) .$$

Proof. Let $(\mathcal{A}_0, \mathcal{A}_1)$ be a (P,T,c)-adversary in the BF-REC game. We construct a $(T+P,c)$-adversary $\mathcal{B}$ in the SA-GUESS game as follows. Algorithm $\mathcal{B}(\mathbf{sa}, z, z_{\mathsf{coll}})$ receives a salt vector $\mathbf{sa}$, z and a collision pattern z_{coll}. It then runs $\mathcal{A}_0()$ to obtain (σ, L).

Algorithm $\mathcal{B}$ now needs to prepare the challenge key vector $\mathbf{k}$ for $\mathcal{A}_1$. To this end, it will use its access to a TEST oracle to find out whether or not a

password-salt pair appears on L. If it does, it uses the provided value in L. Else it will pick the answer randomly, ensuring consistency using the collision pattern of password-salt pairs z_{coll}.

In more detail, for each $(pw|sa, y) \in L$ with $sa = \mathbf{sa}[i,j]$ for some (i,j), algorithm $\mathcal{B}$ queries $\textsc{Test}(pw, sa)$ and obtains a set S of indices. If S is non-empty, it contains indices i for which $(\mathbf{pw}[i], sa[i,j]) = (pw, sa)$ for some j. For these indices, algorithm $\mathcal{B}$ uses y as the challenge value. If S is empty, $\mathcal{B}$ does nothing (the (pw, sa) pair on L is not one of the challenge password-salt pairs). For indices (i,j) such that $(\mathbf{pw}[i], sa[i,j])$ does not appear on L, algorithm $\mathcal{B}$ generates uniform values compatible with z_{coll} as the corresponding challenge keys. Note that for these lazily sampled values the domain point is only partially known. Note also that at this point $\mathcal{B}$ places at most P queries to $\textsc{Test}$.

Let $\mathbf{k}$ be the set of challenge keys sampled as above. Algorithm $\mathcal{B}$ runs $\mathcal{A}_1^{\mathsf{H}[L],\textsc{Cor}}(\mathbf{sa}, \mathbf{k}, \sigma, z)$ as follows. It relays all its $\textsc{Cor}(i)$ queries to its own $\textsc{Cor}(i)$ oracle. For the $\mathbf{pw}[i]$ received, $\mathcal{B}$ updates the corresponding unknown entry part of the domain point with $\mathbf{pw}[i]$. Note that $\mathcal{B}$ places at most c queries to $\textsc{Cor}$.

To answer a random-oracle query $\mathsf{H}[L](pw, sa)$ outside L, if sa does not match $\mathbf{sa}[i,j]$ for any (i,j) it chooses a random value. If $sa = \mathbf{sa}[i,j]$ for some (i,j), algorithm $\mathcal{B}$ queries $\textsc{Test}(pw, sa)$ to get a set of indices S. If this set is empty, $\mathcal{B}$ returns a random value. If S is non-empty, then $\mathbf{pw}[i]$ for $i \in S$ are discovered and the random value generated at the challenge phase is used. Note that for $i \in S$ this value was set consistently using z_{coll}. (Algorithm $\mathcal{B}$ also updates the corresponding unknown half of the domain point.) Note also that $\mathcal{B}$ places at most T queries to $\textsc{Test}$ during this phase. □

The estimated extra P queries to $\textsc{Test}$ queries during challenge preparation may indeed arise for example when passwords are predictable and there are no salts. On the other hand, for large random salts, with overwhelming probability no queries to $\textsc{Test}$ will be made at this stage. Our next theorem formalizes this.

Theorem 7. *Let* $\mathsf{KD}^{\mathsf{H}}(pw, sa) := \mathsf{H}(pw|sa)$ *for random oracle* H*. Then for any* m*-sampler* $\mathcal{P}$*, any salt generator* $\mathsf{Gen} := [K]$ *that outputs uniform salts in a set of size* K*, and any* $\ell, P, T, c \in \mathbb{N}$*,*

$$\mathbf{Adv}^{\text{bf-rec}}_{\mathcal{P},\ell,[K],\mathsf{H}}(P,T,c) \le \left(\binom{T}{m-c} + \frac{m\ell}{K}\binom{T+P}{m-c}\right) \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m-c, c) + \frac{m^2\ell^2}{K} \ .$$

Proof. Let $(\mathcal{A}_0, \mathcal{A}_1)$ be a (P,T,c)-adversary in the BF-$\textsc{Rec}$ game. We construct an adversary $\mathcal{B}$ in the $\textsc{Guess}$ game. Algorithm $\mathcal{B}(z)$ receives z and runs $\mathcal{A}_0()$ to obtain (σ, L). It then generates a salt vector $\mathbf{sa}$ of size $m \times \ell$. If there is a collision among these salts, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ sets z_{coll} to be the all-zero collision pattern and prepares the challenge key vector $\mathbf{k}$ as follows. The difficulty in preparing these values lies in that the values need to be consistent with those specified in L. Let S denote an ordered list of salts and let P_{sa} for $sa \in [K]$ denote the number of passwords which together with sa appear in L. Since L is of size P we have that

$$\sum_{sa \in [k]} P_{sa} = P \ .$$

To prepare the challenge vector consistently, $\mathcal{B}$ calls its $\textsc{Test}(i, pw)$ oracle on each password pw appearing on the L together with some salt $\mathbf{sa}[i, j] \in S$. If a password-salt pair is discovered to be on L, algorithm $\mathcal{B}$ uses the value provided in L, else it picks a random value. At this phase algorithm $\mathcal{B}$ makes $\sum_{sa \in S} P_{sa}$ queries to $\textsc{Test}$.

