A preliminary version of this paper appears in the proceedings of CT-RSA 2023. This is the full version.

Flexible Password-Based Encryption: Securing Cloud Storage and Provably Resisting Partitioning-Oracle Attacks

Mihir Bellare[1] Laura Shea[2]

February 18, 2023

Abstract

We introduce flexible password-based encryption (FPBE), an extension of traditional password-based encryption designed to meet the operational and security needs of contemporary applications like end-to-end secure cloud storage. Operationally, FPBE supports nonces, associated data and salt reuse. Security-wise, it strengthens the usual privacy requirement, and, most importantly, adds an authenticity requirement, crucial because end-to-end security must protect against a malicious server. We give an FPBE scheme called DtE that is not only proven secure, but with good bounds. The challenge, with regard to the latter, is in circumventing partitioning-oracle attacks, which is done by leveraging key-robust (also called key-committing) encryption and a notion of authenticity with corruptions. DtE can be instantiated to yield an efficient and practical FPBE scheme for the target applications.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: mihir@eng.ucsd.edu. URL: http://cseweb.ucsd.edu/~mihir/. Supported in part by NSF grant CNS-2154272.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: lmshea@ucsd.edu. Supported by NSF grants CNS-2048563 and CNS-1513671.

Contents

Type	encryption	salt	Security	
			Privacy	Authenticity
TPBE	randomized	fresh per encryption	✓	
FPBE	deterministic, nonce-based	reusable across encryptions at discretion of application	✓	✓

Figure 1: Traditional password-based encryption (TPBE) and flexible password-based encryption (FPBE), contrasted.

1 Introduction

This paper advances password-based encryption (PBE) to meet the operational and security needs of contemporary applications like end-to-end secure cloud storage. What we call Flexible password-based encryption (FPBE) adds support for nonces and associated data; ups the privacy requirement to IND\$; asks for authenticity in addition to privacy; and gives a scheme that is not only proven secure, but *with good bounds*. The key challenge, with regard to the latter, is provably resisting partitioning-oracle attacks [29]. We begin with some background.

Traditional PBE. PBE, currently, is closely identified with the canonical method of doing it. As rendered in the PKCS#5 standard [26], the method is: to encrypt message M under password P, pick a random salt S, obtain a key $K \leftarrow \mathrm{H}(P, S)$ by hashing the salt and password, and return as ciphertext (S, C) where $C \leftarrow \mathsf{SE.Enc}(K, M)$ is an encryption of M under K using a conventional symmetric encryption scheme SE. (We refer to SE as the *base* scheme.) From this, PBE emerges simply as randomized symmetric encryption in which the key (shared between sender and receiver) is a password, and this indeed was the syntax adopted. For this syntax, BRT [12] give a definition of message-privacy under chosen-plaintext attack, and prove that the canonical scheme meets this if passwords are unpredictable, the base scheme SE provides privacy and H is a random oracle. Importantly for what is coming, these results fail to define, or prove, authenticity.

In summary, traditional PBE (which we abbreviate as TPBE) is randomized, privacy-only encryption with a fresh, per-message salt. We now introduce FPBE. As a quick summary, Figure 1 contrasts TPBE and FPBE.

1.1 Flexible PBE

FPBE involves a new syntax, and security definitions for it, that we discuss in turn. Formal definitions are in Section 5.

Syntax. Unlike a regular symmetric encryption scheme, an FPBE scheme FPBE has neither a prescribed key-length nor a key-generation process; the key, now denoted P to connote a password, can be any string, of any length. (Security will depend on the distribution of P.) Encryption is deterministic, taking the salt S as input, and now, as in AEAD [39], a nonce N and associated data A: we write $C \leftarrow \mathsf{FPBE.Enc}(P, S, N, M, A)$. Decryption recovers as $M \leftarrow \mathsf{FPBE.Dec}(P, S, N, C, A)$, with the salt, nonce and associated data being sent out of band.

Security. We consider a multi-user setting where $\mathbf{P}[i]$ is the password of user $i \in \{1, \ldots, u\}$. The distribution on the vector of passwords, denoted PD, captures the strength of choices made and parameterizes definitions of privacy and authenticity. Then we formalize the following.

1. Privacy. Denoted PIND\$, this asks that ciphertexts under the hidden, target password vector

are indistinguishable from random strings when the salt S is honestly (randomly) chosen by the game and known to the adversary, and a nonce is not repeated for a given salt. (The adversary can at any time ask for a salt refresh, and a nonce is allowed to be reused once this happens.) The game formalizing this gives oracles SALT (to obtain a fresh salt for a given user) and ENC (that returns a challenge ciphertext, obtained either by encryption under the password of the indicated user, or chosen at random).

2. **Authenticity.** Denoted PAUTH, this asks that it be infeasible to produce S, N, C, A that is valid — meaning $\mathsf{FPBE.Dec}(P, S, N, C, A) \neq \perp$ — except in a trivial way. Note that in the forgery attempt, the adversary gets to pick the salt; it does not need to be an honest one used in encryption. The game has oracle ENC return encryptions under the password of the indicated user, and an oracle VERIFY that allows the adversary to make multiple forgery attempts.
3. **PAE.** This captures privacy and authenticity in a single, integrated way. The game gives the adversary oracles SALT, ENC, DEC where SALT, ENC are like in PIND\$ and DEC is similar to VERIFY in PAUTH in the real case and returns $\perp$ in the ideal case.

PIND\$+PAUTH $\Rightarrow$ PAE. Following [11, 16], we show in Theorem 5.1 that if an FPBE scheme separately satisfies PIND\$ and PAUTH, then it also satisfies PAE. Importantly, this result only requires PAUTH to hold for a restricted class of adversaries, called sequential; they make all their ENC, SALT queries before their VERIFY queries. Nonetheless, PAE holds fully, meaning even for non-sequential adversaries. This allows us, for PAUTH, to restrict attention to sequential adversaries, which simplifies proofs.

FEATURES. Our framework allows a salt to be securely reused to encrypt multiple messages, as long as the nonce is different each time. Associated data could be metadata (such as a file handle) and, as per [39], is authenticated but not encrypted. Privacy strengthens that of TPBE by requiring indistinguishability from random rather than indistinguishability of encryptions, which provides some degree of anonymity. But the main value added is authenticity, not present in TPBE, and crucial for the applications to which we now turn.

1.2 Motivation and applications

We discuss three motivations or applications for this work.

SECURING CLOUD STORAGE. Almost all cloud storage providers provide some type of encryption for data at rest. In a first tier, represented by GoogleDrive, DropBox and Microsoft, encryption is under a key known to the server. More interesting is a second tier of services like MEGA [32] and Boxcryptor [17] that aim to provide end-to-end secure storage, where the encryption is under a key known only to the user, so that even the service provider storing the encrypted file cannot decrypt it. This security goal is coupled with an availability one: a user should be able to access the server and decrypt her files from any of her devices. A solution has been to encrypt the files under a user password. This second tier of systems has been highly successful; MEGA alone claims to be storing 1,000 PB of password-encrypted data [33].

Enormous volumes of data thus are, or will be, password-encrypted for cloud storage. So we ask, what PBE schemes should we use? The first answer is traditional PBE. But TPBE is a poor fit for this task because, as we explain below, secure cloud storage doesn't just require privacy; it also requires authenticity. TPBE does not provide this; FPBE does.

Why authenticity? In end-to-end security, the intent is to maintain security even when the server is malicious. (This model reflects a variety of real-world threats. One is insider attacks, mounted by provider employees. Or, the provider's systems may be infiltrated by hackers.) Suppose the user has

placed on the server a ciphertext C encrypting a file M under the user's password. In the absence of authenticity, a malicious server could modify C to another ciphertext C' that, when retrieved by the user, decrypts under the user password to $M' \neq M$. Considering that in this way the malicious server could modify financial or personal data, lack of authenticity has critical consequences. The threat is not merely speculative; there are attacks on MEGA that violate authenticity of stored encrypted files [6]. Authenticity is thus a core requirement for FPBE.

Besides enhancing security, FPBE can reduce storage cost. Specifically, q messages encrypted under TPBE with sl-bit random salts add $\mathrm{sl} \cdot q$ bits of ciphertext storage overhead. With FPBE, one can use one random salt, and then a c-bit counter as nonce, for storage overhead $\mathrm{sl} + qc$. The latter is lower than the former because the counter can be short (say, 16 bits for $q \leq 2^{16}$) while salts need to resist collisions so would need to be 128 bits or more.

Modernizing PBE. Symmetric encryption has evolved. Failures of privacy-only schemes lead to the consensus that the goal should be *authenticated* encryption [10]. Alongside, randomized encryption has given way to nonce-based encryption supporting associated data (AEAD) [39]. Part of our motivation was to reflect these lessons and advances in PBE and align it with AEAD. Thus, FPBE adds support for nonces and associated data and, most importantly, provides authenticity in addition to privacy. The PIND\$, PAUTH and PAE definitions we give mimic corresponding ones for AEAD from the literature [11, 39].

Provably resisting partitioning-oracle attacks. Recall that TPBE uses a (conventional) symmetric encryption scheme that we call the base scheme. (Our **DtE** FPBE scheme will too.) Also recall that such a base scheme is key-robust (also called key-committing) [1,2,7,22,24] if a ciphertext is a commitment to the key. Surprising new attacks, called partitioning-oracle attacks [29], exploit lack of key-robustness in the base scheme to speed up password recovery in the corresponding TPBE scheme. The attacks need access to decryption capability under the target password and thus, crucially, cannot be captured or understood within the prior, ind-cpa-style privacy-only frameworks of PBE [12, 18]. Our FPBE framework fills this gap; the attacks now emerge as aiming to violate *authenticity*. This puts us in a position to ask whether presence of key-robustness in the base scheme provably provides resistance against partitioning-oracle attacks. (We will show that the answer is yes.)

1.3 Security of the DtE scheme

The **DtE** scheme. We build FPBE from two ingredients: a conventional AEAD scheme SE [39] and a password-based key-derivation function (PBKDF) F. Formally our construction is a transform **DtE** (**D**erive **t**hen **E**ncrypt) that defines FPBE scheme $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ as follows: $\mathsf{FPBE.Enc}(P, S, N, M, A)$ derives $K \leftarrow \mathsf{F}(P, S)$ and returns $C \leftarrow \mathsf{SE.Enc}(K, N, M, A)$. This extends BRT [12] and the classical PKCS#5 standard [26] to our setting. Practical choices for the PBKDF F include PBKDF2 [26], BCRYPT [38], SCRYPT [3, 4, 35] or Argon2 [15]. Some of our results assume F is a random oracle [13]. The assumptions on SE vary. The assumptions on passwords are discussed next.

Password strength. PBE (whether TPBE or FPBE) can only provide security when passwords are strong, meaning are hard to guess. (This is due to brute-force dictionary attacks.) Proofs for TPBE [12] made a necessary "password un-guessability" assumption on the password distribution PD, and this work will do so as well.

The metric for un-guessability is the guessing probability $\mathbf{GP}_{\mathsf{PD}}(q)$, defined, for any given integer parameter $q \geq 1$, as the maximum, over all $(i_1, P_1), \ldots, (i_q, P_q)$, of the probability that there is some j such that $P_j = \mathbf{P}[i_j]$ when $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$ [12, 42]. It emerges that un-guessability is

u	q_s, q_e	q		
		Th. 6.1 SE: ind$ xx = pind$	Th. 6.2 SE: auth xx = pauth	Th. 6.3 SE: auth+krob$ xx = pauth
> 1	> 0	q_h	$q_h + \min(q_v, u) \cdot q_h$	$q_h + q_v$
> 1	$= 0$	q_h	$\min(q_v, u) \cdot q_h$	q_v
$= 1$	> 0	q_h	$2q_h$	$q_h + q_v$
$= 1$	$= 0$	q_h	q_h	q_v

Figure 2: **Security of $\mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ as a function of password strength.** For $\text{xx} \in \{\text{pind\$}, \text{pauth}\}$, our results give bounds of the form $\mathbf{Adv}^{\text{xx}}_{\mathbf{DtE}[\mathsf{SE},\mathsf{F}],\mathsf{PD},u}(A) \leq \mathbf{GP}_{\mathsf{PD}}(q) + \delta$. The table shows the value of the number q of password guesses in this bound for privacy (xx = pind\$), authenticity (xx = pauth) when we assume only auth-security of SE (Th. 6.2) and authenticty when we also assume key-robustness (krob\$) of SE (Th. 6.3). Here u is the number of users and q_s, q_e, q_h the number of Salt, Enc, H queries, respectively, of A. Additionally, in the xx = pauth case, q_v is the number of Verify queries of A. The δ term is secondary and is in the theorem statements.

---

not a monolithic assumption; the smaller the number q of guesses, the weaker the assumption. An important element of our bounds is keeping q as low as possible.

Security of **DtE**. The scheme we analyze is $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ with $\mathsf{F}(P, S) = \mathrm{H}(P, S)$ where H is modeled as a random oracle. The analysis can be seen at two levels. The first, more superficial level, is asymptotic (or qualitative), where we seek to name assumptions that imply security. The second, technically deeper and in practice more relevant level is concrete (or quantitative), where we seek to obtain bounds as good as possible. Let us visit these in turn.

Asymptotic security. Assuming passwords are un-guessable, (1) Theorem 6.1 says that if base scheme SE provides privacy, then FPBE meets our PIND$ definition of privacy for FPBE, (2) Theorem 6.2 says that if base scheme SE provides authenticity, then FPBE meets our PAUTH definition of authenticity for FPBE, and (3) Theorem 6.3 says that if base scheme SE provides both authenticity and key-robustness, then FPBE again meets PAUTH. Item (3), at this level, looks redundant; why do we add an extra assumption (key-robustness) on SE to obtain the same conclusion as in (2)? The answer is better concrete security and resistance to partitioning-oracle attacks, which emerges only at the concrete level we discuss next.

Concrete security. The three above-mentioned theorems bound the advantage of a given adversary A, in violating privacy or authenticity of $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$, by an expression of the form $\mathbf{GP}_{\mathsf{PD}}(q) + \delta$, for a q that depends on adversary resources. The number q of guesses emerges as a crucial parameter; the lower it is, the better the result. Our quest is to minimize this value. The δ term in the bound, shown in the theorem statements, will involve advantages of constructed adversaries in violating the security of SE, as well as a salt-collision term. It is secondary to $\mathbf{GP}_{\mathsf{PD}}(q)$ assuming a long enough salt and secure SE.

The primary adversary resource is the number q_h of H queries, corresponding to offline computations of $\mathsf{F} = \mathrm{H}$. Other resources are the number q_s, q_e of queries to the above-mentioned Salt, Enc oracles, corresponding to the number of encryptions performed, and additionally, for PAUTH, the number q_v of queries to the Verify oracle, representing the number of allowed verification attempts. Relevant below is that, due to throttling or other mitigations, q_v could be very small and in particular $q_v \ll q_h$.

The values of q for our bounds are summarized in Figure 2. For privacy (PIND\$) of FPBE, the bound of Theorem 6.1 is $\mathbf{Adv}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \leq \mathbf{GP}_{\mathsf{PD}}(q_h) + \delta$, meaning $q = q_h$. Furthermore, we show this bound is tight: leveraging the classical brute-force attack, Proposition 8.1 gives an attack making q_h H queries and violating PIND\$ with probability about $\mathbf{GP}_{\mathsf{PD}}(q_h)$. This yields a full picture for privacy.

Authenticity is more involved. Theorems 6.2 and 6.3 give bounds of the form $\mathbf{Adv}^{\mathrm{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A)$ $\leq \mathbf{GP}_{\mathsf{PD}}(q) + \delta$. The table of Figure 2 has two segments, with two rows in each. The first segment is the general case with u users, but a simpler example shows the $u = 1$ case of the second segment. In both segments, we consider first the case that encryptions are present ($q_s, q_e > 0$). But the case where they are not ($q_s = q_e = 0$) is in fact important; it can arise when FPBE is used in a protocol aiming for security against dictionary attacks.

We now explain the simplest case, that of the 4th (last) row. While q from Theorem 6.2 is q_h, additionally assuming key-robustness of SE drops it, via Theorem 6.3, to q_v, which, as noted above, is usually significantly smaller than q_h due to throttling or other limitations on verification attempts. The gap is less, but still present, in row 3. The gap is even more stark in the first segment, where the q given by Theorem 6.2 has a product term $\min(q_v, u) \cdot q_h$ that drops to just q_v with Theorem 6.3.

