Multi-Instance Security and its Application to Password-Based Cryptography

Mihir Bellare[*] Stefano Tessaro[†] Thomas Ristenpart[‡]

November 2011

Abstract

This paper develops a theory of multi-instance (mi) security and applies it to provide the first proof-based support for the classical practice of salting in password-based cryptography. Mi-security comes into play in settings (like password-based cryptography) where it is computationally feasible to compromise a single instance, and provides a second line of defense, aiming to ensure (in the case of passwords, via salting) that the effort to compromise all of some large number m of instances grows linearly with m. The first challenge is definitions, where we suggest LORX-security as a good metric for mi security of encryption and support this claim by showing it implies other natural metrics, illustrating in the process that even lifting simple results from the si setting to the mi one calls for new techniques. Next we provide a composition-based framework to transfer standard single-instance (si) security to mi-security with the aid of a key-derivation function. Analyzing password-based KDFs from the PKCS#5 standard to show that they meet our indifferentiability-style mi-security definition for KDFs, we are able to conclude with the first proof that per password salts amplify mi-security as hoped in practice. We believe that mi-security is of interest in other domains and that this work provides the foundation for its further theoretical development and practical application.

Keywords: Passwords, security amplification, indifferentiability, random oracles.

[*]Department of Computer Science & Engineering, University of California San Diego, USA.
[†]MIT CSAIL, USA.
[‡]Department of Computer Science, University of Wisconsin–Madison, USA.

Contents

1 Introduction

This paper develops a theory of *multi-instance security* and applies it to support practices in password-based cryptography.

Background. Password-based encryption (PBE) in practice is based on the PKCS#5 (equivalently, RFC 2898) standard [38]. It encrypts a message M under a password pw by picking a random s-bit *salt* sa, deriving a key $L \leftarrow \mathsf{KD}(pw \| sa)$ and returning $C' \leftarrow C \| sa$ where $C \leftarrow\!\!\$\ \mathcal{E}(L, M)$. Here $\mathcal{E}$ is a symmetric encryption scheme, typically an IND-CPA AES mode of operation, and key-derivation function (KDF) $\mathsf{KD}\colon \{0,1\}^* \to \{0,1\}^n$ is the c-fold iteration $\mathsf{KD} = H^c$ of a cryptographic hash function $H\colon \{0,1\}^* \to \{0,1\}^n$. However, passwords are often poorly chosen [32], falling within a set D called a "dictionary" that is small enough to exhaust. A brute-force attack now recovers the target password pw (thereby breaking the ind-cpa security of the encryption) using cN hashes where $N = |D|$ is the size of the dictionary.[1] Increasing c increases this effort, explaining the role of this iteration count, but c cannot be made too large without adversely impacting the performance of PBE.

Consider now m users, the i-th with password pw_i. If the salt is absent ($s = 0$), the number of hashes for the brute force attack to recover all m passwords remains around cN, but if s is large enough that salts are usually distinct, it rises to mcN, becoming prohibitive for large m. Salting, thus, aims to make the effort to compromise m target passwords scale linearly in m. (It has no effect on the security of encryption under any one, particular target password.)

New directions. This practice, in our view, opens a new vista in theoretical cryptography, namely to look at the multi-instance (mi) security of a scheme. We would seek metrics of security under which an adversary wins when it breaks all of m instances *but not if it breaks fewer*. This means that the mi security could potentially be much higher than the traditional single-instance (si) security. We would have security amplification.

Why do this? As the above discussion of password-based cryptography shows, there are settings where the computational effort t needed to compromise a single instance is feasible. Rather than give up, we provide a second line of defense. We limit the *scale* of the damage, ensuring (in the case of passwords, via the mechanism of salting) that the computational effort to compromise all of m instances is (around) tm and thus prohibitive for large m. We can't prevent the occasional illness, but we can prevent an epidemic.

We initiate the study of multi-instance security with a foundational treatment in two parts. The first part is agnostic to whether the setting is password-based or not, providing definitions for different kinds of mi-security of encryption and establishing relations between them, concluding with the message that what we call LORX-security is a good choice. The second part of our treatment focuses on password-based cryptography, providing a modular framework that proves mi-security of password-based primitives by viewing them as obtained by the composition of a mi-secure KDF with a si-secure primitive, and yielding in particular the first proof that salting works as expected to increase multi-instance security under a strong and formal metric for the latter.

Multi-instance security turns out to be challenging both definitionally (providing metrics where the adversary wins on breaking all instances but not fewer) and technically (reductions need to preserve tiny advantages and standard hybrid arguments no longer work). It also connects in interesting ways to security amplification via direct products and xor lemmas, eg. [43, 17, 21, 33, 15, 30, 40, 31, 41]. (We import some of their methods and export some novel viewpoints.) We believe there are many fruitful directions for future work, both theoretical (pursuing the connection with security amplification) and applied (mi security could be valuable in public-key cryptography where steadily improving attacks are making current security parameters look uncomfortably close to the edge for single-instance security). Let us now look at all this in some more detail.

LORX. We consider a setting with m independent target keys $K_1, \ldots, K_m$. (They may, but need not, be passwords.) In order to show that mi-security grows with m we want a metric (definition) where the

[1] Appendix A details this attack as well its multi-instance variant alluded to below.

$$\begin{tikzcd}
\text{LORX} \arrow[r, shift left, "\text{Th. 2.3}"] \arrow[d, dashed, "\text{Th 2.5}"'] \arrow[dr, "\text{Th 2.1}"'] & \text{RORX} \arrow[l, shift left, "2^m" description, "\text{Th. 2.2}"] \arrow[d, "2^m"] \\
\text{AND} & \text{UKU}
\end{tikzcd}$$

Figure 1: **Notions of multi-instance security for encryption and their relations.** LORX (left-or-right xor indistinguishability) emerges as the strongest, tightly implying RORX (real-or-random xor indistinguishability) and UKU (universal key-unrecoverability). The dashed line indicates that under some (mild, usually met) conditions LORX also implies AND. RORX implies LORX and UKU but with a 2^m loss in advantage where m is the number of instances, making LORX a better choice.

---

adversary wins if it breaks all m instances of the encryption but does not win if it breaks strictly fewer. If "breaking" is interpreted as recovery of the key then such a metric is easily given: it is the probability that the adversary recovers all m target keys. We refer to this as the UKU (Universal Key Unrecoverability) metric. But we know very well that key-recovery is a weak metric of encryption security. We want instead a mi analog of ind-cpa. The first thing that might come to mind is multi-user security [4, 3]. But in the latter the adversary wins (gets an advantage of one) even if it breaks just one instance so the mu-advantage of an adversary can never be less than its si (ind-cpa) advantage. We, in contrast, cannot "give up" once a single instance is broken. Something radically different is needed.

Our answer is LORX (left-or-right xor indistinguishability). Our game picks m independent challenge bits $b_1, \ldots, b_m$ and gives the adversary an oracle $\mathbf{Enc}(\cdot,\cdot,\cdot)$ which given i, M_0, M_1 returns an encryption of M_{b_i} under K_i. The adversary outputs a bit b' and its advantage is $2\Pr[b' = b_1 \oplus \cdots \oplus b_m] - 1$.[2] Why xor? Its well-known "sensitivity" means that even if the adversary figures out $m-1$ of the challenge bits, it will have low advantage unless it also figures out the last. This intuitive and historical support is strengthened by the relations, discussed below, that show that LORX implies security under other natural metrics.

Relations. The novelty of multi-instance security prompts us to step back and consider a broad choice of definitions. Besides UKU and LORX, we define RORX (real-or-random xor indistinguishability, a mi-adaptation of the si ROR notion of [5]) and a natural AND metric where the challenge bits $b_1, \ldots, b_m$ and oracle $\mathbf{Enc}(\cdot,\cdot,\cdot)$ are as in the LORX game but the adversary output is a vector $(b'_1, \ldots, b'_m)$ and its advantage is $\Pr[(b'_1, \ldots, b'_m) = (b_1, \ldots, b_m)] - 2^{-m}$. The relations we provide, summarized in Figure 1, show that LORX emerges as the best choice because it implies all the others with tight reductions. Beyond that, they illustrate that the mi terrain differs from the si one in perhaps surprising ways, both in terms of relations and the techniques needed to establish them.

Thus, in the si setting, LOR and ROR are easily shown equivalent up to a factor 2 in the advantages [5]. It continues to be true that LORX easily implies RORX but the hybrid argument used to prove that ROR implies LOR [5] does not easily extend to the mi setting and the proof that RORX implies LORX is not only more involved but incurs a factor 2^m loss.[3] In the si setting, both LOR and ROR are easily shown to imply KU (key unrecoverability). Showing LORX implies UKU is more involved, needing a boosting argument to ensure preservation of exponentially-vanishing advantages. This reduction is tight

[2] This is a simplification of our actual definition, which allows the adversary to adaptively corrupt instances to reveal the underlying keys and challenge bits. This capability means that LORX-security implies threshold security where the adversary wins if it predicts the xor of the challenge bits of some subset of the instances of its choice. See Section 2 for further justification for this feature of the model.

[3] This (exponential) 2^m factor loss is a natural consequence of the factor of 2 loss in the si case, our bound is tight, and the loss in applications is usually small because advantages are already exponentially vanishing in m. Nonetheless it is not always negligible and makes LORX preferable to RORX.

but, interestingly, the reduction showing RORX implies UKU is not, incurring a 2^m-factor loss, again indicating that LORX is a better choice. We show that LORX usually implies AND by exploiting a direct product theorem by Unger [41], evidencing the connections with this area. Another natural metric of mi-security is a threshold one, but our incorporation of corruptions means that LORX implies security under this metric.

Mi-security of PBE. Under the LORX metric, we prove that the advantage ϵ' obtained by a time t adversary against m instances of the above PBE scheme $\mathcal{E}'$ is at most $\epsilon + (q/mcN)^m$ (we are dropping negligible terms) where q is the number of adversary queries to RO H and ϵ is the advantage of a time t ind-cpa (si) adversary against $\mathcal{E}$. This is the desired result saying that salting works to provide a second line of defense under a strong mi security metric, amplifying security linearly in the number of instances.

Framework. This result for PBE is established in a modular (rather than ad hoc) way, via a framework that yields corresponding results for any password-based primitive. This means not only ones like password-based message authentication (also covered in PKCS#5) or password-based authenticated encryption (WinZip) but public-key primitives like password-based digital signatures, where the signing key is derived from a password. We view a password-based scheme for a goal as derived by composing a key-derivation function (KDF) with a standard (si) scheme for the same goal. The framework then has the following components. (1) We provide a definition of mi-security for KDFs. (2) We provide composition theorems, showing that composing a mi-secure KDF with a si-secure scheme for a goal results in a mi-secure scheme for that goal. (We will illustrate this for the case of encryption but similar results may be shown for other primitives.) (3) We analyze the iterated hash KDF of PKCS#5 and establish its mi security.

The statements above are qualitative. The quantitative security aspect is crucial. The definition of mi-security of KDFs must permit showing mi-security much higher than si-security. The reductions in the composition theorems must preserve exponentially vanishing mi-advantages. And the analysis of the PKCS#5 KDF must prove that the adversary advantage in q queries to the RO H grows as $(q/cmN)^m$, not merely q/cN. These quantitative constraints represent important technical challenges.

Mi-security of KDFs. We expand on item (1) above. The definition of mi-security we provide for KDFs is a simulation-based one inspired by the indifferentiability framework [29, 13]. The attacker must distinguish between the real world and an ideal counterpart. In both, target passwords $pw_1, \ldots, pw_m$ and salts $sa_1, \ldots, sa_m$ are randomly chosen. In the real world, the adversary gets input $(pw_1, sa_1, \mathsf{KD}(pw_1 \| sa_1)), \ldots, (pw_m, sa_m, \mathsf{KD}(pw_m \| sa_1))$ and also gets an oracle for the RO hash function H used by KD. In the ideal world, the input is $(pw_1, sa_1, L_1), \ldots, (pw_m, sa_m, L_m)$ where the keys $L_1, \ldots, L_m$ are randomly chosen, and the oracle is a simulator. The simulator itself has access to a **Test** oracle that will take a guess for a password and tell the simulator whether or not it matches one of the target passwords. Crucially, we require that when the number of queries made by the adversary to the simulator is q, the number of queries made by the simulator to its **Test** oracle is only q/c. This restriction is critical to our proof of security amplification and a source of challenges in the proof.

Related work. Previous work which aimed at providing proof-based assurances for password-based key-derivation has focused on the single-instance case and the role of iteration as represented by the iteration count c. Our work focuses on the multi-instance case and the roles of both salting and iteration.

The UNIX password hashing algorithm maps a password pw to $E^c_{pw}(0)$ where E is a blockcipher and 0 is a constant. Luby and Rackoff [27] show this is a one-way function when $c = 1$ and pw is a random blockcipher key. (So their result does not really cover passwords.) Wagner and Goldberg [42] treat the more general case of arbitrary c and keys that are passwords, but the goal continues to be to establish one-wayness and no security amplification (meaning increase in security with c) is shown. Boyen [10, 11] suggests various ways to enhance security, including letting users pick their own iteration counts.

Yao and Yin [44] give a natural pseudorandomness definition of a KDF in which the attacker gets (K, sa) where K is either $H^c(pw \| sa)$ or a random string of the same length and must determine which. Modeling H as a random oracle (RO) [7] to which the adversary makes q queries, they claim to prove that the adversary's advantage is at most q/cN plus a negligible term. This would establish single-instance security amplification by showing that iteration works as expected to increase attacker effort.[4] However, even though salts are considered, this does not consider multi-instance security let alone establish multi-instance security amplification, and their definition of KDF security does not adapt to allow this. (We use, as indicated above, an indifferentiability-style definition.) In fact the KDF definition of [44] is not even sufficient to establish si security of password-based encryption in the case the latter, as specified in PKCS#5, picks a fresh salt for each message encrypted.

KDFs are for use in non-interactive settings like encryption with WinZip. The issues and questions we consider do not arise with password authenticated key exchange (PAKE), where two parties in joint possession of a password interact to agree on a cryptographic session key in a way that resists off-line dictionary attack [9, 22, 6, 24, 12, 16, 2, 37, 20]. PAKE definitions already guarantee that the session key may be safely used for encryption. There are no salts and no amplification issues.

Abadi and Warinschi [1] provide a si, key-recovery definition for PBE security and connect this with symbolic notions. They do not consider mi security.

Dodis, Gennaro, Håstad, Krawczyk and Rabin [14] treat statistically-secure key derivation using hash functions and block ciphers. As discussed in-depth by Kracwzyk [26], these results and techniques aren't useful for password-based KDFs because passwords aren't large enough, let alone have the sufficient amount of min-entropy. Krawczyk [26] also notes that his two-stage KDF approach could be used to build password-based KDFs by replacing the extraction stage with a key-stretching operation. Our general framework may be used to analyze the mi-security of this as well.

2 The Multi-Instance Terrain

This section defines metrics of multi-instance encryption security and explores the relations between them to establish the notions and results summarized in Figure 1. Our treatment intends to show that the mi terrain is different from the si one in fundamental ways, leading to new definitions, challenges and connections.

2.1 Metrics of mi security

Recall that a symmetric encryption scheme is a triple of algorithms $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation algorithm $\mathcal{K}$ outputs a key. The encryption algorithm $\mathcal{E}$ takes a key K and a message M and outputs a ciphertext $C \leftarrow\$ \mathcal{E}(K, M)$. The deterministic decryption algorithm $\mathcal{D}$ takes the key K and a ciphertext C to return either a string or $\perp$. Correctness requires that $\Pr[\mathcal{D}(K, \mathcal{E}(K, M)) = M]$ for all M where the probability is over $K \leftarrow\$ \mathcal{K}$ and the coins of $\mathcal{E}$.

To illustrate the issues and choices in defining mi security, we start with key unrecoverability which is simple because it is underlain by a computational game and its mi counterpart is easily and uncontentiously defined. When we move to stronger notions underlain by decisional games, definitions will get more difficult and more contentious as more choices will emerge.