Algorithm $\mathcal{B}$ now runs $\mathcal{A}_1(\mathbf{sa}, \mathbf{k}, \sigma, z)$ and answers its corruption queries by queries its own corruption oracle. Primitive queries on (pw, sa), which wlog can be assumed to be outside L, are handled by first querying $\textsc{Test}(i, pw)$ if $sa = \mathbf{sa}[i, j]$ for some (i, j) and accordingly using either a value from the challenge phase, or a uniform value. Thus,

$$\mathbf{Adv}^{\text{bf-rec}}_{\mathcal{P},\ell,[K],\mathsf{H}}(P, T, c) \leq \frac{m^2\ell^2}{K} + \sum_{S \in [K]^{(m\ell)}} \frac{1}{K^{m\ell}} \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T + \sum_{sa \in S} P_{sa}, c) \ ,$$

where $[K]^{(m\ell)}$ denotes all ordered lists of size $m\ell$ with *distinct* entries in K.

By Theorem 2 each summand above can be bound as

$$\mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(T + \sum_{sa \in S} P_{sa}, c) \leq \binom{T + \sum_{sa \in S} P_{sa}}{m - c} \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m - c, c) \ .$$

Now for fixed m and c the right-hand side is a convex function. Thus, by Jensen's inequality, the sum attains its maximum at one of the extremal values where $P_{sa} = P$ for a single salt $sa = sa^*$, and $P_{sa} = 0$ elsewhere. Since the advantage terms are symmetric, without loss of generality, we may assume that $sa^* = 1$.

For this particular distribution of passwords in L, we have that for $(m\ell)!\binom{K-1}{m\ell}$ terms the number of additional queries, $\sum_{sa \in S} P_{sa}$, is zero: choose $m\ell$ salts in $[K] \setminus \{sa^*\}$ and order them. For these cases $\mathcal{B}$ places T queries in total. For $(m\ell)!\binom{K-1}{m\ell-1}$ terms the number of additional queries is $\sum_{sa \in S} P_{sa} = P$: choose one salt to be sa^*, the rest in $[K] \setminus \{sa^*\}$, and order. For these cases $\mathcal{B}$ places $T + P$ queries in total.

Hence we obtain that

$$\begin{aligned}\mathbf{Adv}^{\text{bf-rec}}_{\mathcal{P},\ell,[K],\mathsf{H}}(P, T, c) \leq \frac{m^2\ell^2}{K} + \frac{(m\ell)!}{K^{m\ell}} \cdot \binom{K-1}{m\ell} \cdot \binom{T}{m-c} \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m-c, c) \\ + \frac{(m\ell)!}{K^{m\ell}} \cdot \binom{K-1}{m\ell-1} \cdot \binom{T+P}{m-c} \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m-c, c) \ .\end{aligned}$$

Using the upper bound on the binomial coefficients we have

$$\frac{(m\ell)!}{K^{m\ell}} \cdot \binom{K-1}{m\ell} \leq 1 \qquad \text{and} \qquad \frac{(m\ell)!}{K^{m\ell}} \cdot \binom{K-1}{m\ell-1} \leq \frac{m\ell}{K} \ .$$

The theorem follows. □

4.1 Main theorems

In this section we derive upper bounds on the adversarial advantage in the AI-$\textsc{Rec}$ game based on the bounds established in the previous section. The-

orems 8–10 below upper-bound unrecoverability of hashed passwords in three different settings. We will use Theorems 2–4 to prove these results.

We start with the case of unsalted passwords. We focus on the case with no corruption; the case with corruptions can be dealt with similarly using our results but the involved bounds are more complex.

Theorem 8 (No salts). *Let $\mathcal{P}$ be an m-sampler and consider the empty salt generator. Then for any adversary in the* AI-Rec *game outputting at most S bits of side information, making at most T queries to the random oracle and no corruption queries, for any $\gamma > 0$ and $m \leq T$ we have that*

$$\mathbf{Adv}^{\text{ai-rec}}_{\mathcal{P},\ell,\perp,\mathsf{H}}(S,T,0) \leq 2^m \cdot \left(T + \frac{2T(S+\log\gamma^{-1})}{m}\right)^m \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m,0) + \gamma \ .$$

Proof. Theorems 4 and 6 together yield

$$\mathbf{Adv}^{\text{bf-rec}}_{\mathcal{P},\ell,\perp,\mathsf{H}}(P,T,0) \leq (T+P)^m \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m,0) \ .$$

Using the second (i.e., the unpredictability) part of Theorem 1, noting that in our setting there are $m+T$ calls to H, and assuming that $m \leq T$ for any $\gamma > 0$, we may set

$$P := \frac{(S+\log\gamma^{-1})(m+T)}{m} \leq \frac{2T(S+\log\gamma^{-1})}{m}$$

to deduce the stated bound for any $\gamma > 0$.[10] □

We next consider the case of distinct and potentially low-entropy salts. This is for example the case when salts are an index and consequently the domain of the hash function is separated for different users.

Theorem 9 (Known distinct salts). *Let $\mathcal{P}$ be an m-sampler and consider a salt generator that always outputs distinct, but potentially low-entropy, known salts. Then for any adversary in the* AI-Rec *game outputting at most S bits of side information, making at most T queries to the random oracle and no corruption queries, for any $\gamma > 0$ and $m \leq T$ we have that*

$$\mathbf{Adv}^{\text{ai-rec}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{H}}(S,T,0) \leq 2^m \cdot \left(\frac{\mathrm{e}T + 2\mathrm{e}T(S+\log\gamma^{-1})/m}{m}\right)^m \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m,0) + \gamma \ .$$

Proof. In the case of salted passwords with distinct salts, Theorems 3 and 6 together yield

$$\mathbf{Adv}^{\text{bf-rec}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{H}}(P,T,0) \leq \binom{T+P}{m} \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m,0) \leq \left(\frac{\mathrm{e}(T+P)}{m}\right)^m \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m,0) \ .$$

Using the second part of Theorem 1, we may set P as in the unsalted case (which is close to the optimal) to deduce that the stated bound for any $\gamma > 0$. □

[10] Via differentiation, this value of P is close to the optimal choice.