The conclusion is that key-robustness of SE is significantly improving the quantitative authenticity guarantees for FPBE. This is the proven security against partitioning-oracle attacks that we have sought.

PAE. We clarify that the above results for PAUTH assume that the adversary is sequential. We can confine attention to this case due to Theorem 5.1 which (as indicated above) says that PAUTH for sequential adversaries, combined with PIND\$, implies the integrated PAE definition for *all* (not necessarily sequential) adversaries. Theorems 6.4 and 6.5 put things together to show PAE for **DtE** for unrestricted adversaries.

Bound tightness via attacks. One might worry that the gap above is not real, but rather an artifact of a loose analysis in Theorem 6.2. In fact, attacks show that our bounds are tight and the gap is thus real. Moreover, this is where we complete the circle to partitioning-oracle attacks [29]. Proposition 8.2 shows that if SE is not key-robust then these attacks can be used to violate PAUTH with probability roughly $\mathbf{GP}_{\mathsf{PD}}(q)$ where q is as shown in the Theorem 6.2 column of Figure 2. (The actual claim relies on a more fine-grained parameterization.)

Techniques. A possible perception is that security of $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ is trivial due to the following intuition: the key $K \leftarrow \mathsf{F}(P, S)$ is random so the assumed security of SE yields the conclusion. This only scratches the surface, and ignores concrete security, which is where the main subtleties and challenges arise. In particular, the proof of Theorem 6.3 involves new techniques. The difficulty is that it is not obvious how key-robustness of SE helps improve the bound or how to exploit it in the proof. The naive analysis would have a password-guessing adversary make one guess per hash query of the PAUTH adversary A, returning us to the bound of Theorem 6.2. Very roughly, key-robustness allows us to avoid this by using decryption instead. The proof of Theorem 6.3 (in Appendix C) relies on a lemma, of possibly independent interest, concerning authenticity with corruptions (AUTH-C) of SE. A standard hybrid argument shows that AUTH-C is implied by AUTH with a factor u loss in advantage, where u is the number of users [25]. We show in Lemma 4.2 that a *tight* reduction is possible when there are no encryption queries. Despite the fact that the given adversary A is allowed encryption queries in the PAUTH game, we are able to reduce to the AUTH-C security of SE in the absence of encryption queries and thence, by the lemma, tightly to the regular AUTH security of SE.

INSTANTIATION. To take advantage of the above results in the form of high-security FPBE schemes, we need base AEAD schemes SE that provide privacy, authenticity and key-robustness. Attacks from [2, 29] show that current schemes like GCM [20] fail to be key-robust; indeed, this is the basis of partitioning-oracle attacks. However, key-robust schemes have been provided in [2, 7, 19, 24], with the last work in particular giving a GCM variant that adds key-robustness with essentially no overhead. This yields numerous choices of base scheme SE that, when plugged into **DtE**, yield efficient, high-security FPBE.

COMMITTING SECURITY OF **DtE**. We saw above that **DtE** preserves privacy and authenticity of the base symmetric encryption scheme SE. We also show that it does the same for robustness, or committing security. There are various definitions of robustness or committing security for which we could show this, but we chose to use the strongest, from [7]. They define CMT-ℓ security of the base scheme SE for $\ell = 1, 3, 4$. We extend these to define PCMT-ℓ security of an FPBE scheme. Then in Proposition 9.1 we show that if the base scheme SE is CMT-ℓ secure and F is collision-resistant then the FPBE scheme $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ is PCMT-ℓ-secure.

1.4 Extended setting and results

What we have discussed above are, for simplicity, special cases of our definitions and results; the ones in the body of the paper are more general along several dimensions that we now summarize.

In defining authenticated encryption, we can consider two dimensions. The first dimension relates to nonce reuse; it is either prohibited (unique-nonce or basic security), or allowed with the stronger guarantee of nonce-misuse resistance [40], also called advanced security. The second dimension relates to decryption; in the NBE1 syntax and corresponding AE1 notion of security [11] the nonce is an explicit decryption input, while in the NBE2 syntax and corresponding, stronger, AE2 notion of security, it isn't. With two choices in each of two dimensions, we have four possible models or definitions. What we discussed above has been the simplest case, namely unique nonces and NBE1/AE1; this, called AEAD [39], is what was assumed of the base symmetric encryption scheme, and then extended to FPBE. In the body of the paper, we consider all four models, in a compact and unified way, first giving a single, parameterized syntax and corresponding security definitions for regular symmetric encryption and then also for FPBE. Our results are stated and proved also in a general way, fairly seamlessly covering all these variants. Through **DtE** and our results about it, we now obtain FPBE schemes for all four regimes; in particular we can provide nonce-misuse resistance and AE2 security.

2 Related work

Bellare, Ristenpart and Tessaro (BRT) [12] study PBE in the multi-instance setting, while our results are in the more classical multi-user setting. Demay, Gaźi, Maurer and Tackmann [18] show limits on multi-instance security in the constructive cryptography setting.

In the applications we consider, notably end-to-end secure storage, the server can run brute-force attacks, so security is only possible with strong (un-guessable) passwords. Password hardening through the use of an auxiliary server [21,27,28] could potentially be added to the system to mitigate these attacks.

Better password-based key-derivation methods could also make brute-force attacks more expensive. For example, Argon2 [15], the winner of the 2013–2015 Password Hashing Competition, and other options like BCRYPT [38] and SCRYPT [3, 4, 35] are designed to be memory-hard or otherwise computationally expensive so that brute-force (dictionary) attacks are costly. Our results

in Section 6 (namely, Theorems 6.1, 6.2, 6.3) model the PBKDF as a random oracle, and results are expressed in terms of the number of queries q_h to the random oracle. The particular PBKDF determines how expensive these q_h queries are for an adversary. We suggest a new property of PBKDFs, kd security in Section 7, that yields useful results for FPBE in the standard model.

Len, Grubbs and Ristenpart (LGR) [29] introduced the partitioning-oracle attack and implemented a working attack on a PBE application called Shadowsocks [41]. While they observed that key-robustness can foil the attack, it remained an open question as to how it might provably increase the security of PBE. Our work fills this gap; Theorem 6.3, shows that yes, key-robustness does concretely improve authenticity (PAUTH) guarantees. In Proposition 8.2 of Section 8, we additionally prove that the authenticity bound of Theorem 6.2 is tight, using a partitioning-oracle attack. FPBE thus allows us to resolve how partitioning-oracle attacks and key-robustness fit into the provable security picture of password-based encryption.

Armour and Cid [5] describe how weak key forgeries can be used to mount partitioning-oracle attacks. They generalize the setting of LGR [29] and obtain attacks in new settings that are resistant to the LGR attacks, such as when plaintexts are formatted.

Pijnenburg and Poettering [37] introduce Encrypt-to-Self as a comprehensive model and solution for secure outsourced storage. Their security requirements are stronger than ours; they aim to preserve authenticity of data even if the key (password) is compromised, and for this allow the user to have some amount of local storage for hashes.

Related to robustness, Len, Grubbs and Ristenpart [30] consider AEAD with key identification, where the decryptor has a list of keys and must identify which one decrypts a given ciphertext.

3 Preliminaries

NOTATION AND TERMINOLOGY. By ε we denote the empty string. By $|Z|$ we denote the length of a string Z. By $x\|y$ we denote the concatenation of strings x, y. If S is a finite set, then $|S|$ denotes it size. We say that a set S is *length-closed* if, for any $x \in S$ it is the case that $\{0,1\}^{|x|} \subseteq S$. (This will be a requirement for message, header, nonce and salt spaces.) A vector $\mathbf{V}$ is denoted in bold. We denote its length by $|\mathbf{V}|$ and entry i by $\mathbf{V}[i]$ for $1 \le i \le |\mathbf{V}|$.

If X is a finite set, we let $x \leftarrow\!\!\$\, X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms are deterministic unless otherwise indicated. If A is a deterministic algorithm, we let $y \leftarrow A[\mathrm{O}_1, \ldots](x_1, \ldots)$ denote running A on inputs $x_1, \ldots$, with oracle access to $\mathrm{O}_1, \ldots$, and assigning the output to y. An adversary is an algorithm. Running time is worst case, which for an algorithm with access to oracles means across all possible replies from the oracles. We use $\perp$ (bot) as a special symbol to denote rejection, and it is assumed to not be in $\{0,1\}^*$.

To concisely state our results, it will be helpful to define the function zt (zero test) via $\mathsf{zt}(q) = 0$ if $q = 0$ and $\mathsf{zt}(q) = 1$ if $q \neq 0$. In some of our games and adversaries, we will use an algorithm Find1 that takes a value S and a vector $\mathbf{S}$ to return an integer $i \leftarrow \mathrm{Find1}(S, \mathbf{S}) \in \{0, 1, \ldots, |\mathbf{S}|\}$ such that: if $S \in \{\mathbf{S}[1], \ldots, \mathbf{S}[|\mathbf{S}|]\}$ then i is the smallest integer such that $\mathbf{S}[i] = S$, and otherwise $i = 0$. An extension, algorithm Find2, takes S and a list $\mathbf{S}_1, \ldots, \mathbf{S}_n$ of vectors, returning $i \leftarrow \mathrm{Find2}(S, \mathbf{S}_1, \ldots, \mathbf{S}_n) \in \{0, 1, \ldots, n\}$ such that i is the smallest value such that $\mathrm{Find1}(S, \mathbf{S}_i) \neq 0$ if this exists, and otherwise $i = 0$. That is, Find2 identifies the first vector in which S occurs, if any.

GAMES. We use the code-based game-playing framework of BR [14]. A game G starts with an optional INIT procedure, followed by a non-negative number of additional procedures called oracles, and ends with a FIN procedure. Execution of adversary A with game G consists of running A with oracle access to the game procedures, with the restrictions that A's first call must be to INIT (if present), its last call must be to FIN, and it can call these procedures at most once. The output of

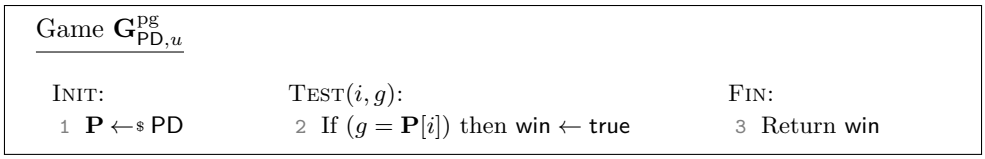

Figure 3: The guessing game for a u-user password distribution PD.

the execution is the output of Fin. By $\Pr[\mathrm{G}(A)]$ we denote the probability that the execution of game G with adversary A results in this output being the boolean true.

Note that our adversaries have no output. The role of what in other treatments is the adversary output is, for us, played by the query to Fin. Different games may have procedures (oracles) with the same names. If we need to disambiguate, we may write G.O to refer to oracle O of game G. In games, integer variables, set variables, boolean variables and string variables are assumed initialized, respectively, to 0, the empty set $\emptyset$, the boolean false and $\perp$. Tables are initialized with all entries being $\perp$.

Password distributions. A distribution over passwords, PD, returns a u-vector of passwords, where u, a parameter associated to PD, is the number of users; we write $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$. This is neither a password-generation algorithm nor a prescription for how to generate passwords; rather it attempts to model and capture choices that people make. The passwords are not assumed to be independent; reflecting password choices in practice, they may be arbitrarily correlated. (In particular, a person may use related passwords for different websites.) We do assume that passwords in a vector are distinct. (Formally, $\mathbf{P}[1], \ldots, \mathbf{P}[u]$ are all distinct, for all $\mathbf{P}$ that may be generated by PD.) This is because usage of the same password across different users leads to trivial attacks.

Password guessing. We are interested in an adversary's ability to guess some entry of a password vector in some number q of tries. Following [12], we measure this via a guessing game. The game $\mathbf{G}^{\mathrm{pg}}_{\mathsf{PD},u}$ is in Figure 3. A guess is captured by a Test query. Note that the Test oracle returns no response to the adversary, so that the attack is effectively non-adaptive. For an adversary A, we define the guessing advantage $\mathbf{Adv}^{\mathrm{pg}}_{\mathsf{PD},u}(A) = \Pr[\mathbf{G}^{\mathrm{pg}}_{\mathsf{PD},u}(A)]$ to be the probability that the game returns true.

In proofs it is convenient to use the game- and advantage-based definition above. However, the results are best expressed via an equivalent information-theoretic formulation in terms of guessing probabilities and min-entropy. For a number q of guesses, we define the guessing probability $\mathbf{GP}_{\mathsf{PD}}(q)$ and min-entropy $\mathbf{H}^q_\infty(\mathsf{PD})$ of PD by

$$\mathbf{GP}_{\mathsf{PD}}(q) = 2^{-\mathbf{H}^q_\infty(\mathsf{PD})} = \max_{(i_1,g_1),\ldots,(i_q,g_q)} \Pr\left[\, \exists\, j \ :\ \mathbf{P}[i_j] = g_j \ \ ::\ \ \mathbf{P} \leftarrow\!\!\$\ \mathsf{PD} \,\right] .$$

These definitions of guessing probability and min-entropy for q guesses generalize the ones of [42], which correspond to the case $u = 1$ of the above. Now, the relation with the game-based formulation is that $\mathbf{GP}_{\mathsf{PD}}(q) = \max_A \mathbf{Adv}^{\mathrm{pg}}_{\mathsf{PD},u}(A)$, where the maximum is over all adversaries A that make q Test queries.

Note that $\mathbf{GP}_{\mathsf{PD}}(1)$ is the probability of the most likely password in the vector, and $\mathbf{GP}_{\mathsf{PD}}(q) \le q \cdot \mathbf{GP}_{\mathsf{PD}}(1)$. In general, however, $\mathbf{GP}_{\mathsf{PD}}(q)$ can be quite a bit smaller than $q \cdot \mathbf{GP}_{\mathsf{PD}}(1)$, which is why we consider the more general definition and parameterization by q.

Suppose the entries of $\mathbf{P}$ are uniformly and independently distributed over a set of size N, subject to being distinct, and $1 \le q \le N$. Then $\mathbf{GP}_{\mathsf{PD}}(q) = q/N$. In general, however, there may not be a simple formula for the guessing probability.

Sometimes we are interested in a finer-grained parameterization of the guessing probability, which, beyond constraining the total number of guesses to some parameter q, also constrains the number of distinct passwords, and the number of distinct users, to parameters q_p, q_w, respectively. Formally we let

$$\mathbf{GP}_{\mathsf{PD}}(q, q_p, q_w) \max_{(i_1,g_1),\ldots,(i_q,g_q)} \Pr[\, \exists j \ :\ \mathbf{P}[i_j] = g_j \ \ ::\ \ \mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}\,]$$

where the maximum is taken over all $(i_1, g_1), \ldots, (i_q, g_q)$ such that $|\{\, g_j : 1 \le j \le q \,\}| \le q_p$ and $|\{\, i_j : 1 \le j \le q \,\}| \le q_w$. The relation with the game-based formulation is that $\mathbf{GP}_{\mathsf{PD}}(q, q_p, q_w) = \max_A \mathbf{Adv}^{\mathrm{pg}}_{\mathsf{PD},u}(A)$, where the maximum is over all adversaries A that make q TEST queries which involve at most q_p distinct passwords and at most q_w distinct users.

4 The tool: Symmetric encryption

We will be building FPBE schemes from symmetric encryption (SE) schemes and accordingly start with the latter. We give definitions that are novel, unifying the AE1 (AEAD) and AE2 notions [11] so that our results can easily apply to both. We give the definition of key-robustness we will assume. We define authenticity with corruptions and give two lemmas about it that we will use.