UKU. Single-instance key unrecoverability is formalized via the game $\mathrm{KU}_{\mathsf{SE}}$ where a target key $K \leftarrow\$ \mathcal{K}$ is initially sampled, and the adversary $\mathcal{A}$ is given an oracle **Enc** which, on input M, returns $\mathcal{E}(K, M)$. Finally, the adversary is asked to output a guess K' for the key, and the game returns true if $K = K'$, and false otherwise. An mi version of the game, $\mathrm{UKU}_{\mathsf{SE},m}$, is depicted in Figure 2. It picks an m-vector $\mathbf{K}$ of target keys and the oracle **Enc** now takes i, M to return $\mathcal{E}(\mathbf{K}[i], M)$. The **Cor** oracle gives the adversary the capability of corrupting a user to obtain its target key. The adversary's output guess is also a m-vector $\mathbf{K}'$ and the game returns the boolean $(\mathbf{K} = \mathbf{K}')$, meaning the adversary wins only if it recovers *all* the target keys. (The "U" in "UKU" reflects this, standing for "Universal.") The advantage of adversary $\mathcal{A}$ is

[4] Unfortunately, we point in Appendix B to a bug in the proof of [44, Lemma 2.2] and explain why the bound claimed by [44, Theorem 1] is wrong. Beyond this, the proof makes some rather large and not fully justified jumps. The special case $m = 1$ of our treatment will fill these gaps and recover the main claim of [44].

```
main $\mathrm{UKU}^{\mathcal{A}}_{\mathsf{SE},m}$
    $\mathbf{K}[1],\ldots,\mathbf{K}[m] \leftarrow\$ \mathcal{K}$
    $\mathbf{K}' \leftarrow\$ \mathcal{A}^{\mathbf{Enc}}$
    Ret $\mathbf{K}' = \mathbf{K}$

proc. $\mathbf{Enc}(i, M)$
    Ret $\mathcal{E}(\mathbf{K}[i], M)$

proc. $\mathbf{Cor}(i)$
    Ret $\mathbf{K}[i]$
```

```
main $\mathrm{LORX}^{\mathcal{A}}_{\mathsf{SE},m}$
    $\mathbf{K}[1],\ldots,\mathbf{K}[m] \leftarrow\$ \mathcal{K}$
    $\mathbf{b} \leftarrow\$ \{0,1\}^m$
    $b' \leftarrow\$ \mathcal{A}^{\mathbf{Enc}}$
    Ret $(b' = \oplus_i \mathbf{b}[i])$

main $\mathrm{AND}^{\mathcal{A}}_{\mathsf{SE},m}$
    $\mathbf{K}[1],\ldots,\mathbf{K}[m] \leftarrow\$ \mathcal{K}$
    $\mathbf{b} \leftarrow\$ \{0,1\}^m$
    $\mathbf{b}' \leftarrow\$ \mathcal{A}^{\mathbf{Enc}}$
    Ret $(\mathbf{b}' = \mathbf{b})$

proc. $\mathbf{Enc}(i, M_0, M_1)$
    If $|M_0| \neq |M_1|$ then Ret $\bot$
    $C \leftarrow\$ \mathcal{E}(\mathbf{K}[i], M_{\mathbf{b}[i]})$
    Ret $C$

proc. $\mathbf{Cor}(i)$
    Ret $(\mathbf{K}[i], \mathbf{b}[i])$
```

```
main $\mathrm{RORX}^{\mathcal{A}}_{\mathsf{SE},m}$
    $\mathbf{K}[1],\ldots,\mathbf{K}[m] \leftarrow\$ (\{0,1\}^k)^m$
    $\mathbf{b} \leftarrow\$ \{0,1\}^m$
    $b' \leftarrow\$ \mathcal{A}^{\mathbf{Enc}}$
    Ret $(b' = \oplus_i \mathbf{b}[i])$

proc. $\mathbf{Enc}(i, M)$
    $C_1 \leftarrow\$ \mathcal{E}(\mathbf{K}[i], M)$
    $M_0 \leftarrow\$ \{0,1\}^{|M|}$
    $C_0 \leftarrow\$ \mathcal{E}(\mathbf{K}[i], M_0)$
    Ret $C_{\mathbf{b}[i]}$

proc. $\mathbf{Cor}(i)$
    Ret $(\mathbf{K}[i], \mathbf{b}[i])$
```

Figure 2: Multi instance security notions for encryption.

$\mathbf{Adv}^{\mathrm{uku}}_{\mathsf{SE},m}(\mathcal{A}) = \Pr[\mathrm{UKU}^{\mathcal{A}}_{\mathsf{SE},m} \Rightarrow \mathsf{true}]$. Naturally, this advantage depends on the adversary's resources. (It could be 1 if the adversary corrupts all instances.) We say that $\mathcal{A}$ is a $(t, \mathbf{q}, q_c)$-adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\mathbf{Enc}(i, \cdot)$ and makes at most q_c corruption queries. Then we let $\mathbf{Adv}^{\mathrm{uku}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{uku}}_{\mathsf{SE},m}(\mathcal{A})$ where the maximum is over all $(t, \mathbf{q}, q_c)$-adversaries.

AND. Single-instance indistinguishabilty for symmetric encryption is usually formalized via left-or-right security [5]. A random bit b and key $K \leftarrow\$ \mathcal{K}$ are chosen, and an adversary $\mathcal{A}$ is given access to an oracle **Enc** that given equal-length messages M_0, M_1 returns $\mathcal{E}(K, M_b)$. The adversary outputs a bit b' and its advantage is $2\Pr[b = b'] - 1$. There are several ways one might consider creating an mi analog. Let us first consider a natural AND-based metric based on game $\mathrm{AND}_{\mathsf{SE},m}$ of Figure 2. It picks at random a vector $\mathbf{b} \leftarrow\$ \{0,1\}^m$ of challenge bits as well as a vector $\mathbf{K}[1], \ldots, \mathbf{K}[m]$ of keys, and the adversary is given access to oracle **Enc** that on input i, M_0, M_1, where $|M_0| = |M_1|$, returns $\mathcal{E}(\mathbf{K}[i], M_{\mathbf{b}[i]})$. Additionally, the corruption oracle **Cor** takes i and returns the pair $(\mathbf{K}[i], \mathbf{b}[i])$. The adversary finally outputs a bit vector $\mathbf{b}'$, and wins if and only if $\mathbf{b} = \mathbf{b}'$. (It is equivalent to test that $\mathbf{b}[i] = \mathbf{b}'[i]$ for all uncorrupted i.) The advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}^{\mathrm{and}}_{\mathsf{SE},m}(\mathcal{A}) = \Pr[\mathrm{AND}^{\mathcal{A}}_{\mathsf{SE},m} \Rightarrow \mathsf{true}] - 2^{-m}$. We say that $\mathcal{A}$ is a $(t, \mathbf{q}, q_c)$-adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\mathbf{Enc}(i, \cdot, \cdot)$ and makes at most q_c corruption queries. Then we let $\mathbf{Adv}^{\mathrm{and}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{and}}_{\mathsf{SE},m}(\mathcal{A})$ where the maximum is over all $(t, \mathbf{q}, q_c)$-adversaries.

This metric has many points in its favor. By (later) showing that security under it is implied by security under our preferred LORX metric, we automatically garner whatever value it offers. But the AND metric also has weaknesses that in our view make it inadequate as the primary choice. Namely, it does not capture the hardness of breaking *all* the uncorrupted instances. For example, an adversary that corrupts instances $1, \ldots, m-1$ to get $\mathbf{b}[1], \ldots, \mathbf{b}[m-1]$, makes a random guess g for $\mathbf{b}[m]$ and returns $(\mathbf{b}[1], \ldots, \mathbf{b}[m-1], g)$ has the high advantage $0.5 - 2^{-m}$ without breaking all instances. We prefer a metric where this adversary's advantage is close to 0.

LORX. To overcome the above issue with the AND advantage, we introduce the XOR advantage measure and use it to define LORX. Game $\mathrm{LORX}_{\mathsf{SE},m}$ of Figure 2 makes its initial choices the same way as game $\mathrm{AND}_{\mathsf{SE},m}$ and provides the adversary with the same oracles. However, rather than a vector, the adversary must output a bit b', and wins if this equals $\mathbf{b}[1] \oplus \cdots \oplus \mathbf{b}[m]$. (It is equivalent to test that $b' = \oplus_{i \in S} \mathbf{b}[i]$ where S is the uncorrupted set.) The advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(\mathcal{A}) = 2\Pr[\mathrm{LORX}^{\mathcal{A}}_{\mathsf{SE},m} \Rightarrow \mathsf{true}] - 1$. We say that $\mathcal{A}$ is a $(t, \mathbf{q}, q_c)$-adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\mathbf{Enc}(i, \cdot, \cdot)$ and makes at most q_c corruption queries. Then we let $\mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(\mathcal{A})$ where the maximum is over all $(t, \mathbf{q}, q_c)$-adversaries. Returning to the example we gave for the AND case, if an adversary corrupts the first $m-1$ instances to get back $\mathbf{b}[1], \ldots, \mathbf{b}[m-1]$, makes a random guess g for $\mathbf{b}[m]$ and outputs $b' = \mathbf{b}[1] \oplus \cdots \oplus \mathbf{b}[m-1] \oplus g$, it will have advantage 0.

RORX. A variant of the si LOR notion, ROR, was given in [5]. Here the adversary must distinguish between an encryption of a message M it provides and the encryption of a random message of length $|M|$. This was shown equivalent to LOR up to a factor 2 in the advantages [5]. This leads us to define the mi analog RORX and ask how it relates to LORX. Game $\mathrm{RORX}_{\mathsf{SE},m}$ of Figure 2 makes its initial choices the same way as game $\mathrm{LORX}_{\mathsf{SE},m}$. The adversary is given access to oracle **Enc** that on input i, M, returns $\mathcal{E}(\mathbf{K}[i], M)$ if $\mathbf{b}[i] = 1$ and otherwise returns $\mathcal{E}(\mathbf{K}[i], M_1)$ where $M_1 \leftarrow\!\!\$\ \{0,1\}^{|M|}$. It also gets the usual **Cor** oracle. It outputs a bit b' and wins if this equals $\mathbf{b}[1] \oplus \cdots \oplus \mathbf{b}[m]$. The advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}^{\mathrm{rorx}}_{\mathsf{SE},m}(\mathcal{A}) = 2\Pr[\mathrm{RORX}^{\mathcal{A}}_{\mathsf{SE},m} \Rightarrow \mathsf{true}] - 1$. We say that $\mathcal{A}$ is a $(t, \mathbf{q}, q_c)$-adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\mathbf{Enc}(i, \cdot)$ and makes at most q_c corruption queries. Then we let $\mathbf{Adv}^{\mathrm{rorx}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{rorx}}_{\mathsf{SE},m}(\mathcal{A})$ where the maximum is over all $(t, \mathbf{q}, q_c)$-adversaries.

Discussion. The multi-user security goal from [4] gives rise to a version of the above games without corruptions and where all instances share the same challenge bit b, which the adversary tries to guess. But it is easy to see that this does *not* measure mi security, since recovering a single key, for example, suffices to learn b.

The above approach extends naturally to providing a mi counterpart to any security definition based on a decisional game, where the adversary needs to guess a bit b. For example we may similarly create mi metrics of CCA security.

Why does the model include corruptions? The following example may help illustrate. Suppose SE is entirely insecure when the key has first bit 0 and highly secure otherwise. (From the si perspective, it is insecure.) In the LORX game, an adversary will be able to figure out around half the challenge bits. If we disallow corruptions, it would still have very low advantage. From the application point of view, this seems to send the wrong message. We want LORX-security to mean that the probability of "large scale" damage is low. But breaking half the instances is pretty large scale. Allowing corruptions removes this defect because the adversary could corrupt the instances it could not break and then, having corrupted only around half the instances, get a very high advantage, breaking LORX-security. In this way, we may conceptually keep the focus on an adversary goal of breaking *all* instances, yet cover the case of breaking some threshold number via the corruption capability.

An alternative way to address the above issue without corruptions is to define threshold metrics where the adversary wins by outputting a dynamically chosen set S and predicting the xor of the challenge bits for the indexes in S. This, again, has much going for it as a metric. But LORX with corruptions, as we define it, will imply security under this metric.

2.2 Relations

We provide formal result statements and proofs in support of the implications claimed in Figure 1.

LORX implies UKU. In the si setting, it is easy to see that LOR security implies KU security. The LOR adversary simply runs the KU adversary. When the latter makes oracle query M, the LOR adversary queries its own oracle with M, M and returns the outcome to the KU adversary. When the latter returns a key K', the LOR adversary submits a last oracle query consisting of a pair M_0, M_1 of random messages to get back a challenge ciphertext C, returning 1 if $\mathcal{D}(K', C) = M_1$ and 0 otherwise. A similar but slightly more involved proof shows that ROR implies KU.

It is important to establish analogs of these basic results in the mi setting, for they function as "tests" for the validity of our mi notions. The following shows that LORX security implies UKU. Interestingly, it is not as simple to establish in the mi case as in the si case. Also, as we will see later, the proof that RORX implies UKU is not only even more involved but incurs a factor 2^m loss, making LORX a better choice as the metric to target in designs.

Theorem 2.1 [LORX $\Rightarrow$ UKU] Let $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with message space $\mathcal{M}$, and let ℓ be such that $\{0,1\}^\ell \subseteq \mathcal{M}$. Then, for all t, q_c, and $\mathbf{q}$, and for all $k \geq 1$,

$$\mathbf{Adv}^{\mathrm{uku}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c) \leq \mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(t', \mathbf{q}', q_c) + m \cdot \left(\frac{1}{2^\ell - 1}\right)^k ,$$

where $t' = t + O(m \cdot k)$, and $\mathbf{q}'[i] = \mathbf{q}[i] + k$ for all $i = 1, \ldots, m$. ▌

The proof is given in Appendix C. Here, let us stress Theorem 2.1 surfaces yet another subtlety of the mi setting. At first, it would seem that proving the case $k = 1$ of the theorem is sufficient (this is what usually done in the si case). However, it is crucial to remark that $\mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(t', \mathbf{q}', q_c)$ may be *very* small. For example, it is not unreasonable to expect $2^{-128 \cdot m}$ if SE is secure in the single-instance setting. Yet, assume that $\mathcal{E}$ encrypts 128-bit messages, then we are only able to set $\ell = 128$, in turn making $m/(2^\ell - 1) \approx m \cdot 2^{-128}$ by far the leading term on the right-hand side. The parameter k hence opens the door to fine tuning of the additive extra term at the cost of an additive complexity loss in the reduction. Also note that the reduction in the proof of Theorem 2.1 is not immediate, as an adversary guessing all the keys in the UKU game with probability ϵ only yields an adversary recovering all the bits $\mathbf{b}[1], \ldots, \mathbf{b}[m]$ in the LORX game with probability ϵ. Just outputting the xor of these bits is not sufficient, as we have to boost the success probability to $\frac{1+\epsilon}{2}$ in order to obtain the desired relation between the two advantage measures.

In analogy to the si setting, UKU does not imply LORX. Just take a scheme $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ encrypting n-bit messages which is UKU-secure, and modify it into a scheme $\mathsf{SE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ where $\mathcal{K} = \mathcal{K}'$ and $\mathcal{E}'(K, M) = \mathcal{E}'(K, M) \,\|\, M[0]$, with $M[0]$ being the first bit of M. Clearly, SE' is still UKU-secure but not LORX-secure

As indicated above, a proof that RORX implies UKU is much more involved and incurs a factor 2^m loss. Roughly speaking, this is because in the si case, in the reduction needed to prove that ROR implies KU, the ROR adversary can only simulate the execution of the KU adversary correctly in the case where the bit is 1, i.e., the encryption oracle returns the actual encryption of a message. This results in a factor two loss in terms of advantage. Upon translating this technique to the mi case, the factor 2 becomes 2^m, as *all* bits need to be 1 for the UKU adversary to output the right keys with some guaranteed probability. However, we will not follow this route for the proof of this result. Instead, we can obtain the same result by combining Theorem 2.2 and Theorem 2.1.

LORX VERSUS RORX. In the si setting, LOR and ROR are the same up to a factor 2 in the advantage [5]. The LOR implies ROR implication is trivial and ROR implies LOR is a simple hybrid argument. We now discuss the relation between the mi counterparts, namely RORX and LORX, which is both more complex and more challenging to establish. We first show the harder direction, namely that RORX security implies LORX security. We find an (inevitable) factor 2^m loss in the reduction. The difficulty is adapting the hybrid argument technique to the mi setting.

Theorem 2.2 [RORX $\Rightarrow$ LORX] Let $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For all m, $t, q_c > 0$, and all vectors $\mathbf{q}$ we have $\mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c) \leq 2^m \cdot \mathbf{Adv}^{\mathrm{rorx}}_{\mathsf{SE},m}(t', \mathbf{q}, q_c)$, where $t' = t + \mathcal{O}(1)$. ▌

As discussed in Section 1, the multiplicative factor 2^m is often of no harm because advantages are already exponentially small in m. The factor is natural, being the mi analogue of the factor 2 appearing in the traditional si proof, and examples can be given showing that the bound is tight.

Proof: Let $\mathcal{A}$ be $(t, \mathbf{q}, q_c)$-distinguisher for $\mathrm{LORX}_{\mathsf{SE},m}$. We build a $(t', \mathbf{q}, q_c)$-distinguisher $\mathcal{B}$ for $\mathrm{RORX}_{\mathsf{SE},m}$ which simulates the execution of $\mathcal{A}$ in $\mathrm{LORX}_{\mathsf{SE},m}$ using the oracles of $\mathrm{RORX}_{\mathsf{SE},m}$. More in detail, $\mathcal{B}$ first selects a random bit-vector $\mathbf{c} \leftarrow\!\!\$\, \{0,1\}^m$, and whenever $\mathcal{A}$ asks a query $\mathbf{Enc}(i, M_0, M_1)$, $\mathcal{B}$ queries $\mathbf{Enc}(i, M_{\mathbf{c}[i]})$, returning the resulting ciphertext to $\mathcal{A}$. Corruption queries $\mathbf{Cor}(i)$ made by $\mathcal{A}$ are simply forwarded to $\mathrm{RORX}_{\mathsf{SE},m}$, and their outputs returned to $\mathcal{A}$. Finally, when $\mathcal{A}$ terminates and outputs a bit b', $\mathcal{B}$ terminates and outputs the bit $b' \oplus \bigoplus_{i=1}^m \mathbf{c}[i] \oplus (m \bmod 2)$, with $m \bmod 2$ denoting the parity of m.

To simplify the analysis of $\mathcal{B}$'s success probability, let us introduce the shorthand $p(\mathbf{b}', \mathbf{c}')$ for all $\mathbf{b}', \mathbf{c}' \in \{0,1\}^m$, denoting the success probability of $\mathcal{B}$ winning $\mathrm{RORX}_{\mathsf{SE},m}$ conditioned on the game setting $\mathbf{b} = \mathbf{b}'$

and $\mathcal{B}$ choosing $\mathbf{c} = \mathbf{c}'$. Similarly, let $q(\mathbf{b}')$ be the probability that $\mathcal{A}$ wins $\mathrm{LORX}_{\mathsf{SE},m}$ conditioned on $\mathbf{b} = \mathbf{b}'$ in the game. It is easy to see that $q(\mathbf{b}') = p(1^m, \mathbf{b}')$ for every $\mathbf{b}' \in \{0,1\}^m$, where 1^m is the all-one vector. This is because when $\mathbf{b} = 1^m$ in $\mathrm{RORX}_{\mathsf{SE},m}$, a query $\mathbf{Enc}(i, M_0, M_1)$ by the simulated $\mathcal{A}$ is answered with $\mathbf{Enc}(i, M_{\mathbf{c}[i]}) = \mathbf{Enc}(i, M_{\mathbf{b}'[i]})$ by $\mathcal{B}$. Moreover, again in the case where $\mathbf{b} = 1^m$, the adversary $\mathcal{B}$ wins $\mathrm{RORX}_{\mathsf{SE},m}$ if $\mathcal{A}$ outputs b' such that $b' \oplus \bigoplus_i \mathbf{b}'[i] \oplus (m \bmod 2) = \bigoplus_i \mathbf{b}[i] = m \bmod 2$, which is equivalent to $b' = \bigoplus_i \mathbf{b}'[i]$.