We finally consider the case of uniform salts.

Theorem 10 (Uniform salts). *Let $\mathcal{P}$ be an m-sampler and consider a salt generator that always outputs uniformly random salts in a set of size K. Then for any adversary in the* AI-Rec *game outputting at most S bits of side information, making at most T queries to the random oracle and no corruption queries, for any $\gamma > 0$ and $m \leq T$ we have that*

$$\mathbf{Adv}^{\text{ai-rec}}_{\mathcal{P},\ell,[K],\mathsf{H}}(S,T,0) \leq 2^m \cdot \left(\left(\frac{\mathrm{e}T}{m} \right)^m + \frac{m\ell}{K} \cdot \left(\frac{\mathrm{e}T + 2\mathrm{e}T(S + \log \gamma^{-1})/m}{m} \right)^m \right) \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m,0) + \frac{m^2\ell^2}{K} + \gamma \ .$$

Proof. Using Theorems 3 and 7 for uniform salts in $[K]$ we get

$$\mathbf{Adv}^{\text{bf-rec}}_{\mathcal{P},\ell,[K],\mathsf{H}}(P,T,0) \leq \left(\left(\frac{\mathrm{e}T}{m} \right)^m + \frac{m\ell}{K} \left(\frac{\mathrm{e}(T+P)}{m} \right)^m \right) \cdot \mathbf{Adv}^{\text{guess}}_{\mathcal{P}}(m,0) + \frac{m^2\ell^2}{K} \ .$$

We may set P as in the previous cases, which is again close to optimal, to deduce the stated bound for any $\gamma > 0$. □

We summarize the above discussion for the case of uniform passwords in $[N]$ in the table below. We have assumed $\log \gamma^{-1} \leq m$ and have removed the additive "$+\gamma$" terms, and in the uniform case "$+\frac{m^2\ell^2}{K}$" terms, to help readability.

	No salts	Known distinct salts	Uniform salts
$S = 0$	$\left(\frac{6T}{N}\right)^m$	$\left(\frac{6\mathrm{e}T}{mN}\right)^m$	$\left(1 + \frac{m\ell}{K}\right) \cdot \left(\frac{6\mathrm{e}T}{mN}\right)^m$
"Large" $S \geq 3m$	$\left(\frac{6ST}{mN}\right)^m$	$\left(\frac{6\mathrm{e}ST}{m^2N}\right)^m$	$\left(\frac{2\mathrm{e}T}{mN}\right)^m + \frac{m\ell}{K} \cdot \left(\frac{6\mathrm{e}ST}{m^2N}\right)^m$

5 Iterated Hashing

A well-known method for reducing vulnerabilities to brute-force attacks is to compute *iterated* hashes of salted passwords. The effects of iteration will be hardly noticeable by the honest users, but for the adversary the cryptanalytic effort will increase by a factor proportional to the number of iteration rounds (converting weeks of effort to years). This mechanism has been used, for example, in classical password-hashing mechanisms such as PBKDF and bcrypt.

The r-iterated construction is

$$\mathsf{KD}^{\mathsf{H}}_r(pw, sa) := \underbrace{\mathsf{H} \circ \cdots \circ \mathsf{H} \circ \mathsf{H}}_{r}(pw|sa) \ ,$$

where $r \in \mathbb{N}$ is the number of rounds, and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ is a hash function that we will model as a random oracle. We also assume that $pw|sa$ is never of length n (and hence such values cannot be a hash output).

Game AI-KDF-REAL$^{\mathcal{D}_0,\mathcal{D}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}$:	Game BF-KDF-REAL$^{\mathcal{D}_0,\mathcal{D}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}$:
$\mathsf{H} \twoheadleftarrow \mathrm{Fun}(D,R)$	$\mathsf{H} \twoheadleftarrow \mathrm{Fun}(D,R)$
$\sigma \twoheadleftarrow \mathcal{D}_0(\mathsf{H})$	$(\sigma, L) \twoheadleftarrow \mathcal{D}_0()$
$(\mathbf{pw}, z) \twoheadleftarrow \mathcal{P}$	$(\mathbf{pw}, z) \twoheadleftarrow \mathcal{P}$
for $(i,j) \in [m] \times [\ell]$ do	for $(i,j) \in [m] \times [\ell]$ do
$\quad \mathbf{sa}[i,j] \twoheadleftarrow \mathsf{Gen}(i,j)$	$\quad \mathbf{sa}[i,j] \twoheadleftarrow \mathsf{Gen}(i,j)$
$\quad \mathbf{k}[i,j] \leftarrow \mathsf{KD}^{\mathsf{H}}(\mathbf{pw}[i], \mathbf{sa}[i,j])$	$\quad \mathbf{k}[i,j] \leftarrow \mathsf{KD}^{\mathsf{H}}(\mathbf{pw}[i], \mathbf{sa}[i,j])$
$b' \twoheadleftarrow \mathcal{D}_1^{\mathrm{PRIM}}(\mathbf{pw}, \mathbf{sa}, \mathbf{k}, z, \sigma)$	$b' \twoheadleftarrow \mathcal{D}_1^{\mathrm{PRIM}}(\mathbf{pw}, \mathbf{sa}, \mathbf{k}, z, \sigma)$
return b'	return b'
Proc. PRIM(w):	Proc. PRIM(w):
return $\mathsf{H}(w)$	return $\mathsf{H}[L](w)$