SE SYNTAX. A symmetric encryption scheme SE specifies a key length $\mathsf{SE.kl} \in \mathbb{N}$, nonce space $\mathsf{SE.NS}$, message space $\mathsf{SE.MS}$, and associated data (header) space $\mathsf{SE.AS}$. These spaces are assumed to be length-closed. Deterministic encryption algorithm $\mathsf{SE.Enc} : \{0,1\}^{\mathsf{SE.kl}} \times \mathsf{SE.NS} \times \mathsf{SE.MS} \times \mathsf{SE.AS} \to \{0,1\}^*$ returns a ciphertext $C \leftarrow \mathsf{SE.Enc}(K, N, M, A)$ that is a string of length $\mathsf{SE.cl}(|M|) \ge |M|$, where $\mathsf{SE.cl} : \mathbb{N} \to \mathbb{N}$ is the ciphertext-length function. Deterministic decryption algorithm $\mathsf{SE.Dec} : \{0,1\}^{\mathsf{SE.kl}} \times \mathsf{SE.NIS} \times \{0,1\}^* \times \mathsf{SE.AS} \to \mathsf{SE.MS} \cup \{\perp\}$ returns an output $M \leftarrow \mathsf{SE.Dec}(K, I, C, A)$ that is either a string in $\mathsf{SE.MS}$ or is $\perp$, where $\mathsf{SE.NIS}$ is the nonce-information space. Decryption correctness requires that $\mathsf{SE.Dec}(K, \mathsf{SE.NI}(N), \mathsf{SE.Enc}(K, N, M, A), A) = M$ for all $K \in \{0,1\}^{\mathsf{SE.kl}}$, $N \in \mathsf{SE.NS}$, $M \in \mathsf{SE.MS}$ and $A \in \mathsf{SE.AS}$, where $\mathsf{SE.NI} : \mathsf{SE.NS} \to \mathsf{SE.NIS}$ is the nonce-information function.

The purpose of nonce-information $(\mathsf{SE.NI}, \mathsf{SE.NIS})$ is to allow us to recover the NBE1 [39] and NBE2 [11] syntaxes as special cases, as follows. When $\mathsf{SE.NI}(N) = N$ and $\mathsf{SE.NIS} = \mathsf{SE.NS}$, the decryption algorithm is getting the nonce as input, which means we have the NBE1 syntax. When $\mathsf{SE.NI}(N) = \varepsilon$ and $\mathsf{SE.NIS} = \{\varepsilon\}$, the decryption algorithm gets no information about the nonce, and we have the NBE2 syntax. Our definition allows us to unify the two and give results that apply to both. More generally it allows us to consider decryption having partial information about the nonce.

SECURITY GAMES AND ADVERSARY CLASSES. There are two levels of security. The basic one requires that an encryption nonce not be reused by a particular user. The advanced one is nonce-misuse resistance, which drops this condition. We want our definitions and results to cover both in as compact and unified a way as possible. For this we follow [11] by giving a single game per security goal and then seeing basic and advanced security as restricting the adversary to an appropriate class, either $\mathcal{A}_{\mathrm{b}}$ (basic) or $\mathcal{A}_{\mathrm{a}}$ (advanced).

The goals (games) we consider are privacy (IND\$), authenticity (AUTH) and joint privacy+authenticity (AE). For each, there is basic and advanced security. Known results [11, 16] say that IND\$+AUTH is equivalent to AE (a scheme meets both IND\$ and AUTH iff it meets AE) for both basic and advanced security, and this is true even when AUTH is restricted to adversaries that are *sequential*, meaning make their VERIFY queries after their ENC queries. We let $\mathcal{A}_{\mathrm{seq}}$ be the class of sequential adversaries.

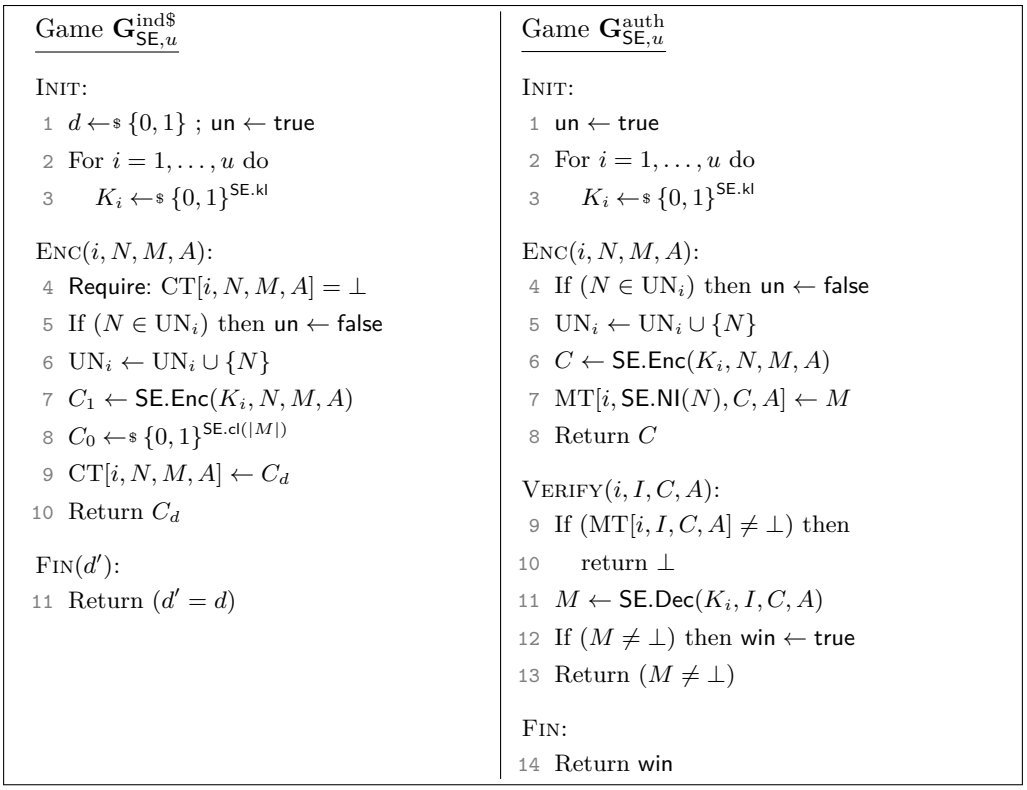

Figure 4: Games defining IND$ (left) and AUTH (right) security for symmetric encryption scheme SE over u users.

Games will use a flag un, for "unique nonce," that begins true. An adversary A is in the class $\mathcal{A}_{\mathrm{b}}$ if its execution with the game never sets un to false. $\mathcal{A}_{\mathrm{a}}$ is simply the class of all adversaries, meaning ones setting un to false are included. Games will at various points assert $\mathsf{Require}$: some condition, which means that all adversaries must obey this condition. This will be used to rule out trivial wins. We now proceed to the particular definitions.

SE privacy. This is defined via game $\mathbf{G}^{\mathrm{ind\$}}_{\mathsf{SE},u}$ in the left panel of Figure 4, where u is the number of users. (This is the multi-user setting.) If A is an adversary, we let $\mathbf{Adv}^{\mathrm{ind\$}}_{\mathsf{SE},u}(A) = 2\Pr[\mathbf{G}^{\mathrm{ind\$}}_{\mathsf{SE},u}(A)] - 1$ be its advantage.

SE authenticity. This is defined via game $\mathbf{G}^{\mathrm{auth}}_{\mathsf{SE},u}$ in the right panel of Figure 4, where u is the number of users. If A is an adversary, we let $\mathbf{Adv}^{\mathrm{auth}}_{\mathsf{SE},u}(A) = \Pr[\mathbf{G}^{\mathrm{auth}}_{\mathsf{SE},u}(A)]$ be its advantage.

Authenticity under corruptions. This is an extended form of authenticity defined via game $\mathbf{G}^{\mathrm{auth\text{-}c}}_{\mathsf{SE},u}$ of Figure 5, where u is the number of users. The new element is the Expose oracle that allows the adversary to obtain they key of a user i. We let $\mathbf{Adv}^{\mathrm{auth\text{-}c}}_{\mathsf{SE},u}(A) = \Pr[\mathbf{G}^{\mathrm{auth\text{-}c}}_{\mathsf{SE},u}(A)]$ be the advantage of adversary A.

We consider authenticity under corruptions because we will use it in the proof of Theorem 6.3. However, the following lemmas say that it is implied by regular authenticity and thus is not an additional assumption on SE. The first lemma, which gives up a factor of the number of users u, is implied by [25]. For completeness we give a proof in Appendix A.

Game $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},u}$

VERIFY(i, I, C, A):
1 If $((\text{MT}[i, I, C, A] \neq \bot)$ or $i \in \text{EU})$ then return $\bot$
2 $M \leftarrow \mathsf{SE.Dec}(K_i, I, C, A)$
3 If $(M \neq \bot)$ then $\mathsf{win} \leftarrow \mathsf{true}$
4 Return $(M \neq \bot)$

EXPOSE(i):
5 $\text{EU} \leftarrow \text{EU} \cup \{i\}$
6 Return K_i

Figure 5: Game defining authenticity under corruptions for symmetric encryption scheme SE over u users. The procedures INIT, ENC, FIN are as in the right panel of Figure 4.

Game $\mathbf{G}^{\text{krob\$}}_{\mathsf{SE},q,\gamma}$

INIT:
1 For $i = 1, \ldots, q$ do $K_i \xleftarrow{\$} \{0,1\}^{\mathsf{SE.kl}}$
2 Return $K_1, \ldots, K_q$

FIN(I, C, A):
3 For $i = 1, \ldots, q$ do $d_i \leftarrow (\mathsf{SE.Dec}(K_i, I, C, A) \neq \bot)$
4 Return $(\exists\, S \subseteq [1..q] : |S| = \gamma \wedge (\forall\, i \in S : d_i = \mathsf{true}))$

Figure 6: Game defining γ-way key-robustness for q keys for SE scheme SE.

Lemma 4.1 *Let SE be a symmetric encryption scheme and $u \geq 1$ a number of users. Let $y \in \{b, a\}$. Suppose $A_{\text{auth-c}} \in \mathcal{A}_y$ is an adversary making q_e ENC queries and q_v VERIFY queries per user in the $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},u}$ game. Then we can construct an adversary $A_{\text{auth}} \in \mathcal{A}_y$ such that*

$$\mathbf{Adv}^{\text{auth-c}}_{\mathsf{SE},u}(A_{\text{auth-c}}) \leq u \cdot \mathbf{Adv}^{\text{auth}}_{\mathsf{SE},1}(A_{\text{auth}}) \; . \tag{1}$$

Adversary A_{auth} makes q_e ENC and q_v VERIFY queries. The running time of A_{auth} is close to that of $A_{\text{auth-c}}$. If A_{auth} is sequential, so is $A_{\text{auth-c}}$.

Our next lemma, which is novel, shows that the factor u blowup above can be reduced to a constant in the absence of encryption queries. We will exploit this for Theorem 6.3. The proof is in Appendix A.

Lemma 4.2 *Let SE be a symmetric encryption scheme and $u \geq 1$ a number of users. Let $y \in \{b, a\}$. Suppose $A_{\text{auth-c}} \in \mathcal{A}_y$ is an adversary making q_v VERIFY queries per user, and no ENC queries, in the $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},u}$ game. Then we can construct an adversary $A_{\text{auth}} \in \mathcal{A}_y$ such that*

$$\mathbf{Adv}^{\text{auth-c}}_{\mathsf{SE},u}(A_{\text{auth-c}}) \leq 2 \cdot \mathbf{Adv}^{\text{auth}}_{\mathsf{SE},u}(A_{\text{auth}}) \; . \tag{2}$$

Adversary A_{auth} makes q_v VERIFY queries and no ENC queries. The running time of A_{auth} is close to that of $A_{\text{auth-c}}$. If A_{auth} is sequential, so is $A_{\text{auth-c}}$.

KEY-ROBUSTNESS. Theorem 6.3 will also assume key-robustness of a symmetric encryption scheme SE. This is defined via game $\mathbf{G}^{\text{krob\$}}_{\mathsf{SE},q,\gamma}$ of Figure 6 associated to scheme SE, number of keys q and size γ of the target collision. If A is an adversary, we let $\mathbf{Adv}^{\text{krob\$}}_{\mathsf{SE},q,\gamma}(A) = \Pr[\mathbf{G}^{\text{krob\$}}_{\mathsf{SE},q,\gamma}(A)]$ be its advantage. Security for $\gamma = 2$ implies it for higher γ, but we directly consider the latter because it

arises in partitioning-oracle attacks [29] and can be proved with better bounds [7]. The "$" in the notation indicates the random choice of keys at line 1. This choice makes our notion weaker than others in the literature [1, 2, 7, 22, 24], but this makes our results stronger because they assume a key-robust scheme and the less that is assumed, the better.

5 The goal: Flexible password-based encryption

We give formal definitions for the FPBE primitive that we introduce. We define both privacy and authenticity, as well as joint authenticated encryption (PAE). In Appendix B, we complete the proof that PAE for FPBE is equivalent to privacy+authenticity. While PAE is the overarching security goal for FPBE, considering privacy and authenticity separately in turn results in more straightforward theorem statements and proofs.

FPBE SYNTAX. A scheme FPBE specifies the following objects and algorithms. The key space is $\mathsf{FPBE.KS} = \{0,1\}^*$, meaning any string, representing a password and thus denoted P, can function as the key. We introduce a salt space, $\mathsf{FPBE.SS}$, and as in SE, use nonce, associated data (header), and message spaces. These spaces are assumed to be length-closed. Deterministic encryption algorithm $\mathsf{FPBE.Enc} : \mathsf{FPBE.KS} \times \mathsf{FPBE.SS} \times \mathsf{FPBE.NS} \times \mathsf{FPBE.MS} \times \mathsf{FPBE.AS} \to \{0,1\}^*$ returns a ciphertext $C \leftarrow \mathsf{FPBE.Enc}(P, S, N, M, A)$ that is a string of length $\mathsf{FPBE.cl}(|M|) \geq |M|$. Deterministic decryption algorithm $\mathsf{FPBE.Dec} : \mathsf{FPBE.KS} \times \mathsf{FPBE.SS} \times \mathsf{FPBE.NIS} \times \{0,1\}^* \times \mathsf{FPBE.AS} \to \mathsf{FPBE.MS} \cup \{\perp\}$ returns an output $M \leftarrow \mathsf{FPBE.Dec}(P, S, I, C, A)$ that is either a string in $\mathsf{FPBE.MS}$ or is $\perp$. Decryption correctness requires that $\mathsf{FPBE.Dec}(P, S, \mathsf{FPBE.NI}(N), \mathsf{FPBE.Enc}(P, S, N, M, A), A) = M$ for all $P \in \mathsf{FPBE.KS}$, $S \in \mathsf{FPBE.SS}$, $N \in \mathsf{FPBE.NS}$, $M \in \mathsf{FPBE.MS}$ and $A \in \mathsf{FPBE.AS}$, where $\mathsf{FPBE.NI} : \mathsf{FPBE.NS} \to \mathsf{FPBE.NIS}$ is the nonce-information function.

SALTS VERSUS NONCES. One may ask why have both a salt and a nonce. In particular, if there is a nonce, why do we also need a salt? The purpose of a salt in password-based encryption is to preclude pre-computation in brute-force attacks, forcing the attacker to do dictionary-size, per-user online work. Nonces will not accomplish this since they can be predictable and the same for different users, so we retain the salt. Then one may ask, why the nonce? One benefit is a shorter amortized ciphertext length, leading to reduced storage cost in cloud encryption. Suppose q messages $M_1, \ldots, M_q$ are encrypted and the ciphertexts $C_1, \ldots, C_q$ are stored on the server. First consider encryption under TPBE (traditional PBE, which has a per-message random salt but no nonce). The salts have to be stored with the ciphertexts to allow decryption. If sl is the salt length, the storage overhead is $\mathrm{sl} \cdot q$. Now consider using FPBE, where the user picks one random salt S and encrypts M_i with S and, as nonce, $\langle i \rangle_c$, a c-bit encoding of the integer i. Now one stores the single salt, and the per-message nonce, so the storage overhead is $\mathrm{sl} + qc$. The latter is lower than $\mathrm{sl} \cdot q$ because c can be small (say, 16 bits for $q \leq 2^{16}$) while salts need to resist collisions so need to be 128 bits or more. In fact one can do even better. Seeing the nonce as given by the index i of the ciphertext in the list $C_1, \ldots, C_q$, only the single salt needs to be stored, for storage overhead sl.

SECURITY GAMES AND ADVERSARY CLASSES. We aim to bring PBE in line with modern symmetric encryption by treating both basic and advanced security. As above, we give a single game per security goal and then restrict to adversary classes that we continue to denote $\mathcal{A}_{\mathrm{b}}$ (basic) or $\mathcal{A}_{\mathrm{a}}$ (advanced) but are redefined for the password-based case. Again, the goals we consider are privacy (PIND$), authenticity (PAUTH) and joint privacy+authenticity (PAE). Games will again use a flag un, for "unique nonce," and adversary A is in the class $\mathcal{A}_{\mathrm{b}}$ if its execution with the game never sets un to false. $\mathcal{A}_{\mathrm{a}}$ is simply the class of all adversaries. We now proceed to the particular definitions.