We also note that for all vectors $\mathbf{b}', \mathbf{c}', \mathbf{c}'' \in \{0,1\}^m$, if there exists $i \in [1\,..\,m]$ such that $\mathbf{b}'[i] = 0$, $\mathbf{c}'[i] = 0$, and $\mathbf{c}''[i] = 1$, then $p(\mathbf{b}', \mathbf{c}') + p(\mathbf{b}', \mathbf{c}'') = 1$. This is due to the fact that the probability that $\mathcal{A}$, when run by $\mathcal{B}$ using $\mathbf{c} = \mathbf{c}'$ in $\mathrm{RORX}_{\mathsf{SE},m}$ with $\mathbf{b} = \mathbf{b}'$, outputs a certain bit b' is the same as when $\mathcal{B}$ runs with $\mathbf{c} = \mathbf{c}''$, because $\mathcal{A}$'s queries $\mathbf{Enc}(i, M_0, M_1)$ are answered with the encryption of a random plaintext in both cases. However, the bits output by $\mathcal{B}$ when $\mathbf{c} = \mathbf{c}'$ and when $\mathbf{c} = \mathbf{c}''$ are exactly the complement of each other, and hence $p(\mathbf{b}', \mathbf{c}'') = 1 - p(\mathbf{b}', \mathbf{c}')$. Further note that as we can always pair strings $\mathbf{c}' \in \{0,1\}^m$ so that they differ in *exactly* one component (just take any perfect matching of the m-dimensional hypercube), then $\sum_{\mathbf{c}' \in \{0,1\}^m} p(\mathbf{b}', \mathbf{c}') = 2^{m-1}$ holds for all $\mathbf{b}' \neq 1^m$.

Putting pieces together, and using $p(\cdot,\cdot)$ and $q(\cdot)$ to express winning probabilities,

$$\Pr\left[\mathrm{RORX}^{\mathcal{B}}_{\mathsf{SE},m} \Rightarrow \mathsf{true}\right] = \frac{1}{2^{2m}} \sum_{\mathbf{b}', \mathbf{c}' \in \{0,1\}^m} p(\mathbf{b}', \mathbf{c}') = \frac{1}{2^{2m}} \left(\sum_{\mathbf{c}' \in \{0,1\}^m} p(1^m, \mathbf{c}') + \sum_{\mathbf{b}' \neq 1^m} \sum_{\mathbf{c}' \in \{0,1\}^m} p(\mathbf{b}', \mathbf{c}') \right)$$

$$= \frac{1}{2^{2m}} \left(\sum_{\mathbf{b}' \in \{0,1\}^m} q(\mathbf{b}') + (2^m - 1) \cdot 2^{m-1} \right) = \frac{1}{2^m} \Pr\left[\mathrm{LORX}^{\mathcal{A}}_{\mathsf{SE},m} \Rightarrow \mathsf{true}\right] + \frac{1}{2} - \frac{1}{2^{m+1}} \,.$$

Rearranging terms yields $2^m \cdot \mathbf{Adv}^{\mathrm{rorx}}_{\mathsf{SE},m}(\mathcal{B}) = \mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(\mathcal{A})$. It is easy to see that $\mathcal{B}$ is a $(t', \mathbf{q}, q_c)$-distinguisher, and the theorem follows by maximizing over all $(t, \mathbf{q}, q_c)$-distinguishers $\mathcal{A}$. ▌

We omit the much simpler proof of the converse.

Theorem 2.3 [LORX $\Rightarrow$ RORX] Let $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. For all m, $t, q_c > 0$, and all vectors $\mathbf{q}$ we have $\mathbf{Adv}^{\mathrm{rorx}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c) \leq \mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(t', \mathbf{q}, q_c)$, where $t' = t + \mathcal{O}(1)$. ▌

LORX IMPLIES AND. Intuitively, one might expect AND security to be a *stronger* requirement than LORX security, as the former seems easier to break than the latter. However we show that under a fairly minimal requirement, LORX implies AND. This brings another argument in support of LORX: Even if an application requires AND security, it turns out that proving LORX security is generally sufficient. The proof relies on the following probabilistic lemma, due to Unger [41].

Lemma 2.4 [41] Let $Y_1, \ldots, Y_m \in \{0,1\}$ be random variables such that there exist $\beta_1, \ldots, \beta_m \in [-1,1]$ and $C, \gamma > 0$ with $\Pr\left[\bigoplus_{i \in S} Y_i = 0\right] \leq (1 + C \cdot \prod_{i \in S} \beta_i + \gamma)/2$ for all $S \subseteq \{1, \ldots, m\}$. Then,

$$\Pr\left[\textstyle\sum_{i=1}^m Y_i = m\right] \;\leq\; \gamma + C \cdot \prod_{i \in S} \frac{1 + \beta_i}{2} \,. \;▌$$

The following theorem is to be interpreted as follows: In general, if we only know that $\mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c)$ is small, we do not know how to prove $\mathbf{Adv}^{\mathrm{and}}_{\mathsf{SE},m}(t', \mathbf{q}, q_c)$ is also small (for $t' \approx t$), or whether this is true at all. As we sketched above, the reason is that we do not know how to use an adversary $\mathcal{A}$ for which the $\mathrm{AND}_{\mathsf{SE},m}$ advantage is large to construct an adversary for which the $\mathrm{LORX}_{\mathsf{SE},m}$ advantage is large. Still, one would expect that such an adversary *might* more easily yield one for which the $\mathrm{LORX}_{\mathsf{SE},k}$ advantage is sufficiently large, for *some* $k \leq m$. The following theorem uses the above probabilistic lemma to confirm this intuition.

Theorem 2.5 Let $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Further, let m, t, $\mathbf{q}$, and q_c be given, and assume that there exist C, ϵ, and γ such that for all $1 \leq i \leq m$,

$$\max_{S \subseteq \{1,\ldots,m\}, |S| = i} \mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},i}(t^*_S, \mathbf{q}[S], q_c) \leq C \cdot \epsilon^i + \gamma \,,$$

where $\mathbf{q}[S]$ is the projection of $\mathbf{q}$ on the components in S, and $t_S^* = t + \mathcal{O}(t_{\mathcal{E}} \cdot \sum_{i \notin S} \mathbf{q}[i])$, with $t_{\mathcal{E}}$ denoting the running time needed for one encryption with $\mathcal{E}$. Then, $\mathbf{Adv}^{\mathrm{and}}_{\mathsf{SE},m}(t, \mathbf{q}, q_c) \leq \gamma + C \cdot \prod_{i=1}^{m}(1 + \epsilon_i)/2$. ▮

Does the converse also hold true? It is worth mentioning that *in general* we are not able to prove that AND implies LORX. Still, we note that in the corruption-free case, one can upper bound $\mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},m}(t, \mathbf{q}, 0)$ in terms of $\mathbf{Adv}^{\mathrm{and}}_{\mathsf{SE},m'}(t', \mathbf{q}', 0)$ for $m' \approx 2m$ and t' and $\mathbf{q}'$ being much larger than $t, \mathbf{q}$. The proof, which we omit, follows the lines of the proof of the XOR Lemma from the Direct Product Theorem given by Goldreich, Nisan, and Wigderson [19], and relies on the Goldreich-Levin theorem [18]. As the loss in concrete security in this reduction is very large, and it only holds in the corruption-free statement, we find this an additional argument to support the usage of the LORX metric.

3 Password-based Encryption via KDFs

We now turn to our main motivating application, that of password based encryption (PBE) as specified in PKCS#5 [38]. The schemes specified there combine a conventional mode of operation (e.g., CBC mode) with a password-based key derivation function (KDF). We start with formalizing the latter.

Password-based KDFs. Formally, a (k, s, c)-KDF is a deterministic map $\mathsf{KD}\colon \{0,1\}^* \times \{0,1\}^s \to \{0,1\}^k$ that may make use of an underlying ideal primitive. Here c is the iteration count, which specifies the multiplicative increase in work that should slow down brute force attacks.

PKCS#5 describes two KDFs [38]. We treat the first in detail and discuss the second in Appendix E. Let $\mathsf{KD1}^H(pw, sa) = H^c(pw \,\|\, sa)$ where H^c is the function that composes H with itself c times. To generalize beyond concatenation, we can define a function $\mathsf{Encode}(pw, sa)$ that describes how to encode its inputs onto $\{0,1\}^*$ with efficiently computable inverse $\mathsf{Decode}(W)$.

PBE schemes. A PBE scheme is just a symmetric encryption scheme where we view the keys as passwords and key generation as a password sampling algorithm. To highlight when we are thinking of key generation as password sampling we will use $\mathcal{P}$ to denote key generation (instead of $\mathcal{K}$). We will also write pw for a key that we think of as a password. Let KD be a (k, s, c)-KDF and let $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme with $\mathcal{K}$ outputting uniformly selected k-bit keys. Then we define the PBE scheme $\mathcal{SE}[\mathsf{KD}, \mathsf{SE}] = (\mathcal{P}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ as follows. Encryption $\overline{\mathcal{E}}(pw, M)$ is done via $sa \leftarrow\!\!{\scriptstyle\$}\; \{0,1\}^s$; $K \leftarrow \mathsf{KD}(pw, sa)$; $C \leftarrow\!\!{\scriptstyle\$}\; \mathcal{E}(K, M)$, returning (sa, C) as the ciphertext. Decryption recomputes the key K by reapplying the KDF and then applies $\mathcal{D}$. If the KDF is $\mathsf{KD1}$ and the encryption scheme is CBC mode, then one obtains the first PBE scheme from PKCS#5 [38].

Password guessing. We aim to show that security of the above constructions holds up to the amount of work required to brute-force the passwords output by $\mathcal{P}$. This begs the question of how we measure the strength of a password sampler. We will formalize the hardness of guessing passwords output by some sampler $\mathcal{P}$ via an adaptive guessing game: It challenges an adversary with guessing passwords adaptively in a setting where the attacker may, also, adaptively learn some passwords via a corruption oracle. Concretely, let $\mathrm{GUESS}_{\mathcal{P},m}$ be the game defined in Figure 3. A (q_t, q_c)-*guessing adversary* is one that makes at most q_t queries to **Test** and q_c queries to **Cor**. An adversary $\mathcal{B}$'s guessing advantage is $\mathbf{Adv}^{\mathrm{guess}}_{\mathcal{P},m}(\mathcal{B}) = \Pr\left[\mathrm{GUESS}^{\mathcal{B}}_{\mathcal{P},m} \Rightarrow \mathsf{true}\right]$. We assume without loss of generality that $\mathcal{A}$ does not make any *pointless queries*: (1) repeated queries to **Cor** on the same value; (2) a query $\mathbf{Test}(i, \cdot)$ following a query of $\mathbf{Cor}(i)$; and (3) a query $\mathbf{Cor}(i)$ after a query $\mathbf{Test}(i, pw)$ that returned true. We also define a variant of the above guessing game that includes salts and allows an attacker to test password-salt pairs against all m instances simultaneously. This will be useful as an intermediate step when reducing to guessing advantage. The game $\mathrm{saGUESS}_{\mathcal{P},m,\rho}$ is shown in Figure 3 and we define advantage via $\mathbf{Adv}^{\mathrm{sa\text{-}guess}}_{\mathcal{P},m}(\mathcal{B}) = \Pr\left[\mathrm{saGUESS}^{\mathcal{B}}_{\mathcal{P},m} \Rightarrow \mathsf{true}\right]$. An easy argument proves the following lemma.

Lemma 3.1 Let $m, \rho > 0$ and $\mathcal{P}$ be a password sampler. Let $\mathcal{A}$ be an (q_t, q_c)-guessing $\mathrm{GUESS}_{\mathcal{P},m}$ adversary. Then there exists a (q_t, q_c)-guessing $\mathrm{saGUESS}_{\mathcal{P},m,rho}$ adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\mathrm{sa\text{-}guess}}_{\mathcal{P},m,\rho}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{guess}}_{\mathcal{P},m}(\mathcal{B}) + m^2\rho^2/2^s$. $\square$

```
main GUESS$_{\mathcal{P},m}$
$\mathbf{pw}[1],\ldots,\mathbf{pw}[m] \leftarrow\$ \mathcal{P}$
$\mathbf{pw}' \leftarrow\$ \mathcal{B}^{\mathbf{Test},\mathbf{Cor}}$
Ret $\bigwedge_{i=1}^{m}(\mathbf{pw}'[i] = \mathbf{pw}[i])$

proc. Test$(i, pw)$
If $(pw = \mathbf{pw}[i])$ then Ret true
Ret $\perp$

proc. Cor$(i)$
Ret $\mathbf{pw}[i]$
```

```
main saGUESS$_{\mathcal{P},m,\rho}$
$\mathbf{pw}[1],\ldots,\mathbf{pw}[m] \leftarrow\$ \mathcal{P}$
For $i = 1$ to $m$ do
    For $j = 1$ to $\rho$ do
        $\mathbf{sa}[i,j] \leftarrow\$ \{0,1\}^s$
$\mathbf{pw}' \leftarrow\$ \mathcal{B}^{\mathbf{Test},\mathbf{Cor}}(\mathbf{sa})$
Ret $\bigwedge_{i=1}^{m}(\mathbf{pw}'[i] = \mathbf{pw}[i])$

proc. Test$(pw, sa)$
For $i = 1$ to $m$ do
    For $j = 1$ to $\rho$ do
        If $(pw, sa) = (\mathbf{pw}[i], \mathbf{sa}[i,j])$ then
            Ret $(i, j)$
Ret $(\perp, \perp)$

proc. Cor$(i)$
Ret $\mathbf{pw}[i]$
```

Figure 3: An adaptive password-guessing game.

Samplers with high min-entropy. Even though the guessing advantage precisely quantifies strength of password samplers, good upper bounds in terms of the adversary's complexity and of some simpler relevant parameters of a password sampler are desirable. One interesting case is samplers with high min-entropy. Formally, we say that $\mathcal{P}$ has min-entropy μ if for all pw' it holds that $\Pr[pw = pw'] \leq 2^{-\mu}$ over the coins used in choosing $pw \leftarrow\$ \mathcal{P}$.

Theorem 3.2 Fix $m \geq q_c \geq 0$ and a password sampler $\mathcal{P}$ with min-entropy μ. Let $\mathcal{B}$ be a (q_t, q_c)-adversary for GUESS$_{\mathcal{P},m}$ making q_i queries of the form $\mathbf{Test}(i,\cdot)$ with $q_t = q_1 + \cdots + q_m$. Let $\delta = q_t/(m2^\mu)$ and let $\gamma = (m - q_c)/m$. Then $\mathbf{Adv}^{\text{guess}}_{\mathcal{P},m}(\mathcal{B}) \leq e^{-m\Delta(\gamma,\delta)}$ where $\Delta(\gamma,\delta) = \gamma\ln(\frac{\gamma}{\delta}) + (1-\gamma)\ln(\frac{1-\gamma}{1-\delta})$. ☐

Using $\Delta(\gamma,\delta) \geq 2(\gamma-\delta)^2$, we see that to win the guessing game for q_c corruptions, $q_t \approx (m - q_c) \cdot 2^\mu$ **Test** queries are necessary, and the brute-force attack is optimal. Note that the above bound is the best we expect to prove: Indeed, assume for a moment that we restrict ourselves to adversaries that want to recover a subset of $m - q_c$ passwords, without corruptions, and make q_t/m queries $\mathbf{Test}(i,\cdot)$, for each i, which are independent from queries $\mathbf{Test}(j,\cdot)$ for other $j \neq i$. Then, each individual password is found, independently, with probability at most $q_t/(m \cdot 2^\mu)$, and if one applies the Chernoff bound, the probability that a subset of size $m - q_c$ of the passwords are retrieved is upper bounded by $e^{-m\Delta(\gamma,\delta)}$. In our case, we have additional challenges: Foremost, queries for each i are not independent. Also, the number of queries may not be the same for each index i. And finally, we allow for corruption queries.

The full proof of Theorem 3.2 is given in Appendix F. At a high level, it begins by showing how to move to a simpler setting in which the adversary wins by recovering a subset of the passwords without the aid of a corrupt oracle. The resulting setting is an example of a threshold direct product game. This allows us to apply a *generalized* Chernoff bound due to Panconesi and Srinivasan [36] (see also [23]) that reduces threshold direct product games to (non-threshold) direct product games. Finally, we apply an amplification lemma due to Maurer, Pietrzak, and Renner [28] that yields a direct product theorem for the password guessing game. Let us also note that using the same technique, the better bound $\mathbf{Adv}^{\text{guess}}_{\mathcal{P},m}(\mathcal{B}) \leq (q_t/m2^\mu)^m$ can be proven for the special case of $(q_t, 0)$-adversaries.

Correlated passwords. By taking independent samples from $\mathcal{P}$ we have captured only the setting of independent passwords. In practice, of course, passwords may be correlated across users or, at least, user accounts. Our results extend to the setting of jointly selecting a vector of m passwords, except of course the analysis of the guessing advantage (whose proof fundamentally relies upon independence). This last only limits our ability to measure, in terms of simpler metrics like min-entropy, the difficulty of a guessing game against correlated passwords. This does not decrease the security proven, as the simulation-based paradigm we introduce below allows one to reduce to the full difficulty of the guessing game.