Game BF/AI-KDF-IDEAL$^{\mathcal{D}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathcal{S}_0,\mathcal{S}_1}$:	Proc. PRIM(w):
$(\sigma, st) \twoheadleftarrow \mathcal{S}_0()$	$(y, st) \twoheadleftarrow \mathcal{S}_1^{\mathrm{TEST}}(w; st)$
$(\mathbf{pw}, z) \twoheadleftarrow \mathcal{P}$	return y
for $(i,j) \in [m] \times [\ell]$ do	
$\quad \mathbf{sa}[i,j] \twoheadleftarrow \mathsf{Gen}(i,j)$	Proc. TEST(pw, sa):
$\quad \mathbf{k}[i,j] \twoheadleftarrow \{0,1\}^k$	$S \leftarrow \{i \in [m] : \exists j \in [\ell]$ st.
$b' \twoheadleftarrow \mathcal{D}_1^{\mathrm{PRIM}}(\mathbf{pw}, \mathbf{sa}, \mathbf{k}, z, \sigma)$	$\quad (\mathbf{pw}[i], \mathbf{sa}[i,j]) = (pw, sa)\}$
return b'	return $\mathbf{k}[S]$

Fig. 4. Simulation-based notion of KDF security in the AI-RO and BF-RO model. Note that the ideal games are syntactically identical.

Following BRT, in this section we adopt a more modular approach to security and formulate two simulation-based notions of security for KDF. In the next section we will then show how to use KDF security to argue for the security of password-based protocols.

AI-KDF SECURITY. Our first definition is a (simulation-based) notion of KDF security which extends that of [BRT12, Section 3.1] to the AI-RO model. We define the KDF advantage of an adversary $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$ in the AI-RO model with respect to a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ as

$$\mathbf{Adv}^{\text{ai-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD},\mathcal{S}_0,\mathcal{S}_1}(\mathcal{D}_0,\mathcal{D}_1) := \Pr\left[\text{AI-KDF-REAL}^{\mathcal{D}_0,\mathcal{D}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}\right] - \Pr\left[\text{AI-KDF-IDEAL}^{\mathcal{D}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathcal{S}_0,\mathcal{S}_1}\right] ,$$

where games AI-KDF-REAL and AI-KDF-IDEAL are defined in Fig. 4.

Our definition differs from that of BRT in a number of aspects. First, it includes a preprocessing stage via $\mathcal{D}_0$ in the real game and a simulated preprocessing stage via $\mathcal{S}_0$ in the ideal game. Second, our games sample salts via Gen, whereas in BRT an arbitrary joint distribution on passwords-salt pairs was considered. Such a notion is infeasible to achieve in the presence of preprocessing as salts need to have entropy. Finally, the TEST procedure in the ideal game returns the set of all indices i for which the i-th password matches the queried

password and some salt associated with it matches the queried salt. We note that, as in BRT, the simulator does *not* get access to the collision pattern of the password-salt pairs: the purpose of the ideal KDF game is to translate security to a setting where keys are truly random, and their collision patterns are not necessarily known.

BF-KDF SECURITY. We now define an analogous notion of KDF security in the BF-RO model. We set

$$\mathbf{Adv}^{\text{bf-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD},\mathcal{S}_0,\mathcal{S}_1}(\mathcal{D}_0,\mathcal{D}_1) := \Pr\left[\text{BF-KDF-REAL}^{\mathcal{D}_0,\mathcal{D}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}}\right] - \Pr\left[\text{BF-KDF-IDEAL}^{\mathcal{D}_1}_{\mathcal{P},\ell,\mathsf{Gen},\mathcal{S}_0,\mathcal{S}_1}\right] ,$$

where games BF-KDF-REAL and BF-KDF-IDEAL are defined in Fig. 4. Note that the ideal AI and BF games are syntactically identical.

Using Theorem 1, we first show that KDF security in the BF-RO model implies KDF security in the AI-RO model.

Theorem 11 (BF-to-AI KDF Security). *Let* KD^{H} *be a key-derivation where* H *is a random oracle and let* Gen *be a salt generator. Then for any AI-KDF distinguisher* $(\mathcal{D}_0,\mathcal{D}_1)$*, where* $\mathcal{D}_0$ *outputs* S *bits of auxiliary information and* $\mathcal{D}_1$ *places at most* T *queries to its* PRIM *oracle, and any* $P \in \mathbb{N}$ *and* $\gamma > 0$ *there is a BF-KDF distinguisher* $(\tilde{\mathcal{D}}_0,\tilde{\mathcal{D}}_1)$*, where* $\tilde{\mathcal{D}}_0$ *output a string of length at most* S *and a list of size at most* P*, and such that for any BF-KDF simulator* $(\mathcal{S}_0,\mathcal{S}_1)$ *and any* $\ell \in \mathbb{N}$*,*

$$\mathbf{Adv}^{\text{ai-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD},\mathcal{S}_0,\mathcal{S}_1}(\mathcal{D}_0,\mathcal{D}_1) \le \mathbf{Adv}^{\text{bf-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD},\mathcal{S}_0,\mathcal{S}_1}(\tilde{\mathcal{D}}_0,\tilde{\mathcal{D}}_1) + \frac{(S+\log\gamma^{-1})\cdot(rm\ell+T)}{P} + \gamma \ .$$

Proof. Let $(\mathcal{D}_0,\mathcal{D}_1)$ be an AI-indifferentiability adversary. We apply the first part of Theorem 1 to the real AI-KDF game and obtain a BF-KDF distinguisher $(\tilde{\mathcal{D}}_0,\tilde{\mathcal{D}}_1)$. In the real game there are in total at most $rm\ell+T$ queries to H. Now let $(\mathcal{S}_0,\mathcal{S}_1)$ be a BF-KDF simulator for $(\tilde{\mathcal{D}}_0,\tilde{\mathcal{D}}_1)$. Then $(\mathcal{S}_0,\mathcal{S}_1)$ is also an AI-KDF simulator as the ideal BF- and AI-KDF games are syntactically identical. □

We now prove that salted iteration of a random oracle achieves KDF security in the BF-RO model. The technical challenge here is to show that the results of [BRT12] can be "lifted" to a setting with auxiliary information. To do so, we can follow the generic approach of Coretti et al. [CDGS18] (i.e., Theorem 1), but this results in a different construction where *every* primitive call in the construction is salted. Standard iterated constructions, however, only salt the innermost call.