Game $\mathbf{G}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}$

INIT:
1 $d \leftarrow\!\!\$\ \{0,1\}$; $\mathsf{un} \leftarrow \mathsf{true}$
2 $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$ // u-vector of passwords

ENC(i, N, M, A):
3 Require: $s(i) \neq 0$
4 Require: $\mathrm{CT}[i, s(i), N, M, A] = \perp$
5 If $(N \in \mathrm{UN}_{i,s(i)})$ then $\mathsf{un} \leftarrow \mathsf{false}$
6 $\mathrm{UN}_{i,s(i)} \leftarrow \mathrm{UN}_{i,s(i)} \cup \{N\}$
7 $C_1 \leftarrow \mathsf{FPBE.Enc}(\mathbf{P}[i], S_i, N, M, A)$
8 $C_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{FPBE.cl}(|M|)}$
9 $\mathrm{CT}[i, s(i), N, M, A] \leftarrow C_d$
10 Return C_d

SALT(i):
11 $s(i) \leftarrow s(i) + 1$; $S_i \leftarrow\!\!\$\ \mathsf{FPBE.SS}$
12 Return S_i

FIN(d'):
13 Return $(d' = d)$

Game $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$

INIT:
1 $\mathsf{un} \leftarrow \mathsf{true}$
2 $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$ // u-vector of passwords

ENC(i, N, M, A):
3 Require: $s(i) \neq 0$
4 If $(N \in \mathrm{UN}_{i,s(i)})$ then $\mathsf{un} \leftarrow \mathsf{false}$
5 $\mathrm{UN}_{i,s(i)} \leftarrow \mathrm{UN}_{i,s(i)} \cup \{N\}$
6 $C \leftarrow \mathsf{FPBE.Enc}(\mathbf{P}[i], S_i, N, M, A)$
7 $\mathrm{MT}[i, S_i, \mathsf{FPBE.NI}(N), C, A] \leftarrow M$
8 Return C

VERIFY(i, S, I, C, A):
9 If $(\mathrm{MT}[i, S, I, C, A] \neq \perp)$ then
10 return $\perp$
11 $M \leftarrow \mathsf{FPBE.Dec}(\mathbf{P}[i], S, I, C, A)$
12 If $(M \neq \perp)$ then $\mathsf{win} \leftarrow \mathsf{true}$
13 Return $(M \neq \perp)$

SALT(i):
14 $s(i) \leftarrow s(i) + 1$; $S_i \leftarrow\!\!\$\ \mathsf{FPBE.SS}$
15 Return S_i

FIN:
16 Return win

Figure 7: Games defining PIND\$ (left) and PAUTH (right) security for FPBE scheme FPBE over u users.

FPBE PRIVACY. Let PD be a distribution over passwords, as above, for u users. Then privacy is defined by game $\mathbf{G}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}$ of Figure 7. If A is an adversary, we let $\mathbf{Adv}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) = 2\Pr[\mathbf{G}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A)] - 1$ be its advantage.

FPBE AUTHENTICITY. Let PD be a distribution over u-vectors of passwords. Authenticity is defined by game $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$ on the right of Figure 7. If A is an adversary, we let $\mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A) = \Pr[\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A)]$ be its advantage.

In this setting, we call an adversary *sequential* if all its VERIFY queries come after all its SALT and ENC queries. We continue to denote the class of such adversaries as $\mathcal{A}_{\text{seq}}$. Theorem 5.1 allows us to restrict attention to sequential adversaries when proving PAUTH.

FPBE AUTHENTICATED ENCRYPTION. For a password distribution PD over u users, authenticated encryption (PAE) is defined by game $\mathbf{G}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}$ of Figure 8. For an FPBE scheme, the advantage of an adversary A is $\mathbf{Adv}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}(A) = 2\Pr[\mathbf{G}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}(A)] - 1$.

Recall that results from [11, 16] say that if a standard symmetric encryption scheme SE is both IND\$-secure and AUTH-secure then it is also AE-secure, and moreover this is true even if AUTH is assumed only for sequential adversaries. In the following theorem we give the analogue of this result for FPBE. Namely, the theorem says that if FPBE is both PIND\$-secure and PAUTH-secure, then

Game $\mathbf{G}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}$

INIT:
1 $d \leftarrow\!\!\$\ \{0,1\}$; $\mathsf{un} \leftarrow \mathsf{true}$
2 $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$ // u-vector of passwords

ENC(i, N, M, A):
3 Require: $s(i) \neq 0$
4 Require: $\mathrm{CT}[i, s(i), N, M, A] = \perp$
5 If $(N \in \mathrm{UN}_{i,s(i)})$ then $\mathsf{un} \leftarrow \mathsf{false}$
6 $\mathrm{UN}_{i,s(i)} \leftarrow \mathrm{UN}_{i,s(i)} \cup \{N\}$
7 $C_1 \leftarrow \mathsf{FPBE.Enc}(\mathbf{P}[i], S_i, N, M, A)$
8 $C_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{FPBE.cl}(|M|)}$
9 $\mathrm{CT}[i, s(i), N, M, A] \leftarrow C_d$
10 $\mathrm{MT}[i, S_i, \mathsf{FPBE.NI}(N), C_d, A] \leftarrow M$
11 Return C_d

DEC(i, S, I, C, A):
12 If $(\mathrm{MT}[i, S, I, C, A] \neq \perp)$ then
13 return $\mathrm{MT}[i, S, I, C, A]$
14 If $(d = 0)$ then return $\perp$
15 $M \leftarrow \mathsf{FPBE.Dec}(\mathbf{P}[i], S, I, C, A)$
16 Return M

SALT(i):
17 $s(i) \leftarrow s(i) + 1$; $S_i \leftarrow\!\!\$\ \mathsf{FPBE.SS}$
18 Return S_i

FIN(d'):
19 Return $(d' = d)$

Figure 8: Game defining PAE security for FPBE over u users.

---

it is also PAE-secure, and this is true even if PAUTH is assumed only for sequential adversaries. This result allows us, in later analyses of PAUTH, to restrict attention to sequential adversaries, and thereby simplify analyses and proofs. The proof of the following, which is in Appendix B, follows the proof of [11].

Theorem 5.1 *Let* FPBE *be an FPBE scheme over $u \geq 1$ users, password distribution* PD*, and salt length* $\mathrm{sl} \geq 1$*, with access to a random oracle* $\mathrm{H} : \mathcal{D} \to \mathcal{R}$*. Let $y \in \{b, a\}$. Suppose $A \in \mathcal{A}_y$ is an adversary making q_s* SALT *queries, q_e* ENC *queries, q_d* DEC *queries and q_h* H *queries in the $\mathbf{G}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}$ game in the ROM. Then we can construct adversaries $A_{\text{pind\$}} \in \mathcal{A}_y$ and $A_{\text{pauth}} \in \mathcal{A}_y \cap \mathcal{A}_{seq}$ in the ROM such that*

$$\mathbf{Adv}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \leq \mathbf{Adv}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\text{pind\$}}) + 2 \cdot \mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\text{pauth}}) \,. \tag{3}$$

The running times of $A_{\text{pind\$}}, A_{\text{pauth}}$ are close to that of A. $A_{\text{pind\$}}$ makes q_s, q_e SALT, ENC *queries, respectively while A_{pauth} makes q_s, q_e, q_d* SALT, ENC, VERIFY *queries, respectively. Both $A_{\text{pind\$}}$ and A_{pauth} make q_h* H *queries.*

Theorem 5.1 allows us to prove PAE (the end goal of FPBE) by proving PIND\$ and PAUTH independently, which simplifies proofs. Most importantly, it demonstrates the utility of defining sequential adversaries. Crucially, we make no restriction on whether the PAE adversary A is sequential or not; A can be non-sequential. Despite this, the constructed adversary A_{pauth} always is sequential. This means we need to prove PAUTH only for sequential adversaries, a simplification we take advantage of in Theorems 6.2, 6.3.

The above theorem is stated in the random oracle model because our later results will be; however the statement holds in the standard model as well. We note that the other direction, PAE $\Rightarrow$ PIND\$ + PAUTH also holds, and is a simple proof that we omit; an adversary breaking PIND\$ or PAUTH can already break PAE with little change beyond notation.

Algorithm $\mathsf{FPBE.Enc}(P, S, N, M, A)$:	Algorithm $\mathsf{FPBE.Dec}(P, S, I, C, A)$:
1 $K \leftarrow \mathsf{F}(P, S)$	4 $K \leftarrow \mathsf{F}(P, S)$
2 $C \leftarrow \mathsf{SE.Enc}(K, N, M, A)$	5 $M \leftarrow \mathsf{SE.Dec}(K, I, C, A)$
3 Return C	6 Return M

Figure 9: Encryption and decryption algorithms of the scheme $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ constructed from symmetric encryption scheme SE and PBKDF F via the **DtE** transform.

6 Security of the DtE scheme

DtE TRANSFORM. We specify a transform **DtE** that, given a symmetric encryption scheme SE and a function $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$, returns a password-based scheme $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$. The name **DtE** stands for "derive-then-encrypt." The encryption and decryption algorithms of FPBE are shown in Figure 9. The salt space is $\mathsf{FPBE.SS} = \{0,1\}^{\mathrm{sl}}$. The message, nonce and header spaces are those of SE, as is the nonce-information algorithm. We refer to F as the password-based key-derivation function (PBKDF). Choices include PBKDF2 [26], BCRYPT [38], SCRYPT [3,4,35] or Argon2 [15]. The results in this section model F as a random oracle [13], but some of the overlying results (Theorems 7.1, 7.2) are under a standard-model assumption on F.

PRIVACY OF **DtE**. The following theorem says that if the base scheme SE is IND$-secure and the password distribution PD has low guessing probability then the constructed scheme $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ is PIND$-secure when F is modeled as a random oracle.

Theorem 6.1 *Let SE be a symmetric encryption scheme and let PBKDF $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ be defined by $\mathsf{F}[\mathrm{H}](P, S) = \mathrm{H}(P, S)$, where we model $\mathrm{H} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ as a random oracle. Let $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$. Let PD be a password distribution for $u \geq 1$ users. Let $y \in \{b, a\}$. Suppose $A \in \mathcal{A}_y$ is an adversary making q_s, q_e, q_h queries to its* SALT, ENC, H *oracles, respectively, in the $\mathbf{G}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}$ game in the ROM. Then we can construct an adversary $A_{\mathsf{SE}} \in \mathcal{A}_y$ such that*

$$\mathbf{Adv}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \leq \mathbf{GP}_{\mathsf{PD}}(q_h) + \mathbf{Adv}^{\mathrm{ind\$}}_{\mathsf{SE},q_s}(A_{\mathsf{SE}}) + \frac{q_s(q_s-1)}{2^{\mathrm{sl}}} \ . \tag{4}$$

Adversary A_{SE} makes q_e ENC *queries and has running time close to that of A.*

The proof of Theorem 6.1 is obtained by combining Theorems 7.1 and 7.3, and is given at the end of Section 7. Note that in Theorem 6.1 the assumed IND$ security of SE is for a number of users that is equal to the number q_s of SALT queries of A, with A_{SE} making q_e ENC queries across all these users.

AUTHENTICITY OF **DtE**. Our first authenticity theorem says that if the base scheme SE is AUTH-secure and PD has low guessing probability, then the derived scheme $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ is PAUTH-secure when F is modeled as a random oracle. The statement below uses the extended parameterization of the guessing probability; in Section 1 we had discussed only the q parameter. We assume $A \in \mathcal{A}_{\mathrm{seq}}$, meaning A is sequential, which is justified by Theorem 5.1.

Theorem 6.2 *Let SE be a symmetric encryption scheme and let PBKDF $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ be defined by $\mathsf{F}[\mathrm{H}](P, S) = \mathrm{H}(P, S)$, where we model $\mathrm{H} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ as a random oracle. Let $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$. Let PD be a password distribution for $u \geq 1$ users. Let $y \in \{b, a\}$. Suppose $A \in \mathcal{A}_y \cap \mathcal{A}_{seq}$ is a sequential adversary making q_s, q_v, q_h queries to its* SALT,

VERIFY, H *oracles, respectively, in the* $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$ *game in the ROM. Then we can construct an adversary* $A_{\mathsf{SE}} \in \mathcal{A}_y \cap \mathcal{A}_{seq}$ *such that*

$$\mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \le \mathbf{GP}_{\mathsf{PD}}(q, q_h, q_w) + \mathbf{Adv}^{\text{auth}}_{\mathsf{SE},q_s+q_v}(A_{\mathsf{SE}}) + \frac{q_s(q_s-1)}{2^{\text{sl}}} \ , \tag{5}$$

where $q_w = \min(q_s + q_v, u)$ *and* $q = \mathsf{zt}(q_s) \cdot q_h + \min(q_v, u) \cdot q_h$*. Adversary* A_{SE} *makes the same number of* ENC *and* VERIFY *queries as* A*, and has running time close to that of* A*.*

The proof of Theorem 6.2, given at the end of Section 7, is obtained by combining Theorems 7.2 and 7.3. We note that the security of FPBE over u users is based on the security of SE over $q_s + q_v$ users, corresponding to keys arising from salts in SALT or VERIFY queries.

BETTER AUTHENTICITY FROM KEY-ROBUSTNESS. Our second authenticity result strengthens the first by showing that if the base scheme additionally is key-robust then the strength of passwords required to guarantee authenticity is reduced. This shows up in the guessing probability term of the bound. The authenticity-under-corruptions term $\mathbf{Adv}^{\text{auth-c}}_{\mathsf{SE},q_h}(A_{\text{auth-c}})$ below can be tightly bounded using standard authenticity via Lemma 4.2, exploiting the fact that the constructed adversary $A_{\text{auth-c}}$ makes no ENC queries. Recall that $\mathsf{zt}(q_s)$ is 0 if $q_s = 0$ and is 1 otherwise. As before we assume $A \in \mathcal{A}_{\text{seq}}$, meaning A below is sequential, which is justified by Theorem 5.1.

Theorem 6.3 *Let* SE *be a symmetric encryption scheme and let PBKDF* $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\text{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *be defined by* $\mathsf{F}[\mathrm{H}](P,S) = \mathrm{H}(P,S)$*, where we model* $\mathrm{H} : \{0,1\}^* \times \{0,1\}^{\text{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *as a random oracle. Let* $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE},\mathsf{F}]$*. Let* PD *be a password distribution for* $u \ge 1$ *users. Let* $\gamma \ge 2$ *be the key-robustness width parameter. Let* $y \in \{b, a\}$*. Suppose* $A \in \mathcal{A}_y \cap \mathcal{A}_{seq}$ *is a sequential adversary making* q_s, q_e, q_v, q_h *queries to its* SALT, ENC, VERIFY, H *oracles, respectively, in the* $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$ *game in the ROM. Then we can construct an adversary* $A_{\text{krob\$}}$*, and adversaries* $A_{\text{auth}}, A_{\text{auth-c}} \in \mathcal{A}_y \cap \mathcal{A}_{seq}$*, such that*

$$\begin{aligned}\mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A) &\le \mathbf{GP}_{\mathsf{PD}}(\mathsf{zt}(q_s) \cdot q_h + (\gamma-1) \cdot q_v) + q_s(q_s-1) \cdot 2^{-\text{sl}-1} \\ &\quad + \mathbf{Adv}^{\text{krob\$}}_{\mathsf{SE},q_h,\gamma}(A_{\text{krob\$}}) + \mathbf{Adv}^{\text{auth}}_{\mathsf{SE},q_s+q_v}(A_{\text{auth}}) + \mathbf{Adv}^{\text{auth-c}}_{\mathsf{SE},q_h}(A_{\text{auth-c}}) \ . \end{aligned} \tag{6}$$

Adversaries $A_{\text{auth}}, A_{\text{auth-c}}$ *make* q_v, q_h VERIFY *queries, respectively.* A_{auth} *makes* q_e ENC *queries, but* $A_{\text{auth-c}}$ *makes none. The running times of* $A_{\text{auth}}, A_{\text{auth-c}}, A_{\text{krob\$}}$ *are close to that of* A*.*

The simplest choice for parameter γ above is $\gamma = 2$, which is what we assumed in Section 1 and Figure 2. We are more general in Theorem 6.3 because there are schemes SE for which slightly increasing γ, even from 2 to 3, will significantly reduce $\mathbf{Adv}^{\text{krob\$}}_{\mathsf{SE},q_h,\gamma}(A_{\text{krob\$}})$ [7], and one may benefit from this tradeoff. We prove Theorem 6.3 in Appendix C.