3.1 Simulation-based Security for KDFs

We define an ideal-functionality style notion of security for KDFs. Figure 4 depicts two games. A message sampler $\mathcal{M}$ is an algorithm that takes input a number r and outputs a pair of vectors $(\mathbf{pw}, \mathbf{sa})$ each

```
main $\mathrm{Real}_{\mathsf{KD},\mathcal{M},r}$
$(\mathbf{pw},\mathbf{sa}) \leftarrow\!\$\ \mathcal{M}(r)$
For $i = 1$ to $r$ do  $\mathbf{K}[i] \leftarrow\!\$\ \mathsf{KD}^H(\mathbf{pw}[i],\mathbf{sa}[i])$
$b' \leftarrow\!\$\ \mathcal{D}^{\mathbf{Prim}}(\mathbf{pw},\mathbf{sa},\mathbf{K})$
Ret $b'$

proc. Prim($X$)
Ret $H(X)$
```

```
main $\mathrm{Ideal}_{S,\mathcal{M},r}$
$(\mathbf{pw},\mathbf{sa}) \leftarrow\!\$\ \mathcal{M}(r)$
For $i = 1$ to $r$ do  $\mathbf{K}[i] \leftarrow\!\$\ \{0,1\}^k$
$b' \leftarrow\!\$\ \mathcal{D}^{\mathbf{Prim}}(\mathbf{pw},\mathbf{sa},\mathbf{K})$
Ret $b'$

proc. Prim($X$)
Ret $S^{\mathbf{Test}}(X)$

sub. Test($pw, sa$)
For $i = 1$ to $r$ do
    If $(\mathbf{pw}[i],\mathbf{sa}[i]) = (pw, sa)$ then
        Ret $\mathbf{K}[i,j]$
Ret $\perp$
```

Figure 4: Games for the simulation-based security notion for KDFs.

having r elements and with $|\mathbf{sa}[i]| = s$ for $1 \le i \le r$. A simulator S is a randomized, stateful procedure. It expects oracle access to a procedure **Test** to which it can query a message. Game $\mathrm{Real}_{\mathsf{KD},\mathcal{M},r}$ gives a distinguisher $\mathcal{D}$ the messages and associated derived keys. Also, $\mathcal{D}$ can adaptively query the ideal primitive H underlying KD. Game $\mathrm{Ideal}_{S,\mathcal{M},r}$ gives $\mathcal{D}$ the messages and keys chosen uniformly at random. Now $\mathcal{D}$ can adaptively query a primitive oracle implemented by a simulator S that, itself, has access to a **Test** oracle. Then we define KDF advantage by

$$\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{M},r}(\mathcal{D}, S) = \Pr\left[\mathrm{Real}^{\mathcal{D}}_{\mathsf{KD},\mathcal{M},r} \Rightarrow 1\right] - \Pr\left[\mathrm{Ideal}^{\mathcal{D}}_{S,\mathcal{M},r} \Rightarrow 1\right].$$

To be useful, we will require proving that there exists a simulator S such that for any $\mathcal{D},\mathcal{M}$ pair the KDF advantage is "small".

This notion is equivalent to applying the indifferentiability framework [29] to a particular ideal KDF functionality. That functionality chooses messages according to an algorithm $\mathcal{M}$ and outputs on its honest interface the messages and uniform keys associated to them. On the adversarial interface is the test routine which allows the simulator to learn keys associated to messages. This raises the question of why not just use indifferentiability from a RO as our target security notion. The reasons are two-fold. First, it is not clear that H^c is indifferentiable from a random oracle. Second, even if it were, a proof would seem to require a simulator that makes at least the same number of queries to the RO as it receives from the distinguisher. This rules out showing security amplification due to the iteration count c. Our approach solves both issues, since we will show KDF security for simulators that make one call to **Test** for every c made to it. For example, our simulator for $\mathsf{KD1}$ will only query **Test** if a chain of c hashes leads to the being-queried point X and this chain is not a continuation of some longer chain. We formally capture this property of simulators next.

c-AMPLIFYING SIMULATORS. Let $\tau = (X_1, Y_1), \ldots, (X_q, Y_q)$ be a (possibly partial) transcript of **Prim** queries and responses. We restrict attention to (k, s, c)-KDFs for which we can define a predicate $\mathsf{final}_{\mathsf{KD}}(X_i, \tau)$ which evaluates to true if there exists exactly one sequence of c indices $j_1 < \cdots < j_c$ such that (1) $j_c = i$, (2) there exist unique (pw, sa) such that evaluating $\mathsf{KD}^H(pw, sa)$ when H is such that $Y_j = H(X_j)$ for $1 \le j \le i$ results exactly in the queries $X_{j_1}, \ldots, X_{j_c}$ in any order where X_i is the last query, and (3) $\mathsf{final}_{\mathsf{KD}}(X_{j_r}, \tau) = \mathsf{false}$ for all $r < c$.

Our simulators only query **Test** on queries X_i for which $\mathsf{final}_{\mathsf{KD}}(X_i, \tau) = \mathsf{true}$; we call such queries KD-*completion queries* and simulators satisfying this are called *c-amplifying*. Note that (3) implies that there are at most q/c total KD-completion queries in a q-query transcript.

HASH-DEPENDENT PASSWORDS. We do not allow $\mathcal{M}$ access to the random oracle H. This removes from consideration hash-dependent passwords. Our results should extend to cover hash-dependent passwords if one has explicit domain separation between use of H during password selection and during key derivation. Otherwise, an indifferentiability-style approach as we use here will not work due to limitations pointed out in [39]. A full analysis of the hash-dependent password setting would therefore appear to require direct analysis of PBE schemes without taking advantage of the modularity provided by simulation-based approaches.

SECURITY OF $\mathsf{KD1}$. For a message sampler $\mathcal{M}$, let $\gamma(\mathcal{M}, r) := \Pr[\exists i \ne j : (\mathbf{pw}[i], \mathbf{sa}[i]) = (\mathbf{pw}[j], \mathbf{sa}[j])]$ where $(\mathbf{pw},\mathbf{sa}) \leftarrow\!\$\ \mathcal{M}(r)$. We prove the following theorem in Appendix G.

Theorem 3.3 Fix $r > 0$. Let $\mathsf{KD1}$ be as above. There exists a simulator S such that for all adversaries $\mathcal{D}$ making q RO queries, of which q_c are chain completion queries, and all message samplers $\mathcal{M}$,

$$\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD1},\mathcal{M},r}(\mathcal{D}, S) \le 4\,\gamma(\mathcal{P}, r) + \frac{2r^2 + 7\,(2q + rc)^2}{2^n}\ .$$

The simulator S makes at most q_c **Test** queries, and answers each query in time $O(c)$. $\square$

3.2 Security of PBE

We are now, finally, in a position to analyze the security of password based encryption as used in PKCS#5. The following theorem uses the multi-user left-or-right security notion from [4], that measures security when given access to multiple left-or-right oracles each using the same bit b. It is formalized in Appendix D. The proof of the following theorem appears in Appendix H.

Theorem 3.4 Let $m \ge 1$, let $\mathcal{SE}[\mathsf{KD}, \mathsf{SE}] = (\mathcal{P}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the encryption scheme built from an (k, s, c)-KDF KD and an encryption scheme $\mathsf{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with k-bit keys. Let $\mathcal{A}$ be an adversary making ρ queries to $\mathbf{Enc}(i, \cdot, \cdot)$ for each $i \in \{1, \ldots, m\}$ and making at most $q_c < m$ corruption queries. Let S be a c-amplifying simulator. Then there exists message sampler $\mathcal{M}$ and adversaries $\mathcal{D}$, $\mathcal{C}$, and $\mathcal{B}$ such that

$$\mathbf{Adv}^{\mathrm{lorx}}_{\mathcal{SE},m}(\mathcal{A}) \le m \cdot \mathbf{Adv}^{\mathrm{mu\text{-}lor}}_{\mathsf{SE},\rho}(\mathcal{C}) + 2 \cdot \mathbf{Adv}^{\mathrm{guess}}_{\mathcal{P},m,\rho}(\mathcal{B}) + 2 \cdot \mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{M},m\rho}(\mathcal{D}, S)$$

If $\mathcal{A}$ makes q queries to H, then: $\mathcal{D}$ makes at most q queries to its H oracle; $\mathcal{B}$ makes at most $\lceil q/c \rceil$ queries to **Test** and at most q_c corruption queries; and $\mathcal{C}$ makes a single query $\mathbf{Enc}(i, \cdot, \cdot)$ for each $1 \le i \le \rho$. Moreover, $\mathcal{C}$'s running time equals $t_{\mathcal{A}} + q \cdot t_S$ plus a small, absolute constant, and where $t_{\mathcal{A}}$ is the running time of $\mathcal{A}$, and t_S is the time needed by S to answer a query. Finally, $\gamma(\mathcal{M}, m\rho) \le m^2\rho^2/2^s$. $\square$

Note that the theorem holds even when SE is only one-time secure (meaning it can be deterministic), which implies that the analysis covers tools such as WinZip (c.f., [25]). In terms of the bound we achieve, Theorem 3.3 for $\mathsf{KD1}$ shows that an adversary that makes $\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{P}^*,m\rho}(\mathcal{D}, S)$ large requires $q \approx 2^{n/2}$ queries to H, provided salts are large. If H is SHA-256 then this is about 2^{128} work. Likewise, a good choice of SE will ensure that $\mathbf{Adv}^{\mathrm{mu\text{-}lor}}_{\mathsf{SE},\mathcal{K},\rho}(\mathcal{C})$ will be very small. Thus the dominating term ends up the guessing advantage of $\mathcal{B}$ against $\mathcal{P}$, which measures its ability to guess $m - q_c$ passwords.

References

[1] M. Abadi and B. Warinschi. Password-based encryption analyzed. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 664–676. Springer, July 2005.

[2] M. Abdalla, D. Catalano, C. Chevalier, and D. Pointcheval. Efficient two-party password-based key exchange protocols in the UC framework. In T. Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 335–351. Springer, Apr. 2008.

[3] O. Baudron, D. Pointcheval, and J. Stern. Extended notions of security for multicast public key cryptosystems. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *ICALP 2000: 27th International Colloquium on Automata, Languages and Programming*, volume 1853 of *Lecture Notes in Computer Science*, pages 499–511. Springer, July 2000.

[4] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, May 2000.

[5] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, Oct. 1997.

[6] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000.

[7] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, Nov. 1993.

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006.

[9] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

[10] X. Boyen. Halting password puzzles: hard-to-break encryption from human-memorable keys. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, page 9. USENIX Association, 2007.

[11] X. Boyen. New paradigms for password security (keynote lecture). In Y. Mu, W. Susilo, and J. Seberry, editors, *ACISP 08: 13th Australasian Conference on Information Security and Privacy*, volume 5107 of *Lecture Notes in Computer Science*, pages 1–5. Springer, July 2008.

[12] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, May 2005.

[13] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, Aug. 2005.

[14] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 494–510. Springer, Aug. 2004.

[15] Y. Dodis, R. Impagliazzo, R. Jaiswal, and V. Kabanets. Security amplification for interactive cryptographic primitives. In O. Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 128–145. Springer, Mar. 2009.

[16] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

[17] O. Goldreich, R. Impagliazzo, L. A. Levin, R. Venkatesan, and D. Zuckerman. Security preserving amplification of hardness. In *31st Annual Symposium on Foundations of Computer Science*, pages 318–326. IEEE Computer Society Press, Oct. 1990.

[18] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32. ACM Press, May 1989.

[19] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR-lemma. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 273–301. Springer, 2011.

[20] A. Groce and J. Katz. A new framework for efficient password-based authenticated key exchange. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 516–525. ACM Press, Oct. 2010.

[21] I. Haitner, D. Harnik, and O. Reingold. On the power of the randomized iterate. In C. Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 22–40. Springer, Aug. 2006.

[22] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. In *ACM CCS 98: 5th Conference on Computer and Communications Security*, pages 122–131. ACM Press, Nov. 1998.

[23] R. Impagliazzo and V. Kabanets. Constructive proofs of concentration bounds. In M. J. Serna, R. Shaltiel, K. Jansen, and J. D. P. Rolim, editors, *APPROX-RANDOM*, volume 6302 of *Lecture Notes in Computer Science*, pages 617–631. Springer, 2010.

[24] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, May 2001.

[25] T. Kohno. Attacking and repairing the winZip encryption scheme. In V. Atluri, B. Pfitzmann, and P. McDaniel, editors, *ACM CCS 04: 11th Conference on Computer and Communications Security*, pages 72–81. ACM Press, Oct. 2004.

[26] H. Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648. Springer, Aug. 2010.

[27] M. Luby and C. Rackoff. A study of password security. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 392–397. Springer, Aug. 1988.

[28] U. M. Maurer, K. Pietrzak, and R. Renner. Indistinguishability amplification. In A. Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 130–149. Springer, Aug. 2007.

[29] U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, Feb. 2004.

[30] U. M. Maurer and S. Tessaro. Computational indistinguishability amplification: Tight product theorems for system composition. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 355–373. Springer, Aug. 2009.

[31] U. M. Maurer and S. Tessaro. A hardcore lemma for computational indistinguishability: Security amplification for arbitrarily weak PRGs with optimal stretch. In D. Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 237–254. Springer, Feb. 2010.

[32] R. Morris and K. Thompson. Password security: a case history. *Commun. ACM*, 22:594–597, November 1979.

[33] S. Myers. Efficient amplification of the security of weak pseudo-random function generators. *Journal of Cryptology*, 16(1):1–24, Jan. 2003.

[34] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM CCS 05: 12th Conference on Computer and Communications Security*, pages 364–372. ACM Press, Nov. 2005.

[35] P. Oechslin. Making a faster cryptanalytic time-memory trade-off. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer, Aug. 2003.

[36] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM J. Comput.*, 26(2):350–368, 1997.

[37] K. G. Paterson and D. Stebila. One-time-password-authenticated key exchange. In R. Steinfeld and P. Hawkes, editors, *ACISP 10: 15th Australasian Conference on Information Security and Privacy*, volume 6168 of *Lecture Notes in Computer Science*, pages 264–281. Springer, July 2010.

[38] PKCS #5: Password-based cryptography standard (rfc 2898). RSA Data Security, Inc., Sept. 2000. Version 2.0.

[39] T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, May 2011.

[40] S. Tessaro. Security amplification for the cascade of arbitrarily weak PRPs: Tight bounds via the interactive hardcore lemma. In Y. Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 37–54. Springer, Mar. 2011.

[41] F. Unger. A probabilistic inequality with applications to threshold direct-product theorems. In *50th Annual Symposium on Foundations of Computer Science*, pages 221–229. IEEE Computer Society Press, Oct. 2009.

[42] D. Wagner and I. Goldberg. Proofs of security for the Unix password hashing algorithm. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 560–572. Springer, Dec. 2000.

[43] A. C. Yao. Theory and applications of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE Computer Society Press, Nov. 1982.

[44] F. F. Yao and Y. L. Yin. Design and analysis of password-based key derivation functions. In A. Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 245–261. Springer, Feb. 2005.

A Brute-force password-recovery attacks

Here we provide some more details on the (well-known) attacks on PBE mentioned in Section 1.

The si case. Recall that we encrypt a message M under a password pw by picking a random s-bit salt sa, deriving a key $L \leftarrow \mathsf{KD}(pw\|sa)$ and returning $C' \leftarrow C\|sa$ where $C \leftarrow\!\!\$\, \mathcal{E}(L, M)$. Here $\mathcal{E}$ is a symmetric encryption scheme, typically an IND-CPA AES mode of operation, and key-derivation function (KDF) KD: $\{0,1\}^* \to \{0,1\}^n$ is the c-fold iteration $\mathsf{KD} = H^c$ of a cryptographic hash function H: $\{0,1\}^* \to \{0,1\}^n$. We assume the attacker has access to an oracle **TestKey** which on input a candidate key L' returns $(L = L')$ where L is the key derived under the target password pw and target salt sa via $L \leftarrow \mathsf{KD}(pw\|sa)$. (For example the attacker may know a message M and corresponding ciphertext $C \leftarrow\!\!\$\, \mathcal{E}(L, M)$ and could test whether $\mathcal{D}(L', C) = M$ where $\mathcal{D}$ is the decryption algorithm corresponding to $\mathcal{E}$. We abstract this capability via the oracle.) The attacker is given sa. The brute-force attack now creates a table T with $\mathsf{T}[pw'] = H^c(pw'\|sa)$ for all $pw' \in D$ and then returns pw' such that **TestKey**$(\mathsf{T}[pw']) = \mathsf{true}$. The attack takes cN computations of H, where $N = |D|$ is the size of the dictionary, as well as N tests.

The mi case. Ask now how hard it is to recover a large number m of target passwords $pw_1, \ldots, pw_m$, the associated salts denoted $sa_1, \ldots, sa_m$ respectively. If $s = 0$, the answer is, not much harder than recovering one target password. The adversary's test oracle **TestKey** now takes i, L' and returns $(L_i = L')$ where $L_i = \mathsf{KD}(pw_i\|sa_i)$. The attacker creates a table T with $\mathsf{T}[pw'] = H^c(pw')$ for all $pw' \in D$ (remember the salt has length 0, meaning is absent). Then for each $pw' \in D$ and each $i = 1, \ldots, m$ it calls **TestKey**$(i, \mathsf{T}[pw'])$, returning pw' if any of these calls returns true. It recovers all the target passwords using cN hashes and mN tests. Rainbow tables and other time-memory trade-offs can be used here to save on space [35, 34].

But with s large (enough to make $sa_1, \ldots, sa_m$ usually distinct), the brute-force attack takes cmN hash computations and mN tests, for it needs a N by m array T where $\mathsf{T}[pw', sa_i] = H^c(pw'\|sa_i)$. This is why we salt. That the increase in effort from cN hashes to cmN hashes to mount the brute-force attack is considered valuable in practice is evidenced by pervasive use of salting, starting with the 1976 UNIX Password Hash and continuing into PKCS#5 [38], today's ubiquitously-employed standard for KDFs and PBE. Yet, this practice has not, until our work, received any proof-based support.