We thus directly establish the *bit-fixing* KDF security of the iterated construction when $pw|sa$ are never an n-bit string where n is the output length of H. We then translate this result to the auxiliary-input setting using the above

theorem. The length restriction on $pw|sa$ allows us to decouple the innermost call to H from the rest of the calls.[11]

Theorem 12 (Bit-fixing KDF security). *Let $\mathsf{KD}^{\mathsf{H}}_r$ be the r-iterated key-derivation function where $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ is a random oracle, and let Gen be a salt generator that outputs uniform salts in a set of size K. Let $N := 2^n$. Then for any BF-KDF distinguisher $(\mathcal{D}_0, \mathcal{D}_1)$ where $\mathcal{D}_1$ outputs a list of size P and makes at most T primitive queries, there is a simulator $(\mathcal{S}_0, \mathcal{S}_1)$ such that for any $\ell \in \mathbb{N}$*

$$\mathbf{Adv}^{\text{bf-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}_r,\mathcal{S}_0,\mathcal{S}_1}(\mathcal{D}_0,\mathcal{D}_1) \le \frac{(r+1)m\ell T}{N} + 3\cdot\left(\frac{rm\ell(rm\ell+P)}{N} + \frac{m\ell P}{K} + \frac{m^2\ell^2}{K}\right).$$

Furthermore, $\mathcal{S}_1$ places at most T/r queries to RO and runs in time $\tilde{\mathcal{O}}(r)$.

We start with a high-level overview of our simulator, which we define in Fig. 5. The initial stage of the simulator $\mathcal{S}_0$ simply runs $\mathcal{D}_0$ to get (σ, L), and populates table H with assignments in L. It passes H (which is essentially L) onto $\mathcal{S}_1$ via state st. The online simulator simulates H via lazy sampling using H and detecting completed chains (as is common in indifferentiability proofs). For a query w, it looks for a *chain* of queries $w_0, w_1, \ldots, w_{r-1}$ such that the chain starts at $w_0 = pw|sa$ for a one of salts (note that the simulator knows the salts) and ends at $w_{r-1} = w$, and furthermore along the chain there were no collisions in H (and hence the chain, if defined, is unique). If a chain is found, the simulator uses the random oracle RO to answer the query; else a fresh random string is chosen. This simulator makes a TEST query as a result of a chain completion. Hence it makes at most T_2/r queries.[12]

Proof (Sketch). We now give an overview of the game transitions used in the proof of BF-KDF security and refer the reader to the full version of the paper for the details.

G_0: In this game we initially populate a table H (used for lazy sampling) with entries in L. We compute the challenge vector using lazy sampling and also answer primitive queries using H.

G_1: In this game we "optimistically" sample the outputs of the random oracle for the challenge values. We also set a bad flag bad if while computing chains we encounter a penultimate value whose hash has already been set. We still use the value already set, but in the next game we would like to set this value to the optimistically chosen one. G_1 and G_0 are identical.

[11] In particular, we do not run into "hash-of-hash" problems as in [DRST12] as *not* every H call is salted.

[12] We emphasize that without the length restriction on password-salt pairs, this simulator can fail. Consider a differentiator that gets $w_r \leftarrow \textsc{Const}(pw, sa)$, $w'_{r+1} \leftarrow \textsc{Prim}(w_r)$, and $w'_1 \leftarrow \textsc{Prim}(pw|sa)$. It the parses w'_1 as (pw', sa'), gets $w_{r+1} \leftarrow \textsc{Const}(pw', sa')$, and checks if $(w_{r+1} = w'_{r+1})$. This attack corresponds to the computation of two overlapping chains. Our $\mathcal{S}_1$ fails as it simulates the two PRIM queries randomly since it won't be able to detect any chains.

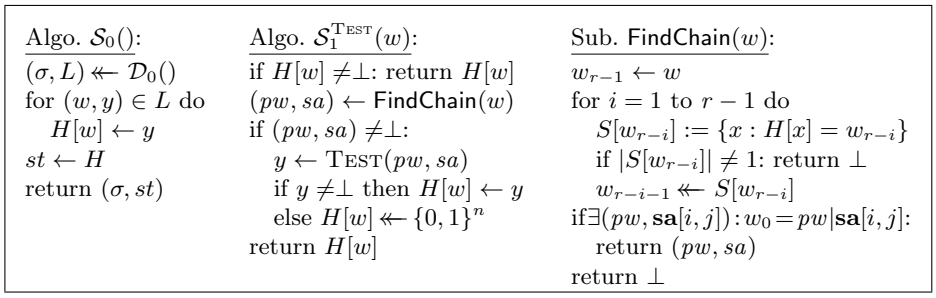

Fig. 5. Simulator for the bit-fixing KDF security of the r-iterated random oracle.

G_2**:** In this game, even if the hash of a penultimate value is already defined and the flag bad gets set, we set the hash of the penultimate value to the optimistically chosen one. (Note that the primitive oracle has not been changed.) G_2 and G_1 are identical until bad.

G_3**:** In this game we introduce two conceptual changes: (1) We no longer set the entry in H for the penultimate values; and (2) We modify the primitive oracle to check if a query is a penultimate value. If so, the primitive oracle uses the optimistically sampled value. G_2 and G_3 are identical.

G_4**:** In this game we change the way the primitive oracle operates by first checking if there is a chain of values of length $r-1$ leading to the query w. (This is done by maintaining a set of edges E for the graph resulting from the queries. If so, we set the hash of w to the optimistically chosen one. Otherwise if the query w matches a penultimate value, we set a flag bad_2 and set hash value to the optimistically chosen one. G_3 and G_4 are identical until bad.