AUTHENTICATED ENCRYPTION FROM **DtE**. Given the above theorems on the privacy and authenticity of **DtE**, and Theorem 5.1 showing the equivalence of PAE and privacy+authenticity, we can consider the impact of key-robustness on PAE overall. The first theorem below combines Theorems 6.1 and 6.2, along with Theorem 5.1. Note that the given adversary A is *not* restricted to be sequential.

Theorem 6.4 *Let* SE *be a symmetric encryption scheme and let PBKDF* $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\text{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *be defined by* $\mathsf{F}[\mathrm{H}](P,S) = \mathrm{H}(P,S)$*, where we model* $\mathrm{H} : \{0,1\}^* \times \{0,1\}^{\text{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *as a random oracle. Let* $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE},\mathsf{F}]$*. Let* PD *be a password distribution for* $u \ge 1$ *users. Let* $y \in \{b, a\}$*. Suppose* $A \in \mathcal{A}_y$ *is an adversary making* q_s, q_e, q_d, q_h *queries to its* SALT, ENC, DEC, H

oracles, respectively, in the $\mathbf{G}^{\mathrm{pae}}_{\mathsf{FPBE},\mathsf{PD},u}$ *game in the ROM. Then we can construct adversaries* $A_{\mathrm{ind\$}} \in \mathcal{A}_y$ *and* $A_{\mathrm{auth}} \in \mathcal{A}_y \cap \mathcal{A}_{seq}$ *such that*

$$\mathbf{Adv}^{\mathrm{pae}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \le \mathbf{GP}_{\mathsf{PD}}(q_h) + 2 \cdot \mathbf{GP}_{\mathsf{PD}}(q, q_h, q_w) + \frac{3q_s(q_s-1)}{2^{\mathrm{sl}}} + \mathbf{Adv}^{\mathrm{ind\$}}_{\mathsf{SE},q_s}(A_{\mathrm{ind\$}}) + 2 \cdot \mathbf{Adv}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_d}(A_{\mathrm{auth}}) \ , \tag{7}$$

where $q_w = \min(q_s + q_d, u)$ *and* $q = \mathsf{zt}(q_s) \cdot q_h + \min(q_d, u) \cdot q_h$*. Adversary* $A_{\mathrm{ind\$}}$ *makes* q_e ENC *queries and adversary* A_{auth} *makes* q_e, q_d ENC, VERIFY *queries. The running times of* $A_{\mathrm{ind\$}}, A_{\mathrm{auth}}$ *are close to that of* A*.*

Our next theorem reconsiders PAE using the authenticity bound in Theorem 6.3 rather than that in Theorem 6.2. Again the given adversary A is not restricted to be sequential. We see that in our goal to minimize the guessing probability parameter in PAE, the most influential term is $2 \cdot \mathbf{GP}_{\mathsf{PD}}(q)$ for a particular q that arises from authenticity. PAE thus maintains the benefits of key-robustness as discussed in Section 1 and Figure 2.

Theorem 6.5 *Let* SE *be a symmetric encryption scheme and let PBKDF* $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *be defined by* $\mathsf{F}[\mathrm{H}](P,S) = \mathrm{H}(P,S)$*, where we model* $\mathrm{H} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *as a random oracle. Let* $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE},\mathsf{F}]$*. Let* PD *be a password distribution for* $u \ge 1$ *users. Let* $\gamma \ge 2$ *be the key-robustness width parameter. Let* $y \in \{b, a\}$*. Suppose* $A \in \mathcal{A}_y$ *is an adversary making* q_s, q_e, q_d, q_h *queries to its* SALT, ENC, DEC, H *oracles, respectively, in the* $\mathbf{G}^{\mathrm{pae}}_{\mathsf{FPBE},\mathsf{PD},u}$ *game in the ROM. Then we can construct adversaries* $A_{\mathrm{krob\$}}$*,* $A_{\mathrm{ind\$}} \in \mathcal{A}_y$ *and* $A_{\mathrm{auth}}, A_{\mathrm{auth\text{-}c}} \in \mathcal{A}_y \cap \mathcal{A}_{seq}$ *such that*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{pae}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \le{}& \mathbf{GP}_{\mathsf{PD}}(q_h) + 2 \cdot \mathbf{GP}_{\mathsf{PD}}(\mathsf{zt}(q_s) \cdot q_h + (\gamma-1) \cdot q_d) + \frac{q_s(q_s-1)}{2^{\mathrm{sl}-1}} \\ &+ \mathbf{Adv}^{\mathrm{ind\$}}_{\mathsf{SE},q_s}(A_{\mathrm{ind\$}}) + 2 \cdot \mathbf{Adv}^{\mathrm{krob\$}}_{\mathsf{SE},q_h,\gamma}(A_{\mathrm{krob\$}}) \\ &+ 2 \cdot \mathbf{Adv}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_d}(A_{\mathrm{auth}}) + 2 \cdot \mathbf{Adv}^{\mathrm{auth\text{-}c}}_{\mathsf{SE},q_h}(A_{\mathrm{auth\text{-}c}}) \ .\end{aligned} \tag{8}$$

Adversaries $A_{\mathrm{auth}}, A_{\mathrm{auth\text{-}c}}$ *make* q_d, q_h VERIFY *queries, respectively. Adversary* $A_{\mathrm{ind\$}}$ *makes* q_e ENC *queries, and* A_{auth} *makes* q_e ENC *queries, but* $A_{\mathrm{auth\text{-}c}}$ *makes none. Their running times, and that of* $A_{\mathrm{krob\$}}$*, are close to that of* A*.*

TIGHTNESS OF BOUNDS VIA ATTACKS. The bounds in Theorems 6.1, 6.2 and 6.3 all involve a term $\mathbf{GP}_{\mathsf{PD}}(q)$ or $\mathbf{GP}_{\mathsf{PD}}(q, q_h, q_w)$ for parameters q, q_w that vary across the results. Our quest to understand the strength of the password needed for the security of $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE},\mathsf{F}]$ comes down to the question of whether these parameters are optimal. In Section 8 , we assess this by consideration of attacks. Briefly, we find that they are indeed essentially optimal in all our theorems, in some cases due to the classical brute-force attack and in other cases due to partitioning-oracle attacks.

7 Proving DtE security via composition and PBKDFs

We give new definitions for password-based key-derivation functions (PBKDFs). Then we give composition theorems that show that if F meets our definition then $\mathbf{DtE}[\mathsf{SE},\mathsf{F}]$ retains both the privacy and authenticity of SE. We then analyze security, under our definition, of a PBKDF modeled as a random oracle, with particular attention to minimizing the number of password guessing queries used to bound adversary advantage. Putting all this together will yield Theorems 6.1 and 6.2 (of Section 6) as corollaries, avoiding ad hoc proofs of the same.

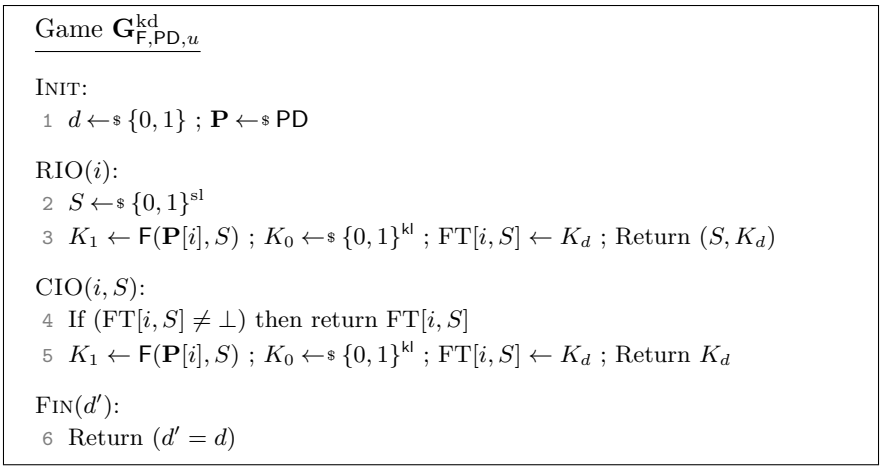

Figure 10: Game defining kd-security of PBKDF F relative to u-user password space PD.

PBKDF syntax. A PBKDF $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathsf{sl}} \to \{0,1\}^{\mathsf{kl}}$ takes a password P and an input S (the notation reflecting that in our usage it will be the salt) to deterministically return an output $\mathsf{F}(P,S)$. (In our usage, the derived symmetric key.) In the random oracle model, F will have oracle access to a random function $\mathrm{H} : \mathcal{D} \to \mathcal{R}$ where $\mathcal{D}, \mathcal{R}$ could depend on the scheme. In Theorems 6.1, 6.2 and 7.3, $\mathcal{D} = \{0,1\}^* \times \{0,1\}^{\mathsf{sl}}$ and $\mathcal{R} = \{0,1\}^{\mathsf{SE.kl}}$, where kl is the key length of the underlying scheme SE.

PBKDF security. Security of a PBKDF F is measured, not in isolation, but relative to a u-user password distribution PD from which passwords are drawn. The game, denoted $\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}$, is in Figure 10, and the kd-advantage of adversary A_{F} is $\mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) = 2\Pr[\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}})] - 1$. We refer to RIO, CIO as the random-input oracle and chosen-input oracle respectively. Oracle RIO is queried with just a user index i. The game picks a random input S and returns either $\mathsf{F}(\mathbf{P}[i], S)$ or a random string, depending on the challenge bit d. It also returns the input S. Oracle CIO is queried with both a user index and an input S (the chosen input) and then returns either $\mathsf{F}(\mathbf{P}[i], S)$ or a random string, depending on d. In the ROM, the game adds a procedure H for the random oracle.

Intuitively, kd-security is asking for prf-security in a multi-user setting in which the keys are passwords, and passwords of different users may be related. This can be seen as a form of security under related-key attack [9], correlated-input hash functions [23] or UCE [8]. Oracle CIO is the usual one for a prf-like setting, while RIO can be seen as representing weak-PRF security [34, 36].

A natural question is, isn't RIO redundant given CIO? Indeed, queries to the former can be simulated via queries to the latter. This means RIO can be dropped without a *qualitative* change in the kd notion, but *quantitatively* there is an important difference that is a key point of Theorem 7.3, namely that RIO queries are "cheaper" in the sense that the number of password guesses needed to bound adversary advantage is less for RIO queries than for CIO queries. Eventually, this translates to better proven quantitative security guarantees for privacy than for authenticity for FPBE.

We say that adversary A_{F} is *sequential* if it makes its CIO queries after its RIO queries. (That is, once the first CIO query has been made, no further RIO queries are allowed.) It will suffice to prove kd-security of F (as in Theorem 7.3) for sequential adversaries because that is all we need for our applications, as simplified by Theorem 5.1.

BRT [12] give a simulation-based definition of security for PBKDFs that is related to the

indifferentiability framework of [31]. We are giving a somewhat simpler and more direct version of their definition (no simulator) that can be used in both the standard and random-oracle models, and we are also introducing the distinction between CIO and RIO queries.

COMPOSITION THEOREMS. The benefit of abstracting the security of the PBKDF via kd-security is that we can see $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ as obtained by composing a PBKDF F with an SE scheme SE, and give modular security proofs for FPBE via composition theorems. In this vein, our first composition theorem says that if the base scheme SE is IND\$-secure and F is kd-secure relative to password distribution PD, then $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ is PIND\$-secure relative to PD. To facilitate the application to deriving Theorem 6.1, F is allowed access to a random oracle H that is provided in game $\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}$ and inherited in game $\mathbf{G}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}$.

Theorem 7.1 *Let SE be a symmetric encryption scheme. Let $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ be a PBKDF with access to a random oracle $\mathrm{H} : \mathcal{D} \to \mathcal{R}$. Let $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$. Let PD be a password distribution for $u \geq 1$ users. Let $y \in \{b, a\}$. Suppose $A \in \mathcal{A}_y$ is an adversary making q_s, q_e, q_h queries to its $\textsc{Salt}, \textsc{Enc}, \mathrm{H}$ oracles, respectively, in the $\mathbf{G}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}$ game in the ROM. Then we can construct adversaries $A_{\mathsf{SE}} \in \mathcal{A}_y$ and A_{F} such that*

$$\mathbf{Adv}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \leq \mathbf{Adv}^{\mathrm{ind\$}}_{\mathsf{SE},q_s}(A_{\mathsf{SE}}) + \mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \ . \tag{9}$$

Adversary A_{SE} makes q_e $\textsc{Enc}$ queries. Adversary A_{F} makes $q_s, 0, q_h$ queries to its $\mathrm{RIO}, \mathrm{CIO}, \mathrm{H}$ oracles, respectively. The running times of $A_{\mathsf{SE}}, A_{\mathsf{F}}$ are close to that of A.

As Eq. (9) indicates, we need IND\$ security of SE in the presence of q_s users. (To each user-salt pair, the PBKDF associates a fresh key for SE, effectively creating a fresh user for SE.) We note that the kd-security of F is needed only for RIO queries, not CIO queries. The proof follows the natural paradigm in which we move from the real game G_1 to a game G_2 in which the outputs of F are replaced by random keys. The assumed kd-security of F means that the adversary will not notice this move. The assumed IND\$ security of SE then allows us to move from G_2 to a game G_0 where ciphertexts are random strings. A full proof of Theorem 7.1 is in Appendix D.

Analogously, our second composition theorem says that if the base scheme SE is AUTH-secure and F is kd-secure relative to password distribution PD, then $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ is PAUTH-secure relative to PD. A novel element relative to Theorem 7.1 is that we now need kd-security in the presence of CIO queries. It suffices, below, to consider sequential A, because of Theorem 5.1.

Theorem 7.2 *Let SE be a symmetric encryption scheme. Let $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ be a PBKDF with access to a random oracle $\mathrm{H} : \mathcal{D} \to \mathcal{R}$. Let $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$. Let PD be a password distribution for $u \geq 1$ users. Let $y \in \{b, a\}$. Suppose $A \in \mathcal{A}_y \cap \mathcal{A}_{seq}$ is a sequential adversary making q_s, q_e, q_v, q_h queries to its $\textsc{Salt}, \textsc{Enc}, \textsc{Verify}, \mathrm{H}$ oracles, respectively, in the $\mathbf{G}^{\mathrm{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$ game in the ROM. Then we can construct adversaries $A_{\mathsf{SE}} \in \mathcal{A}_y \cap \mathcal{A}_{seq}$ and A_{F} such that*

$$\mathbf{Adv}^{\mathrm{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \leq \mathbf{Adv}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_v}(A_{\mathsf{SE}}) + \mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \ . \tag{10}$$

Adversary A_{SE} makes q_e, q_v queries to its $\textsc{Enc}, \textsc{Verify}$ oracles, respectively. Adversary A_{F} is sequential, making q_s, q_v, q_h queries to its $\mathrm{RIO}, \mathrm{CIO}, \mathrm{H}$ oracles, respectively. The running times of $A_{\mathsf{SE}}, A_{\mathsf{F}}$ are close to that of A.

As Eq. (10) indicates, we need AUTH security of SE in the presence of $q_s + q_v$ users, the extra q_v arising from $\textsc{Verify}$ queries with salts that were not results of $\textsc{Salt}$ queries. In its $\textsc{Verify}$ queries, A can choose the salt, which causes A_{F} to need to make CIO queries in order to respond to A's queries. Note that the constructed A_{F} is itself sequential, making its RIO queries before its CIO

queries, which allows us to use this in conjunction with Theorem 7.3. The proof of Theorem 7.2 follows the same paradigm as above, moving from the real game G_0 to a game G_1 in which the outputs of F are replaced by random keys. The assumed kd-security of F means that the adversary will not notice this move, and the assumed AUTH security of SE says the adversary is unlikely to win G_1. A full proof of Theorem 7.2 is in Appendix E.

KD-SECURITY OF H-PBKDF. H-PBKDF is the PBKDF $\mathsf{F}\colon \{0,1\}^* \times \{0,1\}^{\mathsf{sl}} \to \{0,1\}^{\mathsf{kl}}$ defined by $\mathsf{F}[\mathrm{H}](P,S) = \mathrm{H}(P,S)$ where $\mathrm{H}\colon \{0,1\}^* \times \{0,1\}^{\mathsf{sl}} \to \{0,1\}^{\mathsf{kl}}$ is a random oracle. We now want to study its kd-security. Qualitatively, Theorem 7.3 below says that F is kd-secure as long as the password distribution PD has high min-entropy and the input length sl is large enough. We discuss the quantitative interpretation after the theorem statement. Note that A_{F} below is assumed to be sequential, meaning it makes its CIO queries after its RIO queries. Recall that $\mathsf{zt}(q_r)$ is 0 if $q_r = 0$ and is 1 otherwise. The proof of Theorem 7.3 is in Appendix F.