B Problems in the proofs of Yao and Yin

The earlier work of Yao and Yin [44] claimed a security proof for both functions $\mathsf{KD1}$ and $\mathsf{KD2}$ from the PKCS#5 standard for the case $m = 1$, a much more restricted scenario than the one considered in this paper (e.g., their result cannot be used to infer PBE security). Even more importantly, however, the presented proof is incorrect. In attempting to prove [44, Theorem 1], game K is not equivalent to the real world as claimed in [44, Lemma 2.2]. Namely, their proof incorrectly assumes that the $(c + 1)$ intermediate values $u_0, u_1, \ldots, u_c$ are distinct in the evaluation of $\mathsf{KD1}$ on input the randomly selected password pw and random salt sa, where $u_0 = pw \,\|\, sa$ and u_c is the derived key. The bound claimed by [44, Theorem 1] is in fact wrong since it claims to hold for arbitrary c: if c is very large (e.g., 2^n) then an attacker will beat the bound they claim.

C Proofs for Section 2

Proof of of Theorem 2.1: Let $\mathcal{A}$ be a $(t, \mathbf{q}, q_c)$-adversary for $\mathrm{UKU}_{\mathsf{SE},m}$. We use $\mathcal{A}$ to build a $(t', \mathbf{q}', q_c)$-distinguisher $\mathcal{A}'$ for $\mathrm{LORX}_{\mathsf{SE},m}$. Concretely, the distinguisher $\mathcal{A}'$ runs $\mathcal{A}$, answering its encryption queries $\mathbf{Enc}(i, M)$ with the output a query $\mathbf{Enc}(i, M, M)$ in the $\mathrm{LORX}_{\mathsf{SE},m}$ game. Any corruption query $\mathbf{Cor}(i)$ by $\mathcal{A}$ is replied with $\mathcal{A}'$ issuing the query $\mathbf{Cor}(i)$, and upon obtaining the pair $(\mathbf{K}[i], \mathbf{b}[i])$ as the answer, $\mathcal{A}'$ returns $\mathbf{K}[i]$ to $\mathcal{A}$. When $\mathcal{A}$ terminates, it outputs a vector $\mathbf{K}'$.

Subsequently, $\mathcal{A}'$ chooses k pairs of messages $M_0^{(i)}, M_1^{(i)}$ independently and uniformly at random from $\{0,1\}^\ell$, conditioned on $M_0^{(i)} \neq M_1^{(i)}$, and issues the queries $\mathbf{Enc}(i, M_0^{(j)}, M_1^{(j)})$ for all $i = 1, \ldots, m$, and $j = 1, \ldots, k$, returning the corresponding challenge ciphertexts $\mathbf{C}^{(j)}[i]$. Finally, also for all i, $\mathcal{A}$ sets $\mathbf{b}'[i] = b$ if $\mathcal{D}(\mathbf{K}'[i], \mathbf{C}^{(j)}[i]) = M_b^{(j)}$ for all $j = 1, \ldots, k$, and sets $\mathbf{b}'[i]$ to a random bit otherwise.

We now turn to analyzing the advantage of $\mathcal{A}'$: For $S \subseteq \{1, \ldots, m\}$, let bad_S be the event that $\mathbf{K}[i] \neq \mathbf{K}'[i]$ for all $i \in S$, and $\mathbf{K}[i] = \mathbf{K}'[i]$ for all $i \notin S$. In particular, $\mathsf{bad}_\emptyset$ is the event that $\mathbf{K} = \mathbf{K}'$, which in turn implies $\mathrm{P}[\mathsf{bad}_\emptyset] = \mathbf{Adv}^{\mathrm{uku}}_{\mathsf{SE},m}(\mathcal{A})$, since $\mathcal{A}'$ perfectly simulated the $\mathrm{UKU}_{\mathsf{SE},m}$ game to $\mathcal{A}$. Also, note that $\Pr[\mathrm{LORX}^{\mathcal{A}'}_{\mathsf{SE},m} \Rightarrow \mathsf{true} | \mathsf{bad}_\emptyset] = 1$.

Fix now $S \neq \emptyset$. We lower bound $\Pr[\mathrm{LORX}^{\mathcal{A}'}_{\mathsf{SE},m} \Rightarrow \mathsf{true} | \mathsf{bad}_S]$. First, note that for every fixed choice of i, $\mathbf{b}$, $M^j_{\mathbf{b}[i]}$, $\mathbf{K}$ and $\mathbf{K}'$ with $\mathbf{K}[i] \neq \mathbf{K}'[i]$, we have

$$\Pr\left[\mathcal{D}(\mathbf{K}'[i], \mathcal{E}(\mathbf{K}[i], M^j_{\mathbf{b}[i]})) = M^j_{1-\mathbf{b}[i]}\right] \leq \frac{1}{2^\ell - 1} ,$$

because M_0^j is chosen uniformly at random over a set of size $2^\ell - 1$. With bad'_S being the event that $\mathcal{D}(\mathbf{K}'[i], \mathcal{E}(\mathbf{K}[i], M^j_{\mathbf{b}[i]})) = M^j_{1-\mathbf{b}[i]}$ for all $j = 1, \ldots, m$ and all $i \in S$, by the independence of the choice of the message pairs (M_0^j, M_1^j) for all $j = 1, \ldots, k$, and by the union bound,

$$\Pr\left[\, \mathsf{bad}'_S \;\middle|\; \mathsf{bad}_S \,\right] \leq |S| \cdot \left(\frac{1}{2^\ell - 1}\right)^k \leq m \cdot \left(\frac{1}{2^\ell - 1}\right)^k .$$

Note, however, that given $\mathsf{bad}_S \wedge \overline{\mathsf{bad}'_S}$, we either have $\mathcal{D}(\mathbf{K}'[i], \mathcal{E}(\mathbf{K}[i], M^j_{\mathbf{b}[i]})) = M^j_{\mathbf{b}[i]}$ for all j and all $i \in S$, in which case $\mathbf{b}'[i] = \mathbf{b}[i]$ for all $i \in S$, or else there exists j and $i \in S$ such that $\mathcal{D}(\mathbf{K}'[i], \mathcal{E}(\mathbf{K}[i], M^j_{\mathbf{b}[i]})) \notin \{M^j_{\mathbf{b}[i]}, M^j_{1-\mathbf{b}[i]}\}$, and thus $\mathbf{b}'[i]$ is randomly chosen. In both cases, the bit b' is correct with probability at least $\frac{1}{2}$. This in particular implies

$$\begin{aligned}\Pr\left[\mathrm{LORX}^{\mathcal{A}'}_{\mathsf{SE},m} \Rightarrow \mathsf{true} \wedge \overline{\mathsf{bad}'_S} \;\middle|\; \mathsf{bad}_S\right] &\geq \Pr\left[\mathrm{LORX}^{\mathcal{A}'}_{\mathsf{SE},m} \Rightarrow \mathsf{true} \;\middle|\; \overline{\mathsf{bad}'_S} \wedge \mathsf{bad}_S\right] - \Pr\left[\mathsf{bad}'_S | \mathsf{bad}_S\right] \\ &\geq \frac{1}{2} - m \cdot \left(\frac{1}{2^\ell - 1}\right)^k .\end{aligned}$$

```
main mu-LOR$^{\mathcal{A}}_{\mathsf{SE},m}$                proc. Enc$(i, M_0, M_1)$
$\mathbf{K}[1],\ldots,\mathbf{K}[m] \leftarrow\$ \mathcal{K}$           If $|M_0| \neq |M_1|$ then Ret $\perp$
$b \leftarrow\$ \{0,1\}$                         $C \leftarrow\$ \mathcal{E}(\mathbf{K}[i], M_b)$
$b' \leftarrow\$ \mathcal{A}^{\mathbf{Enc}}$                            Ret $C$
Ret $(b' = b)$
```

Figure 5: Multi-user security notion for encryption.

We now use this to conclude

$$\begin{aligned}
\Pr\left[\mathrm{LORX}^{\mathcal{A}'}_{\mathsf{SE},m} \Rightarrow \mathsf{true}\right] &= \Pr\left[\,\mathsf{bad}_{\emptyset}\,\right] + \sum_{S\neq\emptyset} \Pr\left[\,\mathsf{bad}_S\,\right] \cdot \Pr\left[\mathrm{LORX}^{\mathcal{A}'}_{\mathsf{SE},m} \Rightarrow \mathsf{true} \mid \mathsf{bad}_S\right] \\
&\geq \Pr\left[\,\mathsf{bad}_{\emptyset}\,\right] + \sum_{S\neq\emptyset} \Pr\left[\,\mathsf{bad}_S\,\right] \cdot \Pr\left[\mathrm{LORX}^{\mathcal{A}'}_{\mathsf{SE},m} \Rightarrow \mathsf{true} \wedge \overline{\mathsf{bad}'_S} \mid \mathsf{bad}_S\right] \\
&\geq \Pr\left[\,\mathsf{bad}_{\emptyset}\,\right] + \sum_{S\neq\emptyset} \Pr\left[\,\mathsf{bad}_S\,\right] \cdot \left(\frac{1}{2} - m \cdot \left(\frac{1}{2^{\ell}-1}\right)^k\right) \\
&\geq \frac{1}{2} + \frac{1}{2} \cdot \Pr\left[\,\mathsf{bad}_{\emptyset}\,\right] - m \cdot \left(\frac{1}{2^{\ell}-1}\right)^k \\
&= \frac{1 + \mathbf{Adv}^{\mathrm{uku}}_{\mathsf{SE},m}(\mathcal{A})}{2} - m \cdot \left(\frac{1}{2^{\ell}-1}\right)^k .
\end{aligned}$$

The theorem statement follows by rearranging terms. ▌

Proof of of Theorem 2.5: Let $\mathcal{A}$ be a AND adversary. Consider the vector $\mathbf{b}'$ output by $\mathcal{A}$. For all subsets $S \subseteq \{1,\ldots,m\}$, and with $\mathbf{b}$ being the vector chosen in AND, it is not hard to verify that

$$\Pr\left[\bigoplus_{i\in S} \mathbf{b}'[i] = \bigoplus_{i\in S} \mathbf{b}[i]\right] \leq \frac{1 + \mathbf{Adv}^{\mathrm{lorx}}_{\mathsf{SE},|S|}(t^*_S, \mathbf{q}', q_c)}{2} \leq \frac{1 + C\epsilon^{|S|} + \gamma}{2} ,$$

where $\mathbf{q}'$ is the $|S|$-dimensional vector obtained as the projection of $\mathbf{q}$ on components in S. Indeed, for every adversary $\mathcal{A}$ and $S \subseteq \{1,\ldots,m\}$, where $S = \{i_1,\ldots,i_k\}$,, we can build an adversary $\mathcal{B}$ for $\mathrm{LORX}_{\mathsf{SE},|S|}$ as follows: It first chooses random bits $\mathbf{b}[i] \leftarrow\$ \{0,1\}$ and $\mathbf{K}[i] \leftarrow\$ \mathcal{K}$ for $i \notin S$, and then runs $\mathcal{A}$. Whenever $\mathcal{A}$ queries $\mathbf{Enc}(i_j, M_0, M_1)$ for $j = 1,\ldots,k$, $\mathcal{B}$ uses the oracle $\mathbf{Enc}(j, M_0, M_1)$ from the underlying $\mathrm{LORX}_{m,\mathsf{SE}}$ game to answer the query, whereas queries $\mathbf{Enc}(i, M_0, M_1)$ for $i \notin S$ are replied with $\mathcal{E}(\mathbf{K}[i], M_{\mathbf{b}[i]})$. Similarly, corruption queries for $i \in S$ are answered using the corresponding **Cor** oracle for the LORX game, whereas queries for $i \notin S$ are answered directly returning $\mathbf{b}[i]$ and $\mathbf{K}[i]$. At the end of the game, if $\mathcal{A}$ replies with $\mathbf{b}'$, $\mathcal{B}$ outputs $\bigoplus_i \mathbf{b}'[i]$. It is clear by inspection that $\mathcal{B}$ has running time at most t^*_S. To conclude, we apply Lemma 2.4 with $Y_i = \mathbf{b}[i] \oplus \mathbf{b}'[i]$ to obtain an upper bound on $\Pr\left[\sum_{i=1}^m Y_i = m\right] = \Pr\left[\mathbf{b} = \mathbf{b}'\right]$. ▌

D Multi-user Encryption Security

We recall the multi-user security notion from [4]. Game mu-$\mathrm{LOR}_{\mathsf{SE},m}$ of Figure 5 defines the security experiment. The advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}^{\text{mu-lor}}_{\mathsf{SE},m}(\mathcal{A}) = 2\Pr[\mathrm{LORX}^{\mathcal{A}}_{\mathsf{SE},m} \Rightarrow \mathsf{true}] - 1$. We say that $\mathcal{A}$ is a $(t,\mathbf{q})$-adversary if it runs in time t and makes at most $\mathbf{q}[i]$ encryption queries of the form $\mathbf{Enc}(i,\cdot,\cdot)$. Then we let $\mathbf{Adv}^{\text{mu-lor}}_{\mathsf{SE},m}(t,\mathbf{q}) = \max_{\mathcal{A}} \mathbf{Adv}^{\text{mu-lor}}_{\mathsf{SE},m}(\mathcal{A})$ where the maximum is over all $(t,\mathbf{q})$-adversaries.

E More KDFs

The second KDF from PKCS#5 uses a function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ with two designated inputs. Note that in the standard F is referred to as a PRF, but since what is needed is not a PRF in the traditional sense, we refer to it as just a (keyed) function. Then define

$$\mathsf{KD2}^F(pw, sa) = U_1 \oplus U_2 \oplus \cdots \oplus U_c$$

where $U_i = F(pw, U_{i-1})$ for all $i = 1, \ldots, c$, and $U_0 = sa$. We sketch an analysis of this KDF in Appendix G.

For both KDFs we have for simplicity deviated from the standard in that we assume the output length of the hash is equal to the desired key length. Achieving shorter key lengths with $\mathsf{KD1}$ and $\mathsf{KD2}$ just requires truncation, while for $\mathsf{KD2}$ one can also request longer derived keys. This is accomplished by repeated applications of $\mathsf{KD2}$ using domain separation.

F Proof of Theorem 3.2

The proof follows directly by combining two lemmas that we now state and prove. The first shows that a threshold variant of the guessing game implies security of the version given in the body that uses corruptions. The second bounds the advantage against this threshold guessing game using a generalized Chernoff bound due to Panconesi and Srinivasan [36] that reduces threshold direct products theorems to (non-threshold) direct product theorems. Finally, we use an amplification lemma due to Maurer, Pietrzak, and Renner [28] that yields a direct product theorem for the password guessing game (without corruptions).

Corruptions and threshold are equivalent. We define a threshold security variant of the guessing game. Let τ-GUESS$_{\mathcal{P},m}$ be the same as GUESS$_{\mathcal{P},m}$ except: (1) the corruption oracle is removed; (2) the winning condition is now $\bigvee_{S:|S|\geq\tau} \bigwedge_{i\in S}(\mathbf{pw}'[i] = \mathbf{pw}[i])$, i.e., the adversary wins if a subset of size at least τ of the passwords in the output is correct. The following lemma shows that threshold security for $t = m - q_c$ implies security when given q_c corrupt queries.

Lemma F.1 Let $m \geq 0$, q_t, q_c be numbers, and $\tau = m - q_c$. Let $\mathcal{A}$ be a (q_t, q_c)-guessing GUESS$_{\mathcal{P},m}$ adversary. Then there exists a τ-GUESS$_{\mathcal{P},m}$ adversary $\mathcal{B}$ making q_t **Test** queries such that

$$\mathbf{Adv}^{\text{guess}}_{\mathcal{P},m}(\mathcal{A}) \leq \mathbf{Adv}^{\tau\text{-guess}}_{\mathcal{P},m}(\mathcal{B}) \ . \ \square$$

Proof: (Sketch.) The adversary $\mathcal{B}$ works as follows on input $\mathbf{sa}$. It runs $\mathcal{A}^{\mathbf{TestSim},\mathbf{CorSim}}(\mathbf{sa})$ simulating queries as follows. For a **TestSim**(i, pw) query it queries its own test oracle **Test**(i, pw). If the response is $\bot$, it adds pw to a set $\mathcal{T}_i$ and returns $\bot$ to $\mathcal{A}$. Otherwise it returns true to $\mathcal{A}$. For a **CorSim**(i) query, it samples a fresh password $\mathbf{pw}^*[i]$ according to the distribution of $\mathcal{P}$ conditioned on not outputting a password in the set $\mathcal{T}_i$. Finally $\mathbf{pw}^*[i]$ is returned to $\mathcal{A}$.

Note that to be able to sample the above distribution, $\mathcal{B}$ keeps, for each $1 \leq i \leq m$, an array $p_i[\cdot]$ where $p_i[pw]$ initially indicates the probability that $\mathcal{P}$ samples pw for each $pw \in P$, where P is the set of possible passwords output by $\mathcal{P}$. (Recall that $\mathcal{B}$ is not computationally bounded, and in fact we can even allow $\mathcal{B}$ to store arbitrary precision numbers.) When pw is added to $\mathcal{T}_i$, $\mathcal{B}$ sets $p_i[pw]$ to 0, computes $p^* = \sum_{pw'\in P} p_i[pw']$, and sets $p_i[pw'] \leftarrow p_i[pw']/p^*$ for all $pw' \in P$. Assume now we order the elements of P arbitrarily as $pw_1, pw_2, ..., pw_{|P|}$. Then, to sample $\mathbf{pw}^*[i]$, $\mathcal{B}$ picks a random number $x \leftarrow_{\$} [0,1]$, and outputs pw_k for the smallest k such that $\sum_{j\leq k-1} p_i[pw_j] \leq x$, and $\sum_{j\leq k} p_i[pw_j] > x$.