G_5**:** In this game if bad_2 is set, we do not use the optimistically set value, but rather a random value. The two games are identical until bad_2.

G_6**:** This game removes code in computing the challenge keys and setting of bad_2. It also moves populating H with L to the primitive oracle. This game is identical to G_5.

G_7**:** In this game we modify the way chains are detected. Now of course the simulator does not know the password-salt pairs. Hence, we modify this chain detection procedure so that it uses a TEST oracle to check if a password-salt pair that traces to the queried point is indeed one of the challenge password-salt pairs. This game is identical to the ideal AI-KDF game with the simulator in Fig. 5.

Such a path, if it exists, will be unique as long the paths are isolated (that is there are no edges (u, v) on graph of execution E such that v is on the path but u is not) and the path has no loops. Here we use the fact that password-salt pairs have length different than n-bit, so that the adversary cannot "slide" the path.

These conditions ensure that there is at more one path for a given w. In particular the simulator makes at most T/r such paths and thus the simulator's number of queries to TEST is also at most T/r times.

We now bound the distinguishing advantage in the transitions above. The games G_1–G_4 are all identical until bad. So we can bound this difference with a *single* game hop. (G_2 and G_3 were used for reasoning.) The probability of bad is *upper bounded* by the probabilities of (1) hitting a bad starting point with a non-fresh salt, that is $m\ell P/K$; (2) two salts colliding, $m^2\ell^2/K$ (here we do not make any assumptions about the entropy of $\mathcal{P}$ and in particular it could be that the sampled passwords are not distinct); and (3) any value generated collides with a value generated before, or one of the pre-sampled values, i.e., $(rm\ell)(rm\ell+P)/N$. Thus, the overall bound is

$$\frac{m\ell P}{K} + \frac{m^2\ell^2}{K} + \frac{(rm\ell)(rm\ell + P)}{N} \ .$$

The distance between G_4 and G_5 is bounded by the probability of setting bad_2. We have that $\Pr[\mathsf{bad}_2] \leq \Pr[\mathsf{bad}] + \Pr[\mathsf{bad}_2|\neg\mathsf{bad}]$. Now under $\neg\mathsf{bad}$ the values that provoke bad_2 are uniform. Since there are $rm\ell$ of them we get that $\Pr[\mathsf{bad}_s|\neg\mathsf{bad}] \leq rm\ell T/N$.

We now bound the distance between G_6 and G_7. The probability that input-outputs defining the paths from password-salt pairs of length $r-1$ stay disjoint and outside L is given by the bound displayed above. Thus, we pick up three $\Pr[\mathsf{bad}]$ terms in total. The probability that no other queries enter into these paths is at most $rm\ell T/N$. The theorem follows by collecting terms. □

We note our bound above does not involve birthday terms of the form T^2 or $T \cdot P$, which would translate to salt sizes that are too large to be acceptable in practice. We may now apply Theorem 11 to deduce that for any AI-KDF adversary $(\mathcal{D}_0, \mathcal{D}_1)$ that outputs at most S bits of auxiliary information and places at most T queries to the primitive oracle, there is a simulator (namely the simulator for the BF-KD notion) such that for any $\gamma > 0$

$$\mathbf{Adv}^{\text{ai-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}_r,\mathcal{S}_0,\mathcal{S}_1}(\mathcal{D}_0,\mathcal{D}_1) \quad \leq \quad \frac{S'T'}{P} \ + \ 3P\left(\frac{rm\ell}{N} + \frac{rm\ell}{K}\right) \ + \ \cdots \quad ,$$

where $S' := S + \log \gamma^{-1}$ and $T' := rm\ell + T$ and the omitted terms do not involve P. For the optimal P, we set the two terms involving P to a common value and obtain (up to constant factors) that

$$P = \sqrt{\frac{S'T'NK}{3rm\ell(N+K)}} \ .$$

Plugging this back into the bound we finally obtain that

$$\mathbf{Adv}^{\text{ai-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD}_r,\mathcal{S}_0,\mathcal{S}_1}(\mathcal{D}_0,\mathcal{D}_1) \leq 6 \cdot \sqrt{\frac{S'T'NK}{3rm\ell(N+K)}} \cdot \left(\frac{rm\ell}{N} + \frac{rm\ell}{K}\right) + \\ + \frac{3m^2\ell^2}{K} + \frac{3r^2m^2\ell^2 + (r+1)m\ell T}{N} + \gamma \ .$$

Game ai-multi-$\mathrm{G}^{\mathcal{A}_0,\mathcal{A}_1}_{\mathrm{G},\mathcal{P},\mathsf{Gen},\mathsf{KD}}$:
$\mathsf{H} \twoheadleftarrow \mathrm{Fun}(D,R)$
$\sigma \twoheadleftarrow \mathcal{A}_0(\mathsf{H})$
$(b_1,\ldots,b_m) \twoheadleftarrow \{0,1\}^m$
$(\mathbf{pw},z) \twoheadleftarrow \mathcal{P}$
for $(i,j) \in [m] \times [\ell]$ do
  $\mathbf{sa}[i,j] \twoheadleftarrow \mathsf{Gen}(i,j)$
  $\mathbf{k}[i,j] \leftarrow \mathsf{KD}^{\mathsf{H}}(\mathbf{pw}[i],\mathbf{sa}[i,j])$
for $i \in [m]$ do
  $(x_i, st_i) \leftarrow \mathrm{G}(\varepsilon;(\mathbf{k}[i,1],\ldots,\mathbf{k}[i,\ell],b_i,r_i);\varepsilon)$
$b' \twoheadleftarrow \mathcal{A}_1^{\textsc{Game},\textsc{Cor},\mathsf{H}}(\mathbf{sa},x_1,\ldots,x_m,\sigma,z)$
return $(b' = \oplus_{i=1}^m b_i)$

Proc. $\textsc{Game}(i,x)$:
$(y,st_i) \leftarrow \mathrm{G}(x;(\mathbf{k}[i,1],\ldots,\mathbf{k}[i,1],b_i,r_i);st_i)$
return y

Proc. $\textsc{Cor}(i)$:
return $\mathbf{pw}[i]$

Proc. $\mathsf{H}(w)$:
return $\mathsf{H}(w)$

Fig. 6. Security game for the password-based multi-instance extension of G in the presence of auxiliary information on H. States st_i are initialized to ε and r_i are independent random coins of appropriate length. ℓ is the number of random strings that need to be replaced in each instance of G.