Theorem 7.3 *Let PBKDF* $\mathsf{F}\colon \{0,1\}^* \times \{0,1\}^{\mathsf{sl}} \to \{0,1\}^{\mathsf{kl}}$ *be defined by* $\mathsf{F}[\mathrm{H}](P,S) = \mathrm{H}(P,S)$*, where we model* $\mathrm{H}\colon \{0,1\}^* \times \{0,1\}^{\mathsf{sl}} \to \{0,1\}^{\mathsf{kl}}$ *as a random oracle. Let* PD *be a password distribution for* $u \geq 1$ *users. Suppose* $A_{\mathsf{F}} \in \mathcal{A}_{seq}$ *is a sequential adversary making* q_r, q_c, q_h *queries to its* $\mathrm{RIO}, \mathrm{CIO}, \mathrm{H}$ *oracles, respectively, in the* $\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}$ *game in the ROM. Then*

$$\mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \leq \mathbf{GP}_{\mathsf{PD}}(q, q_h, q_w) + \frac{q_r(q_r-1)}{2^{\mathrm{sl}}}\ , \tag{11}$$

where $q_w = \min(q_r + q_c, u)$ *and* $q = \mathsf{zt}(q_r) \cdot q_h + \min(q_c, u) \cdot q_h$*.*

We note that the bound of Eq. (11) is not true if A_{F} is not sequential. Indeed, consider the non-sequential A_{F} that queries $L_i \leftarrow \mathrm{CIO}(1, S_i)$ for $i = 1, \ldots, q_c$ and distinct $S_1, \ldots, S_{q_c}$, then queries $(S'_j, L'_j) \leftarrow \mathrm{RIO}(1)$ for $j = 1, \ldots, q_r$, and returns 1 iff there is some i, j such that $(S_i, L_i) = (S'_j, L'_j)$. Then $\mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \geq q_c q_r \cdot 2^{-\mathrm{sl}} \cdot (1 - 2^{-\mathrm{sl}})$, which could exceed the bound of Eq. (11).

In applications, the input S will be the salt, which can be chosen to have length 128–256 bits, making the second term in Eq. (11) small, so the focus is the first term, namely $\mathbf{GP}_{\mathsf{PD}}(q, q_h, q_w)$. The $\mathsf{zt}(q_r) \cdot q_h$ term in q covers the RIO queries while the $\min(q_c, u) \cdot q_h$ term covers the CIO queries, indicating that the latter are more costly than the former. The difference impacts the bounds for FPBE privacy (where $q_c = 0$) versus authenticity (where q_c could be positive). This differentiation is indeed why we have modeled RIO and CIO separately.

In the proof, the guessing probability is used to bound the probability that a hash query includes a target password $\mathbf{P}[i]$. The difficulty is that the guessing adversary that we build does not know i. A naive analysis accordingly expends q_h TEST queries per user to cover the RIO queries, which our proof reduces to q_h overall. This reduction exploits the randomness of inputs in the RIO queries, and does not work for CIO queries.

PROOFS OF THEOREMS 6.1 AND 6.2. We can now easily obtain the proofs of Theorems 6.1 and 6.2 (of Section 6) by combining the composition theorems with Theorem 7.3. In more detail, starting with Theorem 6.1, we first apply Theorem 7.1 to get adversaries $A_{\mathsf{SE}}, A_{\mathsf{F}}$ such that

$$\mathbf{Adv}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \leq \mathbf{Adv}^{\mathrm{ind\$}}_{\mathsf{SE},q_s}(A_{\mathsf{SE}}) + \mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}})\ ,$$

where A_{F} makes $q_s, 0, q_h$ queries to its $\mathrm{RIO}, \mathrm{CIO}, \mathrm{H}$ oracles, respectively. Now applying Theorem 7.3 with $q_r = q_s$, $q_c = 0$ and q_h unchanged, we get

$$\mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \leq \mathbf{GP}_{\mathsf{PD}}(q) + \frac{q_s(q_s-1)}{2^{\mathrm{sl}}}\ ,$$

where $q = \mathsf{zt}(q_s) \cdot q_h + \min(0, u) \cdot q_h \leq q_h$, which yields Theorem 6.1. Similarly, for Theorem 6.2, we first apply Theorem 7.2 to get adversaries $A_{\mathsf{SE}}, A_{\mathsf{F}}$ such that

$$\mathbf{Adv}^{\mathrm{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \leq \mathbf{Adv}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_v}(A_{\mathsf{SE}}) + \mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \ ,$$

where A_{F} makes q_s, q_v, q_h queries to its $\mathrm{RIO}, \mathrm{CIO}, \mathrm{H}$ oracles, respectively. Now applying Theorem 7.3 with $q_r = q_s$, $q_c = q_v$ and q_h unchanged, we get

$$\mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \leq \mathbf{GP}_{\mathsf{PD}}(q, q_h, q_w) + \frac{q_s(q_s-1)}{2^{\mathrm{sl}}} \ ,$$

where $q_w = \min(q_s + q_v, u)$ and $q = \mathsf{zt}(q_s) \cdot q_h + \min(q_v, u) \cdot q_h$, which yields Theorem 6.2.

8 Attacks

We describe attacks to consider whether the terms in our advantage bounds are tight. In particular, we consider the guessing probability terms $\mathbf{GP}_{\mathsf{PD}}(q)$, and whether the parameter q in the bound is optimal.

Tightness of Theorem 6.1. We begin with privacy (PIND\$), where the tightness of the $\mathbf{GP}_{\mathsf{PD}}(q_h)$ term in Theorem 6.1 is implied by the following proposition. Given q_h, select ℓ large enough so that the subtracted term in Eq. (12) is negligible, and select A_{pg}, making q_h Test queries, so that $\mathbf{Adv}^{\mathrm{pg}}_{\mathsf{PD},u}(A_{\mathrm{pg}}) = \mathbf{GP}_{\mathsf{PD}}(q_h)$. Then Proposition 8.1 implies there is an adversary A making q_h H queries and achieving $\mathbf{Adv}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A)$ close to $\mathbf{GP}_{\mathsf{PD}}(q_h)$. The proof uses the brute-force attack and for completeness is included in Appendix G.

Proposition 8.1 *Let* SE *be a symmetric encryption scheme and let PBKDF* $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *be defined by* $\mathsf{F}[\mathrm{H}](P,S) = \mathrm{H}(P,S)$*, where we model* $\mathrm{H} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *as a random oracle. Let* $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE},\mathsf{F}]$*. Let* PD *be a password distribution for* $u \geq 1$ *users. Let* $y \in \{b, a\}$*. Let* $\ell \geq 1$*. Suppose* A_{pg} *is a password-guessing adversary making* q_h Test *queries in game* $\mathbf{G}^{\mathrm{pg}}_{\mathsf{PD},u}$*. Then we can construct an adversary* $A \in \mathcal{A}_y$ *such that*

$$\mathbf{Adv}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) \geq \mathbf{Adv}^{\mathrm{pg}}_{\mathsf{PD},u}(A_{\mathrm{pg}}) - \frac{q_h}{2^{\mathsf{SE.cl}(\ell)}} \ . \tag{12}$$

Adversary A *makes* q_h H *queries and* u Salt, Enc *queries. Its running time is about that of* A_{pg}*.*

Tightness of Theorem 6.2. Next, we consider authenticity (PAUTH), as expressed in Theorem 6.2. The following proposition implies tightness when $q_s = 0$, meaning that there are no Salt, Enc queries. We additionally assume access to a key-robustness adversary, which finds collisions of arbitrary size with advantage 1; we do so because Theorem 6.2 makes no requirement of key-robustness. This is in fact the setting of the partitioning-oracle attack, which is detailed in the proof of Proposition 8.2.

The proposition implies tightness as follows: Given $q_w \leq u$ and q_h, select A_{pg} making $q_w \cdot q_h$ Test queries, covering q_h password guesses over q_w users, so that $\mathbf{Adv}^{\mathrm{pg}}_{\mathsf{PD},u}(A_{\mathrm{pg}}) = \mathbf{GP}_{\mathsf{PD}}(q_w \cdot q_h, q_h, q_w)$. Then Proposition 8.2 implies there is an adversary A_{po} making q_h H queries and q_w Verify queries, such that A_{po} achieves advantage $\mathbf{Adv}^{\mathrm{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\mathrm{po}})$ close to $\mathbf{GP}_{\mathsf{PD}}(q_w \cdot q_h, q_h, q_w)$. This matches the term in Theorem 6.2 when $q_s = 0$; in particular $\mathbf{GP}_{\mathsf{PD}}(\min(q_v, u) \cdot q_h, q_h, \min(q_v, u))$, where $q_v = q_w \leq u$.

Proposition 8.2 *Let* SE *be a symmetric encryption scheme and let PBKDF* $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *be defined by* $\mathsf{F}[\mathrm{H}](P,S) = \mathrm{H}(P,S)$*, where we model* $\mathrm{H} : \{0,1\}^* \times \{0,1\}^{\mathrm{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$

Game $\mathbf{G}^{\text{cmt-}\ell}_{\mathsf{SE},\gamma}$

FIN$((K_1, N_1, A_1, M_1), \ldots, (K_\gamma, N_\gamma, A_\gamma, M_\gamma))$:
1 Require: $\mathsf{WiC}_\ell(K_i, N_i, A_i, M_i)$ are all distinct
2 Return $(\mathsf{SE.Enc}(K_1, N_1, A_1, M_1) = \cdots = \mathsf{SE.Enc}(K_\gamma, N_\gamma, A_\gamma, M_\gamma))$

Game $\mathbf{G}^{\text{pcmt-}\ell}_{\mathsf{FPBE},\gamma}$

FIN$((P_1, S_1, N_1, A_1, M_1), \ldots, (P_\gamma, S_\gamma, N_\gamma, A_\gamma, M_\gamma))$:
1 Require: $\mathsf{PWiC}_\ell(P_i, S_i, N_i, A_i, M_i)$ are all distinct
2 Return $(\mathsf{FPBE.Enc}(P_1, S_1, N_1, A_1, M_1) = \cdots = \mathsf{FPBE.Enc}(P_\gamma, S_\gamma, N_\gamma, A_\gamma, M_\gamma))$

Figure 11: Games defining key-committing security, for symmetric encryption scheme SE (above) and FPBE (below).

Game $\mathbf{G}^{\text{cr}}_{\mathsf{F}}$

FIN$((P_1, S_1), (P_2, S_2))$:
1 Return $((P_1, S_1) \neq (P_2, S_2) \wedge \mathsf{F}(P_1, S_1) = \mathsf{F}(P_2, S_2))$

Figure 12: Game defining collision-resistance for PBKDF F.

as a random oracle. Let $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$. *Let* PD *be a password distribution for* $u \geq 1$ *users. Let* $y \in \{b, a\}$. *Suppose* A_{pg} *is an adversary in the* $\mathbf{G}^{\text{pg}}_{\mathsf{PD},u}$ *game making* (q, q_h, q_w) TEST *queries, and we are also given an adversary* $A_{\text{krob\$}}$ *which violates* γ*-way robustness with advantage* 1 *for any* γ. *Then we can construct an adversary* $A_{\text{po}} \in \mathcal{A}_y$ *achieving advantage*

$$\mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\text{po}}) \geq \mathbf{Adv}^{\text{pg}}_{\mathsf{PD},u}(A_{\text{pg}}) \; . \tag{13}$$

Adversary A_{po} *makes* q_h H *queries and* $q_w \leq u$ VERIFY *queries.*

The proof of Proposition 8.2 is given in Appendix G, along with a formalization of the partitioning-oracle attack.

9 Key-robustness of DtE

We have seen that **DtE** preserves privacy and authenticity of the base symmetric encryption scheme SE. Now we show that it does the same for robustness, or committing security. There are various definitions of robustness or committing security for which we could show this, but we chose to use the strongest, from [7]. In this setting, the ciphertext can be a commitment to a key, or to more; for example it can be a commitment to the key, message, and nonce. BH [7] define CMT-ℓ security of the base scheme SE for $\ell = 1, 3, 4$. Below we extend these to define PCMT-ℓ security of an FPBE scheme. Then in Proposition 9.1 we show that if the base scheme SE is CMT-ℓ secure and F is collision-resistant then the FPBE scheme $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ is PCMT-ℓ-secure.

The encryption-based definitions of key-committing AE are in Figure 11. The SE notion is the same as that of [7], while we have introduced the natural extension to FPBE. We focus on encryption-based definitions, but note that the decryption-based notions could be used with an appropriate definition of tidiness for our syntax.

The function WiC_ℓ, as in [7], represents **W**hat **i**s **C**ommitted. This categorizes cases where the ciphertext could be a commitment to only the key ($\ell = 1$), or a commitment to all of the key, nonce, associated data and message ($\ell = 4$). For SE, we consider $\ell \in \{1, 4\}$. For FPBE, we consider $\ell = 1$, indicating that only the key P is committed; $\ell = 2$, indicating that the key P and salt S are committed; and $\ell = 5$, where all five FPBE encryption inputs are committed. The key-committing advantage of an adversary A is defined by $\mathbf{Adv}^{\text{cmt-}\ell}_{\mathsf{SE},\gamma}(A) = \Pr[\mathbf{G}^{\text{cmt-}\ell}_{\mathsf{SE},\gamma}(A)]$ for SE and by $\mathbf{Adv}^{\text{pcmt-}\ell}_{\mathsf{FPBE},\gamma}(A) = \Pr[\mathbf{G}^{\text{pcmt-}\ell}_{\mathsf{FPBE},\gamma}(A)]$ for FPBE.

We additionally define PBKDF collision-resistance in Figure 12. The cr advantage of an adversary A is given by $\mathbf{Adv}^{\text{cr}}_{\mathsf{F}}(A) = \Pr[\mathbf{G}^{\text{cr}}_{\mathsf{F}}(A)]$. This is a different requirement than kd (prf) security as discussed in Section 7.

In the following proposition, we show that the **DtE** transform preserves key-committing security of the base scheme, as long as F is collision-resistant. The proof of Proposition 9.1 is in Appendix H.

Proposition 9.1 *Let* SE *be a symmetric encryption scheme, let* $\mathsf{F} : \{0,1\}^* \times \{0,1\}^{\text{sl}} \to \{0,1\}^{\mathsf{SE.kl}}$ *be a PBKDF, and let* $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$. *Let* $\gamma \geq 2$. *Given adversary* A *in the* $\mathbf{G}^{\text{pcmt-}\ell}_{\mathsf{FPBE},\gamma}$ *game, we can construct* $A_{\mathsf{SE}}, A_{\mathsf{F}}$ *such that*

$$\mathbf{Adv}^{\text{pcmt-}\ell}_{\mathsf{FPBE},\gamma}(A) \leq \mathbf{Adv}^{\text{cmt-}\ell'}_{\mathsf{SE},\gamma}(A_{\mathsf{SE}}) + \mathbf{Adv}^{\text{cr}}_{\mathsf{F}}(A_{\mathsf{F}}) \,, \tag{14}$$

where when $\ell \in \{1, 2\}$ *then* $\ell' = 1$, *and when* $\ell = 5$ *then* $\ell' = 4$.

Acknowledgments

We thank the anonymous reviewers for their feedback and suggestions.