By construction, the probability that any pw is returned by $\mathcal{B}$ in response to a corruption query equals the probability that any pw would be returned in response to such a query in GUESS$_{\mathcal{P},m}$. (This also uses that $\mathcal{A}$ make no pointless queries.)

When $\mathcal{A}$ outputs $\mathbf{pw}'$, adversary $\mathcal{B}$ outputs the $\mathbf{pw}'$. As passwords for uncorrupted indices are going to be equal with probability at least $\Pr[\text{GUESS}^{\mathcal{A}}_{\mathcal{P},m}]$, we have $\Pr[\tau\text{-GUESS}^{\mathcal{B}}_{\mathcal{P},m}] \geq \Pr[\text{GUESS}^{\mathcal{A}}_{\mathcal{P},m}]$, which concludes the proof. ▌

A Chernoff lemma. We now turn to the second lemma.

Lemma F.2 Fix $m \geq \tau \geq 0$ and a password sampler $\mathcal{P}$ with min-entropy μ. Let $\mathcal{B}$ be an $\tau\text{-GUESS}_{\mathcal{P},m}$ adversary making q_i queries of the form $\mathbf{Test}(i,\cdot)$ with $q = q_1 + \cdots + q_m$ and $q \leq \tau 2^{\mu}$. Let $\delta = q/(m2^{\mu})$ and let $\gamma = \tau/m$. Then

$$\Pr\left[\, \tau\text{-GUESS}^{\mathcal{B}}_{\mathcal{P},m} \Rightarrow \mathsf{true} \,\right] \leq e^{-m\Delta(\gamma,\delta)}$$

where $\Delta(\gamma,\delta) = \gamma \ln(\frac{\gamma}{\delta}) + (1-\gamma)\ln(\frac{1-\gamma}{1-\delta})$. $\square$

Proof: Consider the $\tau\text{-GUESS}_{\mathcal{P},\tau}$ adversary $\mathcal{B}'$ making $q'_1, \ldots, q'_\tau$ queries of the form $\mathbf{Test}(i,\cdot)$. The the challenge passwords are independent, meaning that the challenge oracles $\mathbf{Test}(i,\cdot)$ and $\mathbf{Test}(j,\cdot)$ are independent for $i \neq j$. We can therefore apply[5] the amplification lemma due to Maurer, Pietrzak, and Renner [28, Lem. 6] to show that for any $\tau \in [1 \,..\, m]$,

$$\Pr\left[\tau\text{-GUESS}^{\mathcal{B}'}_{\mathcal{P},\tau} \Rightarrow \mathsf{true}\right] \leq \prod_{i=1}^{\tau} \frac{q'_i}{2^{\mu}} \,. \tag{1}$$

Let $\delta_i = q_i/2^{\mu}$. We can now show an upper bound on $\mathcal{B}$'s success in terms of the δ_i values using a generalized Chernoff bound due originally to Panconesi and Srinivasan [36], and recently treated Impagliazzo and Kabanets [23]. We restate the latter's formulation below for reference:

Lemma F.3 Let $X_1, \ldots, X_m$ be binary random variables. Suppose that there are $0 \leq \delta_i \leq 1$ for $1 \leq i \leq m$, such that, for every set $S \subseteq [1 \,..\, m]$, $\Pr[\wedge_{i \in S} X_i = 1] \leq \prod_{i \in S} \delta_i$. Let $\delta = (1/m)\sum_{i=1}^{m} \delta_i$. Then, for any γ such that $\delta \leq \gamma \leq 1$, $\Pr[\sum_{i=1}^{m} X_i \geq \gamma m] \leq e^{-m\Delta(\gamma,\delta)}$. $\square$

To apply the lemma, we let: (1) X_i be the binary random variable in In $\tau\text{-GUESS}^{\mathcal{B}}_{\mathcal{P},m}$ which is 1 if $\mathcal{B}$ outputs $\mathbf{pw}'$ with $\mathbf{pw}'[i] = \mathbf{pw}[i]$ (it guessed the i^{th} password), and 0 otherwise; and (2) $\gamma = \tau/m$. Then, for all $S \subseteq [1 \,..\, m]$, we do have

$$\Pr[\wedge_{i \in S} X_i = 1] \leq \prod_{i \in S} \delta_i \,,$$

as otherwise we can build an adversary $\mathcal{B}'$ for $\tau\text{-GUESS}_{\mathcal{P},\tau}$ with $\tau = |S|$ contradicting (1). Namely, the adversary $\mathcal{B}'$ simply runs $\mathcal{B}$, and uses the $\mathbf{Test}$ oracle in $\tau\text{-GUESS}_{\mathcal{P},\tau}$ to simulate calls to $\mathbf{Test}(i,\cdot)$ for $i \in S$, whereas it simulates internally by itself passwords $pw[j]$ and oracle calls $\mathbf{Test}(j,\cdot)$, for $j \notin S$.

Therefore, using

$$\Pr\left[\sum_{i=1}^{m} X_i \geq \gamma m\right] \geq \Pr\left[\, \tau\text{-GUESS}^{\mathcal{B}}_{\mathcal{P},m} \Rightarrow \mathsf{true} \,\right]$$

and also

$$\delta = \frac{1}{m}\sum_{i=1}^{m} \delta_i = \frac{1}{m}\sum_{i=1}^{m} \frac{q_i}{2^{\mu}} = \frac{q}{m2^{\mu}} \,.$$

So $\delta < \gamma = \tau/m$ holds as long as $q \leq \tau 2^{\mu}$, matching the requirement in the statement (of Lemma F.2). ▌

[5]This is because, in the language of [28], we can see our setting as the parallel composition of m random systems with a monotone binary output (MBO) which becomes one when the adversary outputs the corresponding password.

```
proc. $S^{\mathbf{Test}}(x)$
If $H[x] = \perp$ then
    $w_0 \leftarrow \mathrm{FindChain}(x)$
    If $w_0 \neq \perp$ then
        $(pw, sa) \leftarrow \mathsf{Decode}(w_0)$
        $K' \leftarrow \mathbf{Test}(pw, sa)$
        If $K' \neq \perp$ then
            $z \leftarrow\!\!\$\ \{0,1\}^{n-k}$ ; $H[x] \leftarrow K' \,\|\, z$
    If $H[x] = \perp$ then $H[x] \leftarrow\!\!\$\ \{0,1\}^n$
    $H^{-1}[H[x]] \xleftarrow{\cup} \{x\}$
Ret $H[x]$

sub. FindChain($w$)
$w_{c-1} \leftarrow w$
For $i := 1$ to $c-1$ do
    If $|H^{-1}[w_{c-i}]| \neq 1$ then Ret $\perp$
    $w_{c-i-1} \leftarrow H^{-1}[w_{c-i}]$
If $H^{-1}[w_0] = \perp$ then Ret $w_0$
Ret $\perp$
```

Figure 6: The simulator $\mathcal{S}$ for Theorem 3.3.

G Security Analysis of KD1

Analysis of KD1. For a hash function $H : \{0,1\}^* \to \{0,1\}^n$ (which we model as a random oracle), let KD1 be the (k, s, c)-KDF as described in Section 3 with $k \leq n$. (In the following, we assume for simplicity that $k = n$, the analysis for shorter output lengths follows straightforwardly.) Note that a query x is a chain-completion query if there exists previous queries $w_0, w_1, \ldots, w_{c-1} = x$ such that $H(w_i) = w_{i+1}$ for all $i = 0, \ldots, c-2$, $w_0 = \mathsf{Encode}(pw, sa)$ for some pw, sa, and no earlier query returned w_0.

That Encode is 1-1 implies specifically that $\gamma(\mathcal{M}, r)$ also describes the probability that the encodings of two triples collide. Note that we make no further assumptions about Encode. In particular, it can be that it outputs values that are n bit strings. This ends up a key challenge in the proof, as an adversary can possibly produce very long chains. For example, a chain of hashes of length ℓ with $c < \ell < 2c$ that contains subchains corresponding to $H^c(\mathsf{Encode}(pw, sa))$ and $H^c(\mathsf{Encode}(pw', sa))$ where $pw \neq pw'$. The core intuition underlying our proof is that one can show that this strategy does not help, and in fact, when creating a longer chain, only the first subchain will be relevant (thus justifying our notion of chain completion). Unfortunately, as we will see, making this intuition into a rigorous argument takes some work.

Proof of Theorem 3.3: We first provide a description of the simulator S for KD1 with chain length c, key length k, and with salts of lengths s. A pseudocode description of S is given in Figure 6. At a high level, it simulates the RO H via lazy sampling of its function table $H[\cdot]$, given only access to the **Test** oracle. At any point in time, we denote as $H^{-1}[y]$ the set of preimages x for which $H[x] = y$. The simulator attempts to ensure that evaluating KD1 with the simulated RO on inputs associated to the vectors $\mathbf{pw}, \mathbf{sa}$ yields consistent answers with the key vector $\mathbf{K}$.

To do so, upon a query x, S looks for a *chain* of earlier queries $w_0, w_1, \ldots, w_{c-1}$ such that: (1) $w_{c-1} = x$ and $w_0 = \mathsf{Encode}(pw, sa)$ for some pw and sa; (2) $H[w_{i-1}] = w_i$ and $|H^{-1}[w_i]| = 1$ for all $i = 1, \ldots, c-1$, as well as $H^{-1}[w_0] = \emptyset$. If such a chain exists, the simulator queries $\mathbf{Test}(pw, sa)$. If the returned value K' is $\perp$, then $H[w_{c-1}]$ is set to a fresh random value, whereas if $K' \neq \perp$, it lets $H[w_{c-1}] \leftarrow K' \,\|\, z$ for $z \leftarrow\!\!\$\ \{0,1\}^{n-k}$. If no chain exists, $H[x]$ is set to a fresh random n-bit string. By inspection, the simulator makes a **Test** query only as a result of a chain completion query as defined above. (In fact, some chain completion queries do not result in a **Test** query.) From now on, we assume without loss of generality that $k = n$, since giving $\mathcal{D}$ more of the hash output can make its task no harder.

The proof considers a sequence of games. These games are given in full detail in the appendix due to space constraints. The first game G_0 (cf. Figure 7) is the game $\mathrm{Real}_{\mathsf{KD1},\mathcal{M},r}$ with some syntactical modifications: The r keys $\mathbf{K}[1], \ldots, \mathbf{K}[r]$ are initially chosen uniformly at random. The KDF KD1 is then evaluated on $(\mathbf{pw}[i], \mathbf{sa}[i])$ for all i, generating intermediate n-bit values $\mathbf{W}_j[i]$ for $j = 0, \ldots, c-1$ (which we collectively refer to as the $\mathbf{W}$ *values*). Also, if $H[\mathbf{W}_{c-1}[i]]$ is defined, we overwrite $\mathbf{K}[i]$ with $H[\mathbf{W}_{c-1}[i]]$. This just corresponds to the situation that a collision occurs between two outputs of KD1; we set a flag bad if this occurs. We additionally set bad if, during this initialization, any collision occurs

```
proc. main // $\boxed{G_0}, G_1$
$\mathbf{K} \leftarrow_\$ \{0,1\}^{n \cdot r}$
$(\mathbf{pw}, \mathbf{sa}) \leftarrow_\$ \mathcal{M}(r)$
For $i = 1$ to $r$ do
    $\mathbf{W}_0[i] \leftarrow \mathsf{Encode}(\mathbf{pw}[i], \mathbf{sa}[i])$
    For $j = 1$ to $c - 1$ do
        If $H[\mathbf{W}_{j-1}[i]] = \bot$ then
            $H[\mathbf{W}_{j-1}[i]] \leftarrow_\$ \{0,1\}^n$
        $\mathbf{W}_j[i] \leftarrow H[\mathbf{W}_{j-1}[i]]$
    If $H[\mathbf{W}_{c-1}[i]]$ then
        $\mathsf{bad} \leftarrow \mathsf{true}$
        $\boxed{\mathbf{K}[i] \leftarrow H[\mathbf{W}_{c-1}[i]]}$
If $\exists (i,j) \neq (i',j') : \mathbf{W}_j[i] = \mathbf{W}_{j'}[i']$
    then $\mathsf{bad} \leftarrow \mathsf{true}$
$b' \leftarrow \mathcal{D}^{\mathbf{H}}(\mathbf{pw}, \mathbf{sa}, \mathbf{K})$
Ret $b'$
```

```
proc. H(x) // $G_0, G_1$
If $H[x] = \bot$ then
    $H[x] \leftarrow_\$ \{0,1\}^k$
    For $i = r$ down to 1 do
        If $x = \mathbf{W}_{c-1}[i]$ then
            $H[x] \leftarrow \mathbf{K}[i]$
Ret $H[x]$
```

```
proc. H(x) // $B_5$
$X \xleftarrow{\cup} \{x\}$
If $H[x] = \bot$ then
    If $H'[x] = \bot$ then $H'[x] \leftarrow \{0,1\}^n$
    $H[x] \leftarrow H'[x]$
    For $i = r$ down to 1 do
        If $\mathrm{FindChain}(x) = \mathbf{W}_0[i]$ then
            $H[x] \leftarrow \mathbf{K}[i]$ ; $S_i \xleftarrow{\cup} \{x\}$
        Else If $x = \mathbf{W}_{c-1}[i]$ then
            $\mathsf{bad}' \leftarrow \mathsf{true}$
$H^{-1}[H[x]] \xleftarrow{\cup} \{x\}$
Ret $H[x]$
```

```
proc. main // $G_2, G_3, G_4, B_5$
$\mathbf{K} \leftarrow_\$ \{0,1\}^{n \cdot r}$
$(\mathbf{pw}, \mathbf{sa}) \leftarrow_\$ \mathcal{M}(r)$
For $i = 1$ to $r$ do
    $\mathbf{W}_0[i] \leftarrow \mathsf{Encode}(\mathbf{pw}[i], \mathbf{sa}[i])$
    For $j = 1, \ldots, c - 1$ do
        If $H'[\mathbf{W}_{j-1}[i]] = \bot$ then
            $H'[\mathbf{W}_{j-1}[i]] \leftarrow_\$ \{0,1\}^n$
        $\mathbf{W}_j[i] \leftarrow H'[\mathbf{W}_{j-1}[i]]$
If $\exists (i,j) \neq (i',j') : \mathbf{W}_j[i] = \mathbf{W}_{j'}[i']$
    then $\mathsf{bad} \leftarrow \mathsf{true}$
$b' \leftarrow \mathcal{D}^{\mathbf{H}}(\mathbf{pw}, \mathbf{sa}, \mathbf{K})$
Ret $b'$
```

```
proc. H(x) // $\boxed{G_2}, G_3$
If $H[x] = \bot$ then
    If $H'[x] = \bot$ then $H'[x] \leftarrow_\$ \{0,1\}^n$
    $H[x] \leftarrow H'[x]$
    For $i = m$ down to 1 do
        If $x = \mathbf{W}_{c-1}[i]$ then
            If $\mathrm{FindChain}(x) = \mathbf{W}_0[i]$ then
                $H[x] \leftarrow \mathbf{K}[i]$
            Else
                $\mathsf{bad}' \leftarrow \mathsf{true}$ ; $\boxed{H[x] \leftarrow \mathbf{K}[i]}$
    $H^{-1}[H[x]] \xleftarrow{\cup} \{x\}$
Ret $H[x]$
```

```
sub. FindChain(w)
$w_{c-1} \leftarrow w$
For $i := 1$ to $c - 1$ do
    If $|H^{-1}[w_{c-i}]| \neq 1$ then Ret $\bot$
    $w_{c-i-1} \leftarrow H^{-1}[w_{c-i}]$
If $H^{-1}[w_0] = \bot$ then Ret $w_0$
Ret $\bot$
```

```
proc. main // $G_5$
$\mathbf{K} \leftarrow_\$ \{0,1\}^{n \cdot r}$
$(\mathbf{pw}, \mathbf{sa}) \leftarrow_\$ \mathcal{M}(r)$
For $i = 1$ to $r$ do
    $\mathbf{W}_0[i] \leftarrow \mathsf{Encode}(\mathbf{pw}[i], \mathbf{sa}[i])$
$b' \leftarrow \mathcal{D}^{\mathbf{H}}(\mathbf{pw}, \mathbf{sa}, \mathbf{K})$
Ret $b'$
```

```
proc. H(x) // $G_4, G_5$
If $H[x] = \bot$ then
    If $H'[x] = \bot$ then $H'[x] \leftarrow_\$ \{0,1\}^n$
    $H[x] \leftarrow H'[x]$
    For $i = m$ down to 1 do
        If $\mathrm{FindChain}(x) = \mathbf{W}_0[i]$ then
            $H[x] \leftarrow \mathbf{K}[i]$
    $H^{-1}[H[x]] \xleftarrow{\cup} \{x\}$
Ret $H[x]$
```

```
proc. main // $G_6$
$\mathbf{K} \leftarrow_\$ \{0,1\}^{n \cdot r}$
$(\mathbf{pw}, \mathbf{sa}) \leftarrow_\$ \mathcal{M}(r)$
$b' \leftarrow \mathcal{D}^{\mathbf{H}}(\mathbf{pw}, \mathbf{sa}, \mathbf{K})$
Ret $b'$
```

```
proc. H(x) // $G_6$
If $H[x] = \bot$ then
    $H[x] \leftarrow_\$ \{0,1\}^n$
    $w_0 \leftarrow \mathrm{FindChain}(x)$
    If $w_0 \neq \bot$ then
        $(pw, sa) \leftarrow \mathsf{Decode}(w_0)$
        $K' \leftarrow \mathbf{Test}(pw, sa)$
        If $K' \neq \bot$ then $H[x] \leftarrow K'$
    $H^{-1}[H[x]] \xleftarrow{\cup} \{x\}$
Ret $H[x]$
```

```
proc. main // $B_6$
$(\mathbf{pw}, \mathbf{sa}) \leftarrow_\$ \mathcal{M}(r)$ ; $\mathbf{K} \leftarrow_\$ \{0,1\}^{nr}$
For $i = 1$ to $r$ do
    $\mathbf{W}_0[i] \leftarrow \mathsf{Encode}(\mathbf{pw}[i], \mathbf{sa}[i])$ ; $W \xleftarrow{\cup} \{\mathbf{W}_0[i]\}$ ; $K \xleftarrow{\cup} \{\mathbf{K}[i]\}$
If $|W \cup K| < 2r$ then $\mathsf{bad}''' \leftarrow \mathsf{true}$
$b' \leftarrow \mathcal{D}^{\mathbf{H}}(\mathbf{pw}, \mathbf{sa}, \mathbf{K})$
For $i = 1$ to $r$ do
    For $j = 1$ to $c - 1$ do
        If $H'[\mathbf{W}_j[i]] = \bot$ then
            $X' \xleftarrow{\cup} \{\mathbf{W}_j[i]\}$ ; $H'[\mathbf{W}_j[i]] \leftarrow_\$ \{0,1\}^n$
            If $H'[\mathbf{W}_j[i]] \in X' \cup Y' \cup K \cup W$ then $\mathsf{bad}''' \leftarrow \mathsf{true}$
            $Y' \xleftarrow{\cup} \{H'[\mathbf{W}_j[i]]\}$
        $\mathbf{W}_{j+1}[i] \leftarrow H'[\mathbf{W}_j[i]]$
    If $\mathbf{W}_{c-1}[i] \in X \setminus S_i$ then $\mathsf{bad}'' \leftarrow \mathsf{true}$
Ret $b'$
```

```
proc. H(x) // $B_6$
$X \xleftarrow{\cup} \{x\}$ ; $X' \xleftarrow{\cup} \{x\}$
If $H[x] = \bot$ then
    If $H'[x] = \bot$ then $H'[x] \leftarrow_\$ \{0,1\}^n$
    $H[x] \leftarrow H'[x]$
    If $H'[x] \in X \cup Y \cup K \cup W$ then
        $\mathsf{bad}''' \leftarrow \mathsf{true}$
    $Y' \xleftarrow{\cup} \{H'[x]\}$
    For $i = r$ down to 1 do
        If $\mathrm{FindChain}(x) = \mathbf{W}_0[i]$ then
            $H[x] \leftarrow \mathbf{K}[i]$ ; $S_i \xleftarrow{\cup} \{x\}$
    $H^{-1}[H[x]] \xleftarrow{\cup} \{x\}$
Ret $H[x]$
```