6 KDF Security in Applications

Given a game G, as defined in Section 2, we consider a multi-instance extension that runs a *central* adversary $\mathcal{A}_1$ with respect to m independent instances of G. Let

$$\mathbf{Adv}^{\mathrm{single}}_{\mathrm{G}}(\mathcal{A}) := 2 \cdot \Pr\left[\mathrm{G}^{\mathcal{A}}\right] - 1 \ ,$$

be the single-instance advantage. Suppose G uses randomness $(k_1,\ldots,k_\ell,b,r)$ for some ℓ. We are interested in replacing the values k_j with those that are derived from passwords through a KDF (see game ai-multi-G in Fig. 6). Let

$$\mathbf{Adv}^{\text{ai-multi}}_{\mathrm{G},\mathcal{P},\mathsf{Gen},\mathsf{KD}}(\mathcal{A}_0,\mathcal{A}_1) := 2 \cdot \Pr\left[\text{ai-multi-}\mathrm{G}^{\mathcal{A}_0,\mathcal{A}_1}_{\mathrm{G},\mathcal{P},\mathsf{Gen},\mathsf{KD}}\right] - 1 \ .$$

We show that if KD^{H} is a secure KDF in the AI-RO model, this advantage can be upper bounded by those in the single-instance game G and the salted guessing game.

Theorem 13. *Let* G *be a game with* ℓ *keys,* $\mathcal{P}$ *an* m*-sampler,* KD^{H} *a key-derivation function in the RO model, and* Gen *a salt generator. Then for any adversary* $(\mathcal{A}_0,\mathcal{A}_1)$ *in* ai-multi-G *with* $\mathcal{A}_0$ *outputting at most* S *bits of auxiliary information, and* $\mathcal{A}_1$ *placing at most* T *queries to* H *and at most* c *queries to* $\textsc{Cor}$*, there is a AI-KDF distinguisher* $(\mathcal{D}_0,\mathcal{D}_1)$ *where* $\mathcal{D}_0$ *also outputs at most* S *bits of auxiliary information and* $\mathcal{D}_1$ *places at most* T *queries to its* $\textsc{Prim}$ *oracle, and an adversary* $\mathcal{B}$ *against* G *in the single instance setting (with uniform randomness) that uses* S *bits of non-uniformity and runs in time that of* $\mathcal{A}_1$ *plus the time need to run* $m-1$ *instances of* G *such that for any AI-KDF simulator*

$(\mathcal{S}_0, \mathcal{S}_1)$,

$$\mathbf{Adv}^{\text{ai-multi}}_{\mathrm{G},\mathcal{P},\mathsf{Gen},\mathsf{KD}}(\mathcal{A}_0,\mathcal{A}_1) \le 2 \cdot \mathbf{Adv}^{\text{ai-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD},\mathcal{S}_0,\mathcal{S}_1}(\mathcal{D}_0,\mathcal{D}_1) + \\ + 2 \cdot \mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\mathsf{Gen}}(T,c) + m \cdot \mathbf{Adv}^{\text{single}}_{\mathrm{G}}(\mathcal{B}) \ .$$

Proof. The proof follows that of [BRT12, Theorem 3.4], except that we need to deal with general games and also auxiliary information on H.

Let ai-multi-G_0 be identical to ai-multi-G. Let $(\mathcal{S}_0, \mathcal{S}_1)$ be the AI-KDF simulator. We modify ai-multi-G_0 to a game ai-multi-G_1 that uses random keys instead of keys derived from passwords, and where σ and H are stimulated via the AI-KDF simulator $(\mathcal{S}_0, \mathcal{S}_1)$. This transition is justified using AI-KDF security as the inputs and oracles provided in each of the two AI-KDF games (that is, all passwords, real/random keys, and salts) are sufficient for an AI-KDF distinguisher $\mathcal{D}_0, \mathcal{D}_1$ to simulate ai-multi-G_0 or ai-multi-G_1. In this transition the distinguisher also picks coins r_i and bits b_i. When $\mathcal{A}_1$ returns b', the distinguisher returns $(b' = \oplus_{i=1}^{m} b_i)$. Thus,

$$\Pr[\text{ai-multi-}\mathrm{G}_0] - \Pr[\text{ai-multi-}\mathrm{G}_1] \le \mathbf{Adv}^{\text{ai-kdf}}_{\mathcal{P},\ell,\mathsf{Gen},\mathsf{KD},\mathcal{S}_0,\mathcal{S}_1}(\mathcal{D}_0,\mathcal{D}_1) \ .$$