References

[1] M. Abdalla, M. Bellare, and G. Neven. Robust encryption. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Heidelberg, Feb. 2010. 4, 13

[2] A. Albertini, T. Duong, S. Gueron, S. Kölbl, A. Luykx, and S. Schmieg. How to abuse and fix authenticated encryption without key commitment. In *31st USENIX Security Symposium*, 2022. 4, 7, 13

[3] J. Alwen, B. Chen, C. Kamath, V. Kolmogorov, K. Pietrzak, and S. Tessaro. On the complexity of scrypt and proofs of space in the parallel random oracle model. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 358–387. Springer, Heidelberg, May 2016. 4, 7, 16

[4] J. Alwen, B. Chen, K. Pietrzak, L. Reyzin, and S. Tessaro. Scrypt is maximally memory-hard. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 33–62. Springer, Heidelberg, Apr. / May 2017. 4, 7, 16

[5] M. Armour and C. Cid. Partition oracles from weak key forgeries. In M. Conti, M. Stevens, and S. Krenn, editors, *CANS 2021*. LNCS, Springer, December 2021. 8

[6] M. Backendal, M. Haller, and K. G. Paterson. MEGA: Malleable encryption goes awry. In T. Ristenpart and P. Traynor, editors, *IEEE S&P 2023*. IEEE Computer Society Press, May 2023. 4

[7] M. Bellare and V. T. Hoang. Efficient schemes for committing authenticated encryption. In O. Dunkelman and S. Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 845–875. Springer, Heidelberg, May / June 2022. 4, 7, 13, 17, 23, 24

[8] M. Bellare, V. T. Hoang, and S. Keelveedhi. Instantiating random oracles via UCEs. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415. Springer, Heidelberg, Aug. 2013. 19

[9] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, May 2003. 19

[10] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, Dec. 2000. 4

[11] M. Bellare, R. Ng, and B. Tackmann. Nonces are noticed: AEAD revisited. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 235–265. Springer, Heidelberg, Aug. 2019. 3, 4, 7, 10, 14, 15

[12] M. Bellare, T. Ristenpart, and S. Tessaro. Multi-instance security and its application to password-based cryptography. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 312–329. Springer, Heidelberg, Aug. 2012. 2, 4, 7, 9, 19

[13] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 4, 16

[14] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 8, 29, 31, 34, 43, 44

[15] A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson. Argon2 memory-hard function for password hashing and proof-of-work applications. IETF Network Working Group, RFC 9106, September 2021. 4, 7, 16

[16] P. Bose, V. T. Hoang, and S. Tessaro. Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 468–499. Springer, Heidelberg, Apr. / May 2018. 3, 10, 14

[17] Boxcryptor. Technical overview. https://www.boxcryptor.com/en/technical-overview/, visited on October 17, 2022. 3

[18] G. Demay, P. Gazi, U. Maurer, and B. Tackmann. Per-session security: Password-based cryptography revisited. *J. Comput. Secur.*, 27(1):75–111, 2019. 4, 7

[19] Y. Dodis, P. Grubbs, T. Ristenpart, and J. Woodage. Fast message franking: From invisible salamanders to encryptment. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Heidelberg, Aug. 2018. 7

[20] M. Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. National Institute of Standards and Technology SP 800-38D, Nov. 2007. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf. 7

[21] A. Everspaugh, R. Chatterjee, S. Scott, A. Juels, and T. Ristenpart. The pythia PRF service. In J. Jung and T. Holz, editors, *USENIX Security 2015*, pages 547–562. USENIX Association, Aug. 2015. 7

[22] P. Farshim, C. Orlandi, and R. Roşie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symm. Cryptol.*, 2017(1):449–473, 2017. 4, 13

[23] V. Goyal, A. O'Neill, and V. Rao. Correlated-input secure hash functions. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 182–200. Springer, Heidelberg, Mar. 2011. 19

[24] P. Grubbs, J. Lu, and T. Ristenpart. Message franking via committing authenticated encryption. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 66–97. Springer, Heidelberg, Aug. 2017. 4, 7, 13

[25] T. Jager, M. Stam, R. Stanley-Oakes, and B. Warinschi. Multi-key authenticated encryption with corruptions: Reductions are lossy. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 409–441. Springer, Heidelberg, Nov. 2017. 6, 11

[26] B. Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898, Sep. 2000. https://datatracker.ietf.org/doc/html/rfc2898. 2, 4, 16

[27] R. W. F. Lai, C. Egger, M. Reinert, S. S. M. Chow, M. Maffei, and D. Schröder. Simple password-hardened encryption services. In W. Enck and A. P. Felt, editors, *USENIX Security 2018*, pages 1405–1421. USENIX Association, Aug. 2018. 7

[28] R. W. F. Lai, C. Egger, D. Schröder, and S. S. M. Chow. Phoenix: Rebirth of a cryptographic password-hardening service. In E. Kirda and T. Ristenpart, editors, *USENIX Security 2017*, pages 899–916. USENIX Association, Aug. 2017. 7

[29] J. Len, P. Grubbs, and T. Ristenpart. Partitioning oracle attacks. In M. Bailey and R. Greenstadt, editors, *30th USENIX Security Symposium*. USENIX Association, 2021. 2, 4, 6, 7, 8, 13, 47

[30] J. Len, P. Grubbs, and T. Ristenpart. Authenticated encryption with key identification. In S. Agrawal and D. Lin, editors, *ASIACRYPT 2022*. LNCS, Springer, December 2022. 8

[31] 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*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, Feb. 2004. 20

[32] MEGA. Security and why it matters. https://mega.io/security, visited on October 17, 2022. 3

[33] MEGAprivacy. Eight years of mega – tweet. https://twitter.com/MEGAprivacy/status/1352564229044277248, visited on October 17, 2022. 3

[34] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336–375, 1999. 19

[35] C. Percival. Stronger key derivation via sequential memory-hard functions. In *BSDCan*, 2009. 4, 7, 16

[36] K. Pietrzak and J. Sjödin. Weak pseudorandom functions in minicrypt. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 423–436. Springer, Heidelberg, July 2008. 19

[37] J. Pijnenburg and B. Poettering. Encrypt-to-self: Securely outsourcing storage. In L. Chen, N. Li, K. Liang, and S. A. Schneider, editors, *ESORICS 2020, Part I*, volume 12308 of *LNCS*, pages 635–654. Springer, Heidelberg, Sept. 2020. 8

[38] N. Provos and D. Mazieres. A future-adaptable password scheme. In *USENIX Annual Technical Conference, FREENIX Track*, volume 1999, pages 81–91, 1999. 4, 7, 16

[39] P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, Nov. 2002. 2, 3, 4, 7, 10

[40] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006. 7

[41] Shadowsocks. https://github.com/shadowsocks, visited on October 18, 2022. 8

[42] J. Woodage, R. Chatterjee, Y. Dodis, A. Juels, and T. Ristenpart. A new distribution-sensitive secure sketch and popularity-proportional hashing. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 682–710. Springer, Heidelberg, Aug. 2017. 4, 9

A Proofs of authenticity lemmas

Proof of Lemma 4.1: We design adversary A_{auth} as follows. It chooses a random user index J between 1 and u; the index J will refer to the one user in A_{auth}'s own game. It chooses each of $K_i \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}}$ for $1 \le i \le u$, $i \ne J$. It then runs $A_{\text{auth-c}}$, responding to $\textsc{Enc}(i, N, M, A)$ queries by doing:

If $i = J$ then $C \leftarrow \mathbf{G}^{\text{auth}}_{\mathsf{SE},1}.\textsc{Enc}(1, N, M, A)$ else $C \leftarrow \mathsf{SE.Enc}(K_i, N, M, A)$
$\text{MT}[i, \mathsf{SE.NI}(N), C, A] \leftarrow M$; Return C

When $A_{\text{auth-c}}$ makes a $\textsc{Verify}(i, I, C, A)$ query, it responds by doing:

If $(\text{MT}[i, I, C, A] \ne \perp)$ then return $\perp$
If $i = J$ then return $\mathbf{G}^{\text{auth}}_{\mathsf{SE},1}.\textsc{Verify}(1, I, C, A)$
  Else $M \leftarrow \mathsf{SE.Dec}(K_i, I, C, A)$; return $(M \ne \perp)$

When $A_{\text{auth-c}}$ makes an $\textsc{Expose}(i)$ query, it does:

If $i = J$ then return $\perp$ else return K_i

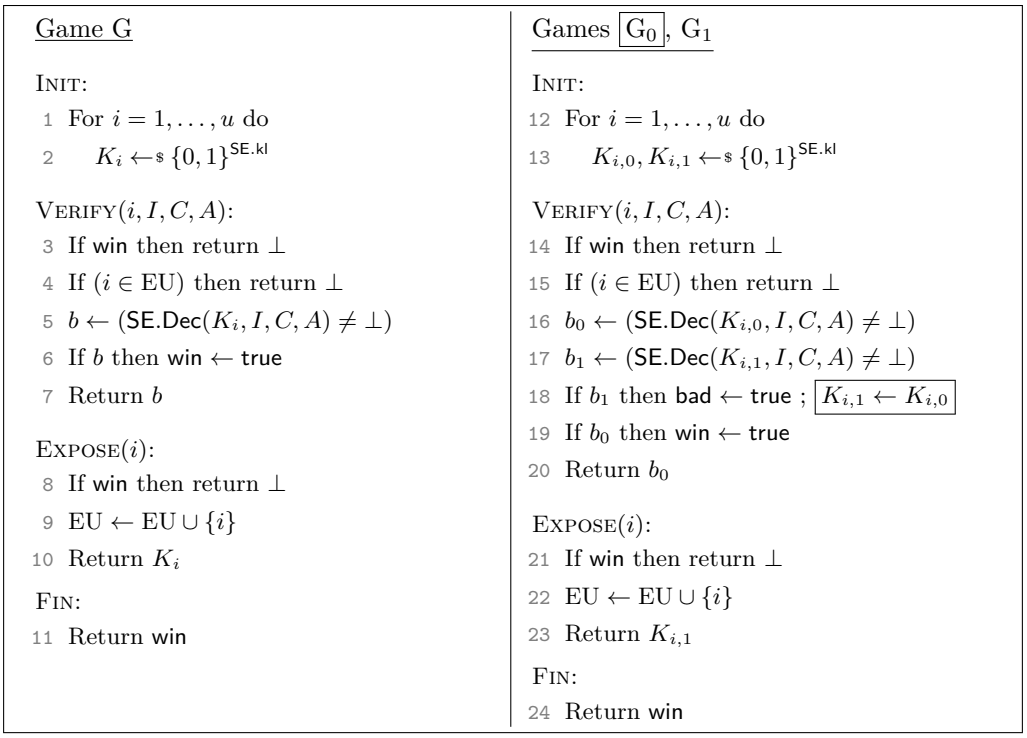

Figure 13: Games for the proof of Lemma 4.2, where G_0 includes the boxed code and G_1 does not.

Note that if $A_{\text{auth-c}}$ makes q_e, q_v queries per user (in particular for user J) then A_{auth} makes q_e, q_v total queries. Moreover, A_{auth} inherits the order of queries and the uniqueness of nonces from $A_{\text{auth-c}}$, meaning that it preserves the adversary class and sequential status.

What is required for A_{auth} to win in its one-user game? $A_{\text{auth-c}}$ must win during a VERIFY$(J, \cdot, \cdot, \cdot)$ query, which necessarily reqiuires that EXPOSE(J) was never queried prior. Since J is chosen uniformly at random (independent of the execution of $A_{\text{auth-c}}$) this means

$$\mathbf{Adv}^{\text{auth}}_{\mathsf{SE},1}(A_{\text{auth}}) \geq \frac{1}{u} \cdot \mathbf{Adv}^{\text{auth-c}}_{\mathsf{SE},u}(A_{\text{auth-c}}) \ .$$

This is the desired bound in Lemma 4.1. ▌

Proof of Lemma 4.2: We can assume the adversary in the y = b case never sets un to false in game $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},u}$, and thus drop writing un variables of that game. Having done this, game G of Figure 13 further silences the VERIFY, EXPOSE oracles (meaning, has them return $\perp$) if win is set. Games $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},u}$, G are thus the same until win is set, but FIN returns win, so we have

$$\mathbf{Adv}^{\text{auth-c}}_{\mathsf{SE},u}(A_{\text{auth-c}}) = \Pr[\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},u}(A_{\text{auth-c}})] = \Pr[\mathrm{G}(A_{\text{auth-c}})] \ .$$

Now consider games $\mathrm{G}_0, \mathrm{G}_1$ in Figure 13, where the former includes the boxed code and the latter does not. At line 13, two keys, $K_{i,0}, K_{i,1}$, are chosen per user i, with the oracle VERIFY generating results under both (lines 16,17). The setting of win, and what VERIFY returns, is done as per $K_{i,0}$, but EXPOSE(i) returns $K_{i,1}$, the intent being that EXPOSE is now easily simulated by an authenticity adversary. Line 18 sets bad if decryption under $K_{i,1}$ was successful, the boxed code

reverting $K_{i,1}$ to $K_{i,0}$ in this case. We claim that

$$\Pr[\mathrm{G}(A_{\text{auth-c}})] = \Pr[\mathrm{G}_0(A_{\text{auth-c}})] \ . \tag{15}$$

Let us explain Eq. (15). The silencing ensures the games are the same after win is set, so assume it is not yet set. Consider the first time bad is set. We have $b_1 = \mathsf{true}$ and two cases for b_0: (1) $b_0 = \mathsf{true}$, or (2) $b_0 = \mathsf{false}$. If (2) then the boxed code ensures consistency of the exposed key with the VERIFY response. If (1) then win is set, so the silencing ensures consistency.

Games $\mathrm{G}_0, \mathrm{G}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [14],

$$\begin{aligned}\Pr[\mathrm{G}_0(A_{\text{auth-c}})] &= \Pr[\mathrm{G}_1(A_{\text{auth-c}})] + \Pr[\mathrm{G}_0(A_{\text{auth-c}})] - \Pr[\mathrm{G}_1(A_{\text{auth-c}})] \\ &\leq \Pr[\mathrm{G}_1(A_{\text{auth-c}})] + \Pr[\mathrm{G}_1(A_{\text{auth-c}}) \text{ sets } \mathsf{bad}] \ .\end{aligned}$$

We now observe that

$$\Pr[\mathrm{G}_1(A_{\text{auth-c}}) \text{ sets } \mathsf{bad}] = \Pr[\mathrm{G}_1(A_{\text{auth-c}})] \ .$$

This is by symmetry. We can think of the roles of $K_{i,0}$ and $K_{i,1}$ as being swapped. Putting the above together we now have

$$\mathbf{Adv}^{\text{auth-c}}_{\mathsf{SE},u}(A_{\text{auth-c}}) \leq 2 \cdot \Pr[\mathrm{G}_1(A_{\text{auth-c}})] \ .$$

To conclude, we build A_{auth} so that

$$\Pr[\mathrm{G}_1(A_{\text{auth-c}})] \leq \mathbf{Adv}^{\text{auth}}_{\mathsf{SE},u}(A_{\text{auth}}) \ . \tag{16}$$

Adversary A_{auth} picks $K_{1,1}, \ldots, K_{u,1} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}}$ and initializes win, EU to $\mathsf{false}, \emptyset$, respectively. It then runs $A_{\text{auth-c}}$, replying to its oracle queries as follows. On query $\textsc{Verify}(i, I, C, A)$, it responds via:

If win then return $\perp$
If $(i \in \mathrm{EU})$ then return $\perp$
$b_0 \leftarrow \mathbf{G}^{\text{auth}}_{\mathsf{SE},u}.\textsc{Verify}(i, I, C, A)$; Return b_0

It responds to an $\textsc{Expose}(i)$ query as per lines 21–23 of G_1. As long as $A_{\text{auth-c}}$ makes a winning VERIFY query so will A_{auth}, proving Eq. (16) and completing the proof of Lemma 4.2. ∎

B Proof of Theorem 5.1

Proof of Theorem 5.1: For this proof we refer to the games in Figure 14. We assume that adversaries meet required conditions and omit writing those checks.

We first claim that $\Pr[\mathbf{G}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}(A)] = \Pr[\mathrm{G}_0(A)]$. This follows simply by observing that G_0 consists of all the same steps as $\mathbf{G}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}$, with the omission of conditions we have assumed to be met. The games of Figure 14 have also added a procedure H for the random oracle. Note that H is accessible to the FPBE scheme, and all adversaries in the statement of Theorem 5.1 have query access to the same RO H. It will suffice for adversaries to forward queries to H rather than to implement its functionality.