Figure 7: Games for the proof of Theorem 3.3.

among the $\mathbf{W}$ values. (If bad is set in the earlier place, then it is necessarily set again here.) Procedure $\mathbf{H}$ implements a random oracle consistently with the table H. It also ensures that queries $x = \mathbf{W}_{c-1}[i]$ are answered with $\mathbf{K}[i]$ for all $i = 1, \ldots, r$.

We transform G_0 into a new game G_1 (boxed statement omitted). The sole difference is that no $\mathbf{K}$ value is overwritten at initialization. Clearly, until bad occurs, games G_0 and G_1 are equivalent and so

$$\Pr\left[\, G_0^{\mathcal{D}} \Rightarrow 1 \,\right] - \Pr\left[\, G_1^{\mathcal{D}} \Rightarrow 1 \,\right] \leq \Pr\left[\, G_0^{\mathcal{D}} \text{ sets } \mathsf{bad} \,\right] .$$

To upper bound the latter probability, for notational simplicity, we denote as $W_1, W_2, \ldots, W_{r(c-1)}$ the sequence of $\mathbf{W}_j[i]$ values for $j > 0$ in the order they are generated. Also, let $\widetilde{W}_i := \mathbf{W}_0[i]$ for $i = 1, \ldots, r$. For $i = 1, \ldots, r(c-1)$, define $\mathsf{COLL}(i)$ as the event that there is a collision among the $r + i$ n-bit values $\{\widetilde{W}_1, \ldots, \widetilde{W}_r, W_1, \ldots, W_i\}$. Then,

$$\Pr\left[\, G_0^{\mathcal{D}} \text{ sets } \mathsf{bad} \,\right] = \Pr\left[\bigvee_{i=0}^{r(c-1)} \mathsf{COLL}(i)\right] \leq \gamma(\mathcal{M}, r) + \sum_{i=1}^{r(c-1)} \frac{i + r}{2^n} \leq \gamma(\mathcal{M}, r) + \frac{(r \cdot c)^2}{2^n} .$$

where for the second term we have used that W_i is always set uniformly conditioned on $\mathsf{COLL}(i-1)$ not having occurred.

We modify G_1 into a further game G_2. First, we introduce a new table H' that tracks the choices of $\mathbf{W}$ values in **main** (as was previously done by H). Procedure $\mathbf{H}$ is modified to ensure consistency with H' (the second and third lines of code). The table H' will be useful later. Second, we remove the (now extraneous) if statement from **main**. Third, in $\mathbf{H}$ we give FindChain a chance at recovering $\mathbf{W}_0[i]$. A flag bad' is set if it fails to do so, but the boxed statement (included in G_2) ensures that the same value is returned as would have been in G_1. We introduce the table H^{-1} because FindChain requires it. The above changes are conservative and so $\Pr[G_1^{\mathcal{D}} \Rightarrow 1] = \Pr[G_2^{\mathcal{D}} \Rightarrow 1]$. Game G_3 drops the boxed statement in G_2 and so

$$\begin{aligned}\Pr\left[\, G_2^{\mathcal{D}} \Rightarrow 1 \,\right] - \Pr\left[\, G_3^{\mathcal{D}} \Rightarrow 1 \,\right] &\leq \Pr\left[\, G_2^{\mathcal{D}} \text{ sets } \mathsf{bad}' \,\right] \\ &= \Pr\left[\, G_3^{\mathcal{D}} \text{ sets } \mathsf{bad}' \,\right] .\end{aligned}$$

We will upper bound the probability $\Pr\left[\, G_2^{\mathcal{D}} \text{ sets } \mathsf{bad}' \,\right]$ below, but for now continue presenting the main sequence of games.

Game G_4 simplifies $\mathbf{H}$ by omitting the check that $x = \mathbf{W}_{c-1}[i]$. We now argue that

$$\begin{aligned}\Pr\left[\, G_3^{\mathcal{D}} \Rightarrow 1 \,\right] - \Pr\left[\, G_4^{\mathcal{D}} \Rightarrow 1 \,\right] &\leq \Pr\left[\, G_3^{\mathcal{D}} \text{ sets } \mathsf{bad} \,\right] \\ &\leq \gamma(\mathcal{M}, r) + \frac{(r \cdot c)^2}{2^n}\end{aligned}$$

where the last inequality comes from the same analysis as above. To argue the first inequality, we note that in game G_3, as long as bad is not set, for x such that $H[x] = \perp$, it cannot be that FindChain(x) returns $\mathbf{W}_0[i]$ while $x \neq \mathbf{W}_{c-1}[i]$. To see this, assume that this is not true and consider the first point in time where a fresh $\mathbf{H}$ query $x \neq \mathbf{W}_{c-1}[i]$ is made, but FindChain(x) returns $\mathbf{W}_0[i]$. Then, there is a sequence of values $w_0, w_1, \ldots, w_{c-1}$ such that $w_0 = \mathbf{W}_0[i]$, $w_j = H[w_{j-1}]$ for a $j = 1, \ldots, c-1$, and $w_{c-1} = x \neq \mathbf{W}_{c-1}[i]$. This means in particular that there exists $j \in \{0, \ldots, c-2\}$ with $w_j = \mathbf{W}_j[i]$, but $H[w_j] \neq H'[w_j]$. But this can only have happened if $\mathbf{W}_{j-1}[i] = \mathbf{W}_{c-1}[i']$ for some i', which implies bad.

We now observe that G_4's $\mathbf{H}$ code only needs to know the values $\mathbf{W}_0[i]$'s from the initialization stage, whereas all other $\mathbf{W}$ values (and corresponding entries in $H'[\cdot]$) can be generated (implicitly) on the fly when answering an $\mathbf{H}$ query. We can thus simplify the initialization procedure, obtaining game G_5.

We undertake a final step leading us to game G_6, where we introduce use the procedure **Test** as in the ideal game $\mathrm{Ideal}_{S,\mathcal{M},r}$: On input (pw, sa), it returns $\mathbf{K}[i]$ for the smallest i such that $\mathbf{pw}[i] = pw$ and $\mathbf{sa}[i] = sa$, and $\perp$ if no such key exists. We then modify the procedure $\mathbf{H}$ to answer queries using **Test** without knowing $\mathbf{W}_0$ and $\mathbf{K}$. It is clear that

$$\Pr\left[\, G_5^{\mathcal{D}} \Rightarrow 1 \,\right] = \Pr\left[\, G_6^{\mathcal{D}} \Rightarrow 1 \,\right] = \Pr\left[\, \mathrm{Ideal}_{S,\mathcal{M},r}^{\mathcal{D}} \Rightarrow 1 \,\right] .$$

We now return to upper bounding the probability of bad' occurring in G_2 and G_3. We proceed by introducing a new game (B_5) which is similar to G_4, with the exception that it additionally sets bad' as in G_3. A similar reasoning as above tells us that as long as bad does not occur, G_3 and B_5 are identical, in particular, also with respect to the setting of bad'. Therefore,

$$\begin{aligned} \Pr\left[\, G_3^{\mathcal{D}} \text{ sets } \mathsf{bad}' \,\right] &\leq \Pr\left[\, B_5^{\mathcal{D}} \text{ sets } \mathsf{bad}' \,\right] + \Pr\left[\, G_3^{\mathcal{D}} \text{ sets } \mathsf{bad} \,\right] \\ &\leq \Pr\left[\, B_5^{\mathcal{D}} \text{ sets } \mathsf{bad}' \,\right] + \gamma(\mathcal{M}, r) + \frac{(r \cdot c)^2}{2^n} . \end{aligned}$$

The game B_5 also keeps track of $\mathbf{H}$ queries (in the set X, initially empty), as well as those $\mathbf{H}$ queries whose output has been set to $\mathbf{K}[i]$ (set S_i, initially empty). We note that if bad' occurs, then this means that at the end of the game there exists i such that $\mathbf{W}_{c-1}[i] \in X \setminus S_i$. We therefore change to a game B_6 which delays assigning the $\mathbf{W}$ values until after $\mathcal{D}$ executes. (Game B_6 also introduces other book-keeping code related to a new flag bad''', but this does not affect the game's functionality.) Queries to $\mathbf{H}$ are handled as before; the setting of bad' is omitted. The distribution of points assigned to $\mathbf{W}$ does not change compared to B_5. At the end, the game B_6 sets bad'' if there exists i with $\mathbf{W}_{c-1}[i] \in X \setminus S_i$. Thus, it can be verified that

$$\Pr\left[\, B_5^{\mathcal{D}} \text{ sets } \mathsf{bad}' \,\right] \leq \Pr\left[\, B_6^{\mathcal{D}} \text{ sets } \mathsf{bad}'' \,\right] .$$

The game B_6 also sets an additional condition bad''' if one of the following holds.

- First, the set $K := \{\mathbf{K}[1], \ldots, \mathbf{K}[r]\}$ and the set $W := \{\mathbf{W}_0[1], \ldots, \mathbf{W}_0[r]\}$ together contain overall less than $2r$ elements;

- Second, $H'[x]$ is set to a value which is in $X' \cup Y' \cup W \cup K$, where X' is the set of all entries x for which $H'[x]$ has been defined so far (including x), and Y' is the set of all previously set values $H'[x']$.

We now prove that if bad'' is set by the end of the game, then bad''' must have been set, which in turn yields that

$$\Pr\left[\, B_6^{\mathcal{D}} \text{ sets } \mathsf{bad}'' \,\right] \leq \Pr\left[\, B_6^{\mathcal{D}} \text{ sets } \mathsf{bad}''' \,\right] .$$

To see this, we show that $(\mathsf{bad}'' = \mathsf{true}) \wedge (\mathsf{bad}''' = \mathsf{false})$ at the end of the game yields a contradiction. More specifically, if $\mathsf{bad}''' = \mathsf{false}$ holds at the end of the game, $\mathbf{K}$, $\mathbf{W}$ values, and $H[\cdot]$ and $H'[\cdot]$ entries jointly satisfy the following:

- First, the values $\mathbf{W}_j[i]$ for $i = 1, \ldots, r$, $j = 0, \ldots, c-1$, as well as the keys $\mathbf{K}[1], \ldots, \mathbf{K}[r]$, are all distinct: This is because all elements in $K \cup W$ are distinct, and moreover, since $\mathbf{W}_j[i] = H'[\mathbf{W}_{j-1}[i]]$ for $j \geq 1$, $\mathbf{W}_j[i]$ cannot equal an element of $K \cup W$, or a previously set $\mathbf{W}$ value, without provoking bad'''.

- Second, for all $i = 1, \ldots, r$, there exists no w' such that $H'[w'] = \mathbf{W}_0[i]$, as otherwise bad''' would have occurd when defining $H'[w']$.

- Third, for all $j = 0, \ldots, c-2$ and $i = 1, \ldots, r$, we have $H[\mathbf{W}_j[i]] \in \{H'[\mathbf{W}_j[i]], \perp\}$. This is shown by induction over j: Assume it is true up to some $j \leq c-3$. Then, for all i, $H[\mathbf{W}_{j+1}[i]]$ being defined to something different than $H'[\mathbf{W}_{j+1}[i]]$ implies that $\mathrm{FindChain}(\mathbf{W}_{j+1}[i]) \neq \perp$, which cannot be by the above two bullets and the induction assumption that $H[\mathbf{W}_{j'}[i]] \in \{H'[\mathbf{W}_{j'}[i]], \perp\}$ for all $j' = 0, \ldots, j$.

Now assume that we indeed have $w^* = \mathbf{W}_{c-1}[i] \in X \setminus S_i$, and let us look at all entries in $H[\cdot]$ and $H'[\cdot]$ at the end of the game. There must exist $j^* \in \{0, \ldots, c-2\}$ such that, when the $\mathbf{H}$ query w^* was made,

$H'[\mathbf{W}_{j^*}[i]] = \bot$ (and hence $H[\mathbf{W}_{j^*}[i]] = \bot$ by the third bullet), because otherwise, we must have had $H[\mathbf{W}_j[i]] = H'[\mathbf{W}_j[i]]$ for all $j = 0, \ldots, c-2$ by the third bullet, and hence FindChain(w^*) would have returned $\mathbf{W}_0[i]$. Thus, in particular, $\mathbf{W}_{j^*}[i] \notin X'$ when w^* was queried. Therefore, there must exist $j^{**} \in \{j^*, \ldots, c-2\}$ such that $\mathbf{W}_j[i] \in X'$ for all $j = j^{**}+1\ldots, c-1$, but $\mathbf{W}_{j^{**}}[i] \notin X'$ when w^* was queried. Since by the end of the game $H'[\mathbf{W}_{j^{**}}[i]]$ is defined, however, bad''' must have been set to true at some point, i.e., a contradiction.

To upper bound the probability that bad''' is set, note that the probability that $W \cup K$ contains less than $2r$ elements is upper bounded by $\gamma(\mathcal{M}, r) + 2r^2 \cdot 2^{-n}$. Moreover, at most $(q + r(c-1))$ random values are inserted in the table $H'[\cdot]$, and each one of them can collide with at most $2(q + r(c-1)) + 2r = 2(q + cr)$ values. Therefore,

$$\Pr\left[\, B_6^{\mathcal{D}} \text{ sets } \mathsf{bad}''' \,\right] \le \gamma(\mathcal{M}, r) + \frac{2r^2 + 4(q+cr)^2}{2^n} \, .$$

The statement in the theorem follows by collecting terms. ▮

Analysis of $\mathsf{KD2}$. One can provide a similar analysis to the above for the (k, s, c)-KDF $\mathsf{KD2}$ described in Appendix E. The analysis models the function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ underlying $\mathsf{KD2}$ as a random oracle. We provide some brief intuition regarding such an analysis.

Here, a query (pw, x) is a chain completion query if there exist values $(pw, w_0), \ldots, (pw, w_{c-1})$ where $w_{c-1} = x$ and $w_0 = da \| sa$ for some $sa \in \{0,1\}^s$, and previous queries to F have set $F(pw, w_i) = w_{i+1}$ for all $i = 0, \ldots, c-2$. However, no previous query (pw, w') returned w_0.

The simulator keeps the history of the function F in form of a table $F[\cdot, \cdot]$. (As above, we denote by $F^{-1}[y]$ the corresponding set of preimages of y.) A sequence of values $w_0, w_1, \ldots, w_{c-1}$ is a *pw-chain* if there exists sa such that

- $w_0 = sa$ and $w_i = F[pw, w_{i-1}]$ for all $i = 1, \ldots, c-1$;
- For all $i = 1, \ldots, c-1$, we have $|F^{-1}[w_i]| = 1$; and $F^{-1}[w_0] = \emptyset$.

Upon a F query (pw, w), the simulator checks if there exists a pw-chain $w_0, w_1, \ldots, w_{c-1} = w$. In the affirmative case, it makes a **Test** query with input (pw, sa), obtaining an output K': If $K' \neq \bot$, then $F[pw, w_{c-1}]$ is set to $K' \oplus \bigoplus_{i=0}^{c-2} F[pw, w_i]$. In any other case (i.e., if $K' = \bot$ or no pw-chain exists), $F[pw, w]$ is set to a fresh random value.

A bound close to the one of Theorem 3.3 can be shown via a similar sequence of games.