We now modify ai-multi-G_1 to ai-multi-G_2 that sets flag bad and terminates if the simulator queries *all* passwords to its Test oracle. The two games are identical until bad. We can upper-bound the probability of bad by building a sa-Guess adversary as follows. Run the initial simulator $\mathcal{S}_0$ to generate σ. (Note that this step entails that the sa-Guess adversary that we build is potentially unbounded.) Pick randomness (including keys) to simulate the m instances of the games. The Cor oracle is simulated by relaying queries to and the corruption oracle provided in sa-Guess. For the Test query, use the Test oracle in sa-Guess to get a set S of indices. If this set is non-empty, return the corresponding set of keys $\mathbf{k}[S]$; otherwise return $\perp$. Whenever bad is set, sa-Guess is won and thus

$$\Pr[\text{ai-multi-}\mathrm{G}_1] - \Pr[\text{ai-multi-}\mathrm{G}_2] \le \mathbf{Adv}^{\text{sa-guess}}_{\mathcal{P},\ell,\mathsf{Gen}}(T,c) \ .$$

We now bound the probability of winning ai-multi-G_2 in terms of winning a single instance of G with random keys. This is done via a simple guessing argument. In game ai-multi-G_2 at the onset an adversary $\mathcal{B}$ guesses an index i^* among the m instances which won't be corrupted. By the bad flag introduced in the previous game, if an index remains uncorrupted, i^* will be a good guess with probability $1/m$. For the reduction, $\mathcal{B}$ chooses σ, passwords, salts, and randomness for all games except for the i^*-th game. All games except the i^*-th instance are simulated using these values. The i^*-th instance is simulated using the provided game G. If i^* is corrupted, $\mathcal{B}$ returns a random bit. When $\mathcal{A}_1$ terminates with b', algorithm $\mathcal{B}$ returns $b' \oplus_{i \neq i^*} b_i$ as its guess in the single instance game. This guess is correct guess whenever $b' = b_{i^*}$. Thus,

$$\Pr[\mathrm{G}] = 1/m \cdot \Pr[\text{ai-multi-}\mathrm{G}_2] + (1 - 1/m) \cdot 1/2 \ ,$$

and hence

$$\Pr[\text{ai-multi-G}_2] - 1/2 = m \cdot (\Pr[\text{G}] - 1/2) = m/2 \cdot \mathbf{Adv}^{\text{single}}_{\text{G}}(\mathcal{B}) \ .$$

The theorem follows by collecting the terms above and using the definitions of the advantage functions. Note that $\mathcal{B}$ uses $|\sigma|$ bits of non-uniformity in this reduction. □

Our result generalizes [BRT12, Theorem 3.4] to a larger class of games, which among others includes IND-CPA security for symmetric encryption (as considered by BRT without auxiliary information), as well as other games such as AE or CCA security for symmetric encryption, unforgeability for MACs, and may others all in the presence of auxiliary information. We emphasize that this result does not extend to games that access H (i.e., the random oracle in G and that used by the KD are shared). Indeed, when attempting to prove such a result, the initial two sequence of games above go through, but the last step fails: the oracle access in instance i^* cannot be simulated. (And indeed, attacks do exist.) Despite this, if the domains of access for H are separated for G and KD such an extension can be established.

Acknowledgments. Tessaro was partially supported by NSF grants CNS-1930117 (CAREER), CNS-1926324, CNS-2026774, a Sloan Research Fellowship, and a JP Morgan Faculty Award.

References

AS15. Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, June 2015.

BRT12. Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro. Multi-instance security and its application to password-based cryptography. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 312–329. Springer, Heidelberg, August 2012.

CDG18. Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 693–721. Springer, Heidelberg, August 2018.

CDGS18. Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 227–258. Springer, Heidelberg, April / May 2018.

CDMP05. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.

DGK17. Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 473–495. Springer, Heidelberg, April / May 2017.

DRST12. Yevgeniy Dodis, Thomas Ristenpart, John P. Steinberger, and Stefano Tessaro. To hash or not to hash again? (In)differentiability results for H^2 and HMAC. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 348–366. Springer, Heidelberg, August 2012.

DTT10. Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 649–665. Springer, Heidelberg, August 2010.

GT00. Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st FOCS*, pages 305–313. IEEE Computer Society Press, November 2000.

Kal00. Burt Kaliski. Pkcs# 5: Password-based cryptography specification version 2.0. 2000.

MRH04. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

Oec03. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 617–630. Springer, Heidelberg, August 2003.

PJ16. C. Percival and S. Josefsson. The scrypt Password-Based Key Derivation Function. RFC 7914 (Informational), August 2016.

PM99. Niels Provos and David Mazières. A future-adaptable password scheme. In *Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference, June 6-11, 1999, Monterey, California, USA*, pages 81–91. USENIX, 1999.

Unr07. Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, Heidelberg, August 2007.

Wee05. Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 523–532. ACM Press, May 2005.

A Multi-Instance Hellman

We present a simple adaptation of Hellman's space-time trade-off algorithm for inverting random permutations. Consider the cycle graph of the permutation $\pi : [N] \to [N]$. We pick S points that are roughly equidistant on the graph. We store each such point together with a pointer to a point T/m steps behind. Now given $\pi(x) = y$, where x is within distance T/m from one of the S points on the cycle graph, we can successfully recover x in T steps: We apply π iteratively to y until we reach one of the S points. We then jump backwards by T/m steps (following the stored pointer), and apply π until we reach x. This process takes exactly T/m evaluations of π. (This is as in the single-instance case except for the T/m instead of T.)

Now, in the multi-instance setting with m points $x_1, \ldots, x_m$, if all of the x_i's land within distance T/m from one of the S points, we can recover each x_i by evaluating π for T/m times, and thus in the worst case we make at most

$m \cdot T/m = T$ queries. The probability that this happens is $(ST/mN)^m$. With c corruptions, we first reduce m to $m - c$. Thus, for uniform passwords in $[N]$,

$$\mathbf{Adv}^{\text{ai-rec}}_{[N]^m,\ell,\perp,\pi}(S,T,c) \geq \left(\frac{ST}{(m-c)N}\right)^{m-c} .$$

With no corruptions, if we have sufficiently large side information, we may well need time $T = mN/S$. In particular, this means that we have a *direct sum* situation (without introducing salts). That is, the time to break m instances scales linearly with m.