H Proof of Theorem 3.4

Proof: We start with a game G_0 (boxed statements included, barred statements excluded) that implements the same functionality as $\mathrm{LORX}_{\mathcal{SE},m}$ for adversaries that queries $\mathbf{Enc}(i, \cdot, \cdot)$ ρ times for each $1 \le i \le m$. (There are two extra lines of code, one each preceding a boxed statement, but these have no effect on the game.) By construction

$$\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\mathrm{lorx}}_{\mathcal{SE},m}(\mathcal{A}) = \Pr\left[\, G_0^{\mathcal{A}} \Rightarrow \mathsf{true} \,\right] . \qquad (2)$$

Let S be a simulator. Game G_1 (Figure 8, boxed statements and bracketed statements omitted) transitions to using keys that are uniform and independent of the passwords. We have $m\rho$ keys and $m\rho$ salts and m passwords. We index passwords using i and salts/keys via (i, j). Here the adversary $\mathcal{A}$'s oracle **Prim** is replaced by the simulator S that has access to an oracle **Test**.

Let $\mathcal{M}$ be the message sampler that expects input $m\rho$ and first selects $\mathbf{pw}[1], \ldots, \mathbf{pw}[m] \leftarrow\!\$\; \mathcal{P}$; then selects $\mathbf{sa}[1], \ldots, \mathbf{sa}[m\rho] \leftarrow\!\$\; \{0,1\}^s$; and finally outputs the vector consisting of ρ copies of each of $\mathbf{pw}[i]$ for $i \in [1\,..\,m]$ and $\mathbf{sa}$. The random choice of salts gives that $\gamma(\mathcal{M}, \rho m) \le m^2\rho^2/2^s$.

To bound the transition between G_0 and G_1 we build a distinguisher $\mathcal{D}$ for $\mathcal{M}$ that works as follows. On input $(\mathbf{pw}, \mathbf{sa}, \mathbf{K})$, $\mathcal{D}$ implements G_0 using $\mathbf{pw}, \mathbf{sa}, \mathbf{K}$ (instead of generating them) and using its own

```
main $\boxed{G_0}$   $G_1$   $\langle\langle G_2 \rangle\rangle$
$\mathbf{pw}[1], \ldots, \mathbf{pw}[m] \leftarrow\$ \mathcal{P}$
For $j = 1$ to $m$ do
    $\mathbf{r}[i] \leftarrow 0$
    For $j = 1$ to $\rho$ do
        $\mathbf{sa}[i,j] \leftarrow\$ \{0,1\}^s$
        $\mathbf{K}[i,j] \leftarrow\$ \{0,1\}^k$
        $\boxed{\mathbf{K}[i,j] \leftarrow \mathsf{KD}^H(\mathbf{pw}[i], \mathbf{sa}[i,j])}$
$\mathbf{b} \leftarrow\$ \{0,1\}^m$
$b' \leftarrow\$ \mathcal{A}^{\mathbf{Enc},\mathbf{Cor},\mathbf{Prim}}$
Ret $(b' = \bigoplus_i \mathbf{b}[i])$
```

```
proc. Enc$(i, M_0, M_1)$
$\mathbf{r}[i] \leftarrow \mathbf{r}[i] + 1$ ; $j \leftarrow \mathbf{r}[i]$
$C \leftarrow\$ \mathcal{E}_{\mathbf{K}[i,j]}(M_{\mathbf{b}[i]})$
Ret $(\mathbf{sa}[i,j], C)$

proc. Prim$(X)$
$Y \leftarrow S^{\mathbf{Test}}(X)$
$\boxed{Y \leftarrow H(X)}$
Ret $Y$
```

```
proc. Cor$(i)$
$\mathsf{bad}_i \leftarrow \mathsf{true}$
Ret $(\mathbf{b}[i], \mathbf{pw}[i])$

proc. Test$(pw, sa)$
For $i = 1$ to $m$ do
    For $j = 1$ to $\rho$ do
        If $(pw, sa) = (\mathbf{pw}[i], \mathbf{sa}[i,j])$ then
            $\mathsf{bad}_i \leftarrow \mathsf{true}$
            $\langle\langle$ If $(\bigwedge_{i=1}^m \mathsf{bad}_i)$ then Ret $\perp$ $\rangle\rangle$
            Ret $\mathbf{K}[i,j]$
Ret $\perp$
```

```
main $\mathcal{B}^{\mathbf{Test},\mathbf{Cor}}(\mathbf{sa})$
For $i = 1$ to $m$ do
    $\mathbf{r}[i] \leftarrow 0$
    For $j = 1$ to $\rho$ do
        $\mathbf{K}[i,j] \leftarrow\$ \{0,1\}^k$
$\mathbf{b} \leftarrow\$ \{0,1\}^m$
$b' \leftarrow\$ \mathcal{A}^{\mathbf{Enc},\mathbf{CorSim},\mathbf{Prim}}$
Ret $\mathbf{pw}'$
```

```
proc. Enc$(i, M_0, M_1)$
$\mathbf{r}[i] \leftarrow \mathbf{r}[i] + 1$ ; $j \leftarrow \mathbf{r}[i]$
$C \leftarrow\$ \mathcal{E}_{\mathbf{K}[i,j]}(M_{\mathbf{b}[i]})$
Ret $(\mathbf{sa}[i,j], C)$

proc. Prim$(X)$
Ret $S^{\mathbf{TestSim}}(X)$
```

```
proc. CorSim$(i)$
$\mathbf{pw}'[i] \leftarrow \mathbf{Cor}(i)$
Ret $(\mathbf{b}[i], \mathbf{pw}'[i])$

proc. TestSim$(pw, sa)$
$(i, j) \leftarrow \mathbf{Test}(pw, sa)$
If $i \neq \perp$ then
    $\mathbf{pw}'[i] \leftarrow pw$
    Ret $\mathbf{K}[i,j]$
Ret $\perp$
```

```
main $G_3$   $\boxed{G_4}$
$\mathbf{pw}[1], \ldots, \mathbf{pw}[m] \leftarrow\$ \mathcal{P}$
For $j = 1$ to $m$ do
    $\mathbf{r}[i] \leftarrow 0$
    For $j = 1$ to $\rho$ do
        $\mathbf{sa}[i,j] \leftarrow\$ \{0,1\}^s$
        $\mathbf{K}[i,j] \leftarrow\$ \{0,1\}^k$
$\mathbf{b} \leftarrow\$ \{0,1\}^m$
$i^* \leftarrow\$ [1\,..\,m]$
$b' \leftarrow\$ \mathcal{A}^{\mathbf{Enc},\mathbf{Cor},\mathbf{Prim}}$
Ret $(b' = \bigoplus_i \mathbf{b}[i])$
```

```
proc. Enc$(i, M_0, M_1)$
$\mathbf{r}[i] \leftarrow \mathbf{r}[i] + 1$ ; $j \leftarrow \mathbf{r}[i]$
$C \leftarrow\$ \mathcal{E}_{\mathbf{K}[i,j]}(M_{\mathbf{b}[i]})$
Ret $(\mathbf{sa}[i,j], C)$

proc. Prim$(X)$
$Y \leftarrow S^{\mathbf{Test}}(X)$
Ret $Y$
```

```
proc. Cor$(i)$
$\mathsf{bad}_i \leftarrow \mathsf{true}$
If $(i = i^*)$ then $\mathsf{bad} \leftarrow \mathsf{true}$ $\boxed{\text{; Ret } \perp}$
Ret $(\mathbf{b}[i], \mathbf{pw}[i])$

proc. Test$(pw, sa)$
For $i = 1$ to $m$ do
    For $j = 1$ to $\rho$ do
        If $(pw, sa) = (\mathbf{pw}[i], \mathbf{sa}[i,j])$ then
            $\mathsf{bad}_i \leftarrow \mathsf{true}$
            If $(\bigwedge_{i=1}^m \mathsf{bad}_i)$ then Ret $\perp$
            If $i = i^*$ then $\mathsf{bad} \leftarrow \mathsf{true}$ $\boxed{\text{; Ret } \perp}$
            Ret $\mathbf{K}[i,j]$
Ret $\perp$
```

```
main $\mathcal{C}^{\mathbf{Enc}}$
$\mathbf{pw}[1], \ldots, \mathbf{pw}[m] \leftarrow\$ \mathcal{P}$
For $i = 1$ to $m$ do
    $\mathbf{r}[i] \leftarrow 0$
    For $j = 1$ to $\rho$ do
        $\mathbf{K}[i,j] \leftarrow\$ \{0,1\}^k$
$\mathbf{b} \leftarrow\$ \{0,1\}^m$
$i^* \leftarrow\$ [1\,..\,m]$
$b' \leftarrow\$ \mathcal{A}^{\mathbf{EncSim},\mathbf{Cor},\mathbf{Prim}}$
$c \leftarrow \bigoplus_{i \neq i^*} \mathbf{b}[i]$
If $(\mathsf{bad} = \mathsf{true})$ then $d \leftarrow\$ \{0,1\}$
Else $d \leftarrow 1$
Ret $b' \oplus c \oplus d$
```

```
proc. EncSim$(i, M_0, M_1)$
$\mathbf{r}[i] \leftarrow \mathbf{r}[i] + 1$ ; $j \leftarrow \mathbf{r}[i]$
If $i = i^*$ then $C \leftarrow \mathbf{Enc}(j, M_0, M_1)$
Else $C \leftarrow\$ \mathcal{E}_{\mathbf{K}[i,j]}(M_{\mathbf{b}[i]})$
Ret $(\mathbf{sa}[i,j], C)$

proc. Prim$(X)$
Ret $S^{\mathbf{TestSim}}(X)$
```

```
proc. Cor$(i)$
$\mathsf{bad}_i \leftarrow \mathsf{true}$
If $(i = i^*)$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; Ret $\perp$
Ret $(\mathbf{b}[i], \mathbf{pw}[i])$

proc. TestSim$(pw, sa)$
For $i = 1$ to $m$ do
    For $j = 1$ to $\rho$ do
        If $(pw, sa) = (\mathbf{pw}[i], \mathbf{sa}[i,j])$ then
            $\mathsf{bad}_i \leftarrow \mathsf{true}$
            If $(\bigwedge_{i=1}^m \mathsf{bad}_i)$ then Ret $\perp$
            If $i = i^*$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; Ret $\perp$
            Ret $\mathbf{K}[i,j]$
Ret $\perp$
```

Figure 8: Games and adversaries used in the proof of Theorem 3.4.

Prim oracle to respond to $\mathcal{A}$'s primitive queries. When $\mathcal{A}$ outputs b', the adversary $\mathcal{D}$ returns 1 if $b' = \bigoplus_i \mathbf{b}[i]$ and 0 otherwise. By inspection one can see that

$$\Pr\left[\,\mathrm{Real}^{\mathcal{D}}_{\mathsf{KD},\mathcal{M},m\rho} \Rightarrow 1\,\right] = \Pr\left[\,G_0^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] \quad \text{and} \quad \Pr\left[\,\mathrm{Ideal}^{\mathcal{D}}_{S,\mathcal{M},m\rho} \Rightarrow 1\,\right] = \Pr\left[\,G_1^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] .$$

Applying the above equations to (2) yields that

$$\Pr\left[\,G_0^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] \leq \Pr\left[\,G_1^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] + \mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{M},m\rho}(\mathcal{D}, S) \tag{3}$$

Game G_2 (Figure 8, boxed statement omitted, bracketed statements included) is the same as G_1 except after all flags bad_i for $1 \leq i \leq m$ are set. Let bad be a flag set to true when $(\bigwedge_{i=1}^{m} \mathsf{bad}_i)$ first becomes true in either game G_1 or G_2. Games G_1 and G_2 are identical-until-bad and so by the fundamental lemma of game-playing [8]

$$\Pr\left[\,G_1^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] \leq \Pr\left[\,G_2^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] + \Pr\left[\,G_1^{\mathcal{A}} \text{ sets } \mathsf{bad}\,\right] . \tag{4}$$

We are now in position to bound the probability that bad is set in G_2 by building an $\mathrm{GUESS}_{\mathcal{P},m,\rho}$ adversary $\mathcal{B}$. It works as shown in Figure 8. On input $\mathbf{sa}$, it sets $\mathbf{r}$, $\mathbf{K}$ and $\mathbf{b}$ as in G_2's main procedure and then runs $\mathcal{A}$. To answer encryption query $\mathbf{Enc}(i, M_0, M_1)$, $\mathcal{B}$ executes the procedure from G_2. To answer **Test** queries from S, it implements the **TestSim** procedure by querying its own **Test** oracle on the queried pw, sa. If the returned pair $(i,j) \neq (\perp,\perp)$ then it sets $\mathbf{pw}'[i]$ to be pw and returns $\mathbf{K}[i,j]$. Otherwise it returns $\perp$. To answer **CorSim** queries, it queries its own corruption oracle **Cor**, sets $\mathbf{pw}'[i]$ to the returned password pw and returns $\mathbf{b}[i], pw$. By construction

$$\Pr\left[\,G_2 \text{ sets } \mathsf{bad}\,\right] \leq \mathbf{Adv}^{\mathrm{guess}}_{\mathcal{P},m,\rho}(\mathcal{B}) .$$

We now move on to bound the success probability that $\mathcal{A}$ can force the game to output true in G_2. Game G_3 (boxed statement omitted) is equivalent to G_2, so $\Pr[G_3^{\mathcal{A}} \Rightarrow \mathsf{true}] = \Pr[G_2^{\mathcal{A}} \Rightarrow \mathsf{true}]$. Game G_4 adds the boxed statement to game G_3. It ensures that $\perp$ is always returned in the case that the simulator guesses correctly a password and salt pair associated with instance i^* or in the case that i^* is corrupted. Games G_3 and G_4 are identical-until-bad, where the flag is set now if the value i^* failed to predict the value of i for which bad_i was last to be set to true (or not set at all). Applying a variant of the fundamental lemma of game playing [8] gives that

$$\frac{1}{m}\Pr\left[\,G_3^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] = \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow \mathsf{true} \wedge \mathsf{good}\,\right]$$

where good is the event that bad was not set in the course of the game. Now in G_4 the keys and challenge bit associated to instance i^* are never used outside of **Enc**. We can therefore build from game G_4 an mu-$\mathrm{LOR}_{\mathsf{SE},\mathcal{K},\rho}$ adversary $\mathcal{C}$ that will play against the i^* instance, using its own multi-user oracle to respond to encryption queries against that instance. All the other instances, and all the passwords and salts, are simulated by $\mathcal{C}$ for $\mathcal{A}$. A specification of $\mathcal{C}$ is given in Figure 8. Note that it chooses extraneous values $\mathbf{K}[i^*, j]$ and $\mathbf{b}[i^*]$ which are not used in the rest of the game. The final output of $\mathcal{C}$ is the guess of $\mathcal{A}$ but with the rest of the (known) challenge bits stripped away. Should bad have been set while responding to **Test** queries, then $\mathcal{C}$ outputs a random bit at the end of the game. In the following, let good be again the event that bad is not set in the course of the game, which corresponds to the guess i^* being apt. By inspection we see that $\mathcal{C}$ simulates perfectly G_4 for $\mathcal{A}$. Thus

$$\begin{aligned}
\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{mu-lor}}_{\mathsf{SE},\mathcal{K},\rho}(\mathcal{C}) &= \Pr\left[\,\text{mu-LOR}^{\mathcal{C}}_{\mathcal{SE},\mathcal{K},\rho} \Rightarrow \mathsf{true}\,\right] \\
&= \Pr\left[\,\text{mu-LOR}^{\mathcal{C}}_{\mathcal{SE},\mathcal{K},\rho} \Rightarrow \mathsf{true} \wedge \mathsf{good}\,\right] + \frac{1}{2}\left(1 - \Pr\left[\,\mathsf{good}\,\right]\right) \\
&= \Pr\left[\,G_4^{\mathcal{A}} \Rightarrow \mathsf{true} \wedge \mathsf{good}\,\right] + \frac{1}{2} - \frac{1}{2m} .
\end{aligned}$$

We have that $\Pr[\mathsf{good}] = 1/m$. Rearranging the final equivalence gives

$$\Pr\left[\,G_4^{\mathcal{A}} \Rightarrow \mathsf{true} \wedge \mathsf{good}\,\right] = \frac{1}{2}\mathbf{Adv}^{\text{mu-lor}}_{\mathsf{SE},\mathcal{K},\rho}(\mathcal{C}) + \frac{1}{2m}$$

which, combining with the inequalities above, gives the following sequence of inequalities

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{lorx}}_{\mathcal{SE},\mathcal{P},m}(\mathcal{A}) &= 2\cdot\Pr\left[\,G_0^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] - 1 \\
&\leq 2\cdot\Pr\left[\,G_1^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] + 2\cdot\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{M},m\rho}(\mathcal{D},S) - 1 \\
&\leq 2\cdot\Pr\left[\,G_2^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] + 2\cdot\mathbf{Adv}^{\mathrm{guess}}_{\mathcal{P},m,\rho}(\mathcal{B}) + 2\cdot\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{M},m\rho}(\mathcal{D},S) - 1 \\
&= 2\cdot\Pr\left[\,G_3^{\mathcal{A}} \Rightarrow \mathsf{true}\,\right] + 2\cdot\mathbf{Adv}^{\mathrm{guess}}_{\mathcal{P},m,\rho}(\mathcal{B}) + 2\cdot\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{M},m\rho}(\mathcal{D},S) - 1 \\
&= 2m\cdot\Pr\left[\,G_4^{\mathcal{A}} \Rightarrow \mathsf{true} \wedge \mathsf{good}\,\right] + 2\cdot\mathbf{Adv}^{\mathrm{guess}}_{\mathcal{P},m,\rho}(\mathcal{B}) + 2\cdot\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{M},m\rho}(\mathcal{D},S) - 1 \\
&= m\cdot\mathbf{Adv}^{\mathrm{mu\text{-}lor}}_{\mathsf{SE},\mathcal{K},\rho}(\mathcal{C}) + 2\cdot\mathbf{Adv}^{\mathrm{guess}}_{\mathcal{P},m,\rho}(\mathcal{B}) + 2\cdot\mathbf{Adv}^{\mathrm{kdf}}_{\mathsf{KD},\mathcal{M},m\rho}(\mathcal{D},S)
\end{aligned}$$

▮