Games $\mathrm{G}_0, \mathrm{G}_1$ are identical-until-bad so the Fundamental Lemma of Game Playing [14] implies that

$$\Pr[\mathrm{G}_0(A)] - \Pr[\mathrm{G}_1(A)] \leq \Pr[\mathrm{G}_1(A) \text{ sets } \mathsf{bad}] \ .$$

Games $\boxed{\mathrm{G}_0}$, G_1

INIT:
1 $d \leftarrow\!\!\$\ \{0,1\}$; $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$

ENC(i, N, M, A):
2 $C_1 \leftarrow \mathsf{FPBE.Enc}(\mathbf{P}[i], S_i, N, M, A)$; $C_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{FPBE.cl}(|M|)}$
3 $\mathrm{MT}[i, S_i, \mathsf{FPBE.NI}(N), C_d, A] \leftarrow M$; Return C_d

DEC(i, S, I, C, A):
4 If $(\mathrm{MT}[i, S, I, C, A] \neq \bot)$ then return $\mathrm{MT}[i, S, I, C, A]$
5 If $(d = 0)$ then return $\bot$
6 $M \leftarrow \bot$; $M' \leftarrow \mathsf{FPBE.Dec}(\mathbf{P}[i], S, I, C, A)$
7 If $(M' \neq \bot)$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{M \leftarrow M'}$
8 Return M

SALT(i):
9 $s(i) \leftarrow s(i) + 1$; $S_i \leftarrow\!\!\$\ \mathsf{FPBE.SS}$; Return S_i

H(X):
10 Return $\mathrm{H}(X)$

FIN(d'):
11 Return $(d' = d)$

Figure 14: Games for the proof of Theorem 5.1.

We next design $A_{\mathrm{pind\$}}, A_{\mathrm{pauth}}$ such that

$$\Pr[\mathrm{G}_1(A)] \leq \Pr[\mathbf{G}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\mathrm{pind\$}})] \tag{17}$$

$$\Pr[\mathrm{G}_1(A) \text{ sets } \mathsf{bad}] \leq \Pr[\mathbf{G}^{\mathrm{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\mathrm{pauth}})] \,. \tag{18}$$

Adversary $A_{\mathrm{pind\$}}$ operates as follows. It runs A and responds to ENC and SALT queries by forwarding them to its own oracles, and returning the responses to A. It also forwards H queries. On a DEC query, $A_{\mathrm{pind\$}}$ simply returns $\bot$ to A. When A guesses a bit d', $A_{\mathrm{pind\$}}$ guesses that same challenge bit. This is precisely the setting of $\mathrm{G}_1(A)$, and the challenge bits are consistent, justifying Eq. (17).

Next we turn to A_{pauth}. It chooses a bit d then runs A. In either case, it responds to H queries by forwarding them to H itself. If $d = 1$, it responds to ENC queries by forwarding them to its own oracle, then returning the response to A. If $d = 0$, encryption responses are a random ciphertext (of the appropriate length). Salt queries are all forwarded to its own oracle, with the response forwarded to A. When A makes a query $\mathrm{DEC}(i, S, I, C, A)$, A_{pauth} makes a query $\mathrm{VERIFY}(i, S, I, C, A)$ and regardless of the response, returns $\bot$ to A. Recall that in the SAUTH game, A_{pauth} wins if it makes a verification query $\mathrm{VERIFY}(i, S, I, C, A)$ such that $\mathsf{FPBE.Dec}(\mathbf{P}[i], S, I, C, A) \neq \bot$. This is precisely the bad condition on lines 6,7. Because A_{pauth} wins as long as this condition is reached, Eq. (18) holds.

Since A_{pauth} only forwards queries of A, it preserves the adversary class (basic or advanced) of A, as does $A_{\mathrm{pind\$}}$. Even if A is not sequential, A_{pauth} can be made sequential by making all of its VERIFY queries at the end. Because all of the DEC responses given to A are $\bot$, it makes no difference if

A_{pauth} waits until the end to make its VERIFY queries. This justifies the claim that A_{pauth} is always sequential, which we use to motivate various simplifying assumptions about sequential adversaries (throughout other proofs in this paper).

Combining the above results and advantage definitions, we have

$$\begin{aligned}
\mathbf{Adv}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}(A) &= 2\Pr[\mathbf{G}^{\text{pae}}_{\mathsf{FPBE},\mathsf{PD},u}(A)] - 1 \\
&\le 2\Pr[\mathrm{G}_0(A)] - 1 \\
&\le 2\left(\Pr[\mathrm{G}_1(A)] + \Pr[\mathrm{G}_1(A) \text{ sets } \mathsf{bad}]\right) - 1 \\
&\le 2\left(\Pr[\mathbf{G}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\text{pind\$}})] + \Pr[\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\text{pauth}})]\right) - 1 \\
&\le \mathbf{Adv}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\text{pind\$}}) + 2 \cdot \mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\text{pauth}})
\end{aligned}$$

This completes the proof of Eq. (3). ∎

C Proof of Theorem 6.3

The proof of Theorem 6.3 involves technical challenges. As discussed in Section 1, it is not obvious why key-robustness of SE helps to improve the bound. The proof makes a connection to AUTH-C and exploits our Lemma 4.2.

Proof of Theorem 6.3: Recall that in the ROM, game $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$ adds a procedure H for the random oracle. Below we will often, without explicit mention, exploit the assumption, from the definition of a password distribution, that $\mathbf{P}[1], \ldots, \mathbf{P}[u]$ are distinct. Similarly, we will exploit the assumption that A is sequential, meaning it makes all its ENC, SALT queries before any of its VERIFY queries. The algorithms Find1, Find2, used in some games and constructed adversaries, were defined in Section 3.

Consider the games of Figure 15, where G_1 includes the boxed code and G_0 does not. We have let $q_{s,i}$ be the number of SALT(i) queries, so that $q_s = q_{s,1} + \cdots + q_{s,u}$. Let $S_{i,j}$ denote the salt that SALT(i) would pick the j-th time it is called. The games start by picking these values up front in INIT, together with keys $K_{i,j}$ that ENC would use. However, while G_0 picks the $S_{i,j}$ at random, G_1 ensures that they are distinct. Down the line, this will allow password-guessing adversary A_{pg} to use the input to uniquely identify the user in an ENC query and thereby minimize the number of TEST queries it makes. For now, we just note that game G_0 is equivalent to game $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$, so

$$\mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A) = \Pr[\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A)] = \Pr[\mathrm{G}_0(A)] \ .$$

Trivially we have

$$\Pr[\mathrm{G}_0(A)] = \Pr[\mathrm{G}_1(A)] + (\Pr[\mathrm{G}_0(A)] - \Pr[\mathrm{G}_1(A)]) \ . \tag{19}$$

We bound the terms above in turn, starting with the second. Games $\mathrm{G}_0, \mathrm{G}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [14] we have

$$\Pr[\mathrm{G}_0(A)] - \Pr[\mathrm{G}_1(A)] \le \Pr[\mathrm{G}_0(A) \text{ sets } \mathsf{bad}] \ .$$

Flag bad is set when two of the q_s salts collide, so

$$\Pr[\mathrm{G}_0(A) \text{ sets } \mathsf{bad}] \le \frac{q_s(q_s-1)}{2^{\mathrm{sl}+1}} \ .$$

Game G_0 / $\boxed{\mathrm{G}_1}$

INIT:
1 $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$
2 For $i = 1, \ldots, u$ and $j = 1, \ldots, q_{s,i}$ do
3 $\quad S_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$; $K_{i,j} \leftarrow \mathrm{H}(\mathbf{P}[i], S_{i,j})$
4 $\quad$ If $(S_{i,j} \in \mathrm{OS})$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{S_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}} \setminus \mathrm{OS}}$
5 $\quad \mathrm{OS} \leftarrow \mathrm{OS} \cup \{S_{i,j}\}$

ENC(i, N, M, A):
6 $C \leftarrow \mathsf{SE.Enc}(K_{i,s(i)}, N, M, A)$; $\mathrm{MT}[i, S_{i,s(i)}, \mathsf{SE.NI}(N), C, A] \leftarrow M$; Return C

VERIFY(i, S, I, C, A):
7 If $(\mathrm{MT}[i, S, I, C, A] \neq \bot)$ then return $\bot$
8 If $(\mathrm{KT}[i, S] = \bot)$ then $\mathrm{KT}[i, S] \leftarrow \mathrm{H}(\mathbf{P}[i], S)$
9 $M \leftarrow \mathsf{SE.Dec}(\mathrm{KT}[i, S], I, C, A)$
10 If $(M \neq \bot)$ then $\mathsf{win} \leftarrow \mathsf{true}$
11 Return $(M \neq \bot)$

SALT(i):
12 $s(i) \leftarrow s(i) + 1$; Return $S_{i,s(i)}$

H(P, S):
13 If $(\mathrm{HT}[P, S] \neq \bot)$ then return $\mathrm{HT}[P, S]$
14 $\mathrm{HT}[P, S] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$; Return $\mathrm{HT}[P, S]$

FIN:
15 Return win

Figure 15: Games $\mathrm{G}_0, \mathrm{G}_1$ for the proof of Theorem 6.3, where G_1 includes the boxed code and G_0 does not.

INIT: // Games $\mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_4$
1 $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$
2 For $i = 1, \ldots, u$ and $j = 1, \ldots, q_{s,i}$ do
3 $\quad S_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}} \setminus \mathrm{OS}$; $K_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$; $\mathrm{OS} \leftarrow \mathrm{OS} \cup \{S_{i,j}\}$; $\mathrm{FT}[i, S_{i,j}] \leftarrow K_{i,j}$

ENC(i, N, M, A): // Games $\mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_4$
4 $C \leftarrow \mathsf{SE.Enc}(K_{i,s(i)}, N, M, A)$; $\mathrm{MT}[i, S_{i,s(i)}, \mathsf{SE.NI}(N), C, A] \leftarrow M$; Return C

SALT(i): // Games $\mathrm{G}_2, \mathrm{G}_3, \mathrm{G}_4$
5 $s(i) \leftarrow s(i) + 1$; Return $S_{i,s(i)}$

FIN: // Games $\mathrm{G}_2, \mathrm{G}_3$
6 Return win

Figure 16: Some oracles for subsequent games in the proof of Theorem 6.3.

Returning to Eq. (19), we now bound the first term. Towards this, Figures 16 and 17 together define games $\mathrm{G}_2, \mathrm{G}_3$, where the former includes the boxed code and the latter does not. We claim that G_2 is equivalent to G_1, meaning

$$\Pr[\mathrm{G}_1(A)] = \Pr[\mathrm{G}_2(A)] \ . \tag{20}$$

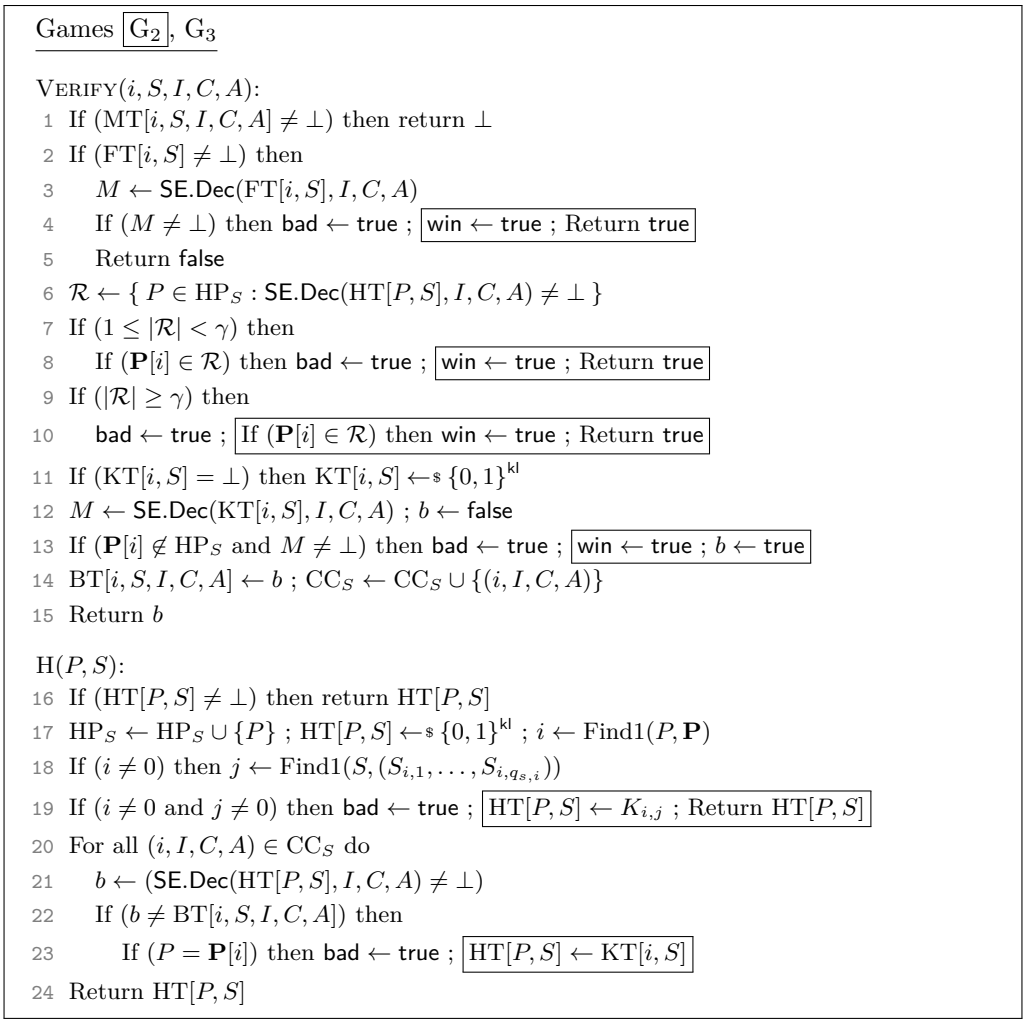

Figure 17: VERIFY and H oracles for games $\mathrm{G}_2, \mathrm{G}_3$ for the proof of Theorem 6.3, where G_2 includes the boxed code and G_3 does not. The other oracles are in Figure 16.

We now explain G_2 to justify Eq. (20). As in G_1, oracle INIT picks distinct salts $S_{i,j}$, but optimistically picks the $K_{i,j}$ to be random. In G_3 it stays that way. However, G_2, via the inclusion of the boxed code at line 19 of Figure 17, ensures that $K_{i,j} = \mathrm{H}(\mathbf{P}[i], S_{i,j})$. So responses to ENC queries in G_2 adhere to those in G_1. We now turn to arguing that the same is true for VERIFY queries.

Lines 2–5 handle the case that the S in the query is one of the $S_{i,j}$, so now suppose not. Set HP_S (defined through line 17) holds all candidate passwords P for which $\mathrm{H}(P, S)$ has been queried and $\mathrm{HT}[P, S]$ is thus defined. In a VERIFY(i, S, I, C, A) query, if $\mathbf{P}[i] \in \mathrm{HP}_S$, then the key $\mathrm{HT}[\mathbf{P}[i], S]$ for the base scheme SE is known to A, and we cannot exploit the authenticity of SE under this key. An obvious step is to now set bad and bound the probability of this via the advantage of a password-guessing adversary A_{pg} which, via its oracle, tests all $P \in \mathrm{HP}_S$ for user i. However, $|\mathrm{HP}_S|$ could be as large as q_h, leading to q_h oracle queries for each VERIFY query, for a total of $q_h q_v$, which would bring us back to the result of Theorem 6.2. The intent of the present result is exactly to reduce this number of test queries by exploiting key-robustness. Towards this, say that $P \in \mathrm{HP}_S$ is a *likely suspect* if $\mathsf{SE.Dec}(\mathrm{HT}[P, S], I, C, A) \neq \perp$. (Recall i, S, I, C, A is the query to

VERIFY.) At line 6, we have let $\mathcal{R} \subseteq \mathrm{HP}_S$ be the set of all likely suspects. We then consider the following cases.

The first case (lines 7,8) is that $1 \leq |\mathcal{R}| < \gamma$. We set bad if $\mathbf{P}[i]$ equals one of the likely suspects, the boxed code ensuring the right actions and response. To bound the probability that bad is set here, the password-guessing adversary will need at most $\gamma - 1$ queries per VERIFY query, as opposed to $|\mathrm{HP}_S|$-many.

The second case (lines 9,10) is that $|\mathcal{R}| \geq \gamma$. Testing $\mathbf{P}[i] \in \mathcal{R}$ would now need more than the $\gamma - 1$ password-guessing queries than we want to expend. However, we expect this case to not happen due to the key-robustness of SE. (This is where we will use this assumption.) Accordingly, we set bad if it happens, the boxed code again ensuring correctness.

If lines 8,10 fail to return true it must be that $\mathbf{P}[i] \notin \mathcal{R}$, and we reach line 11. There are two possibilities: either (1) $\mathbf{P}[i] \in \mathrm{HP}_S \setminus \mathcal{R}$, meaning $\mathrm{HT}[\mathbf{P}[i], S]$ is defined but $\mathsf{SE.Dec}(\mathrm{HT}[\mathbf{P}[i], S], I, C, A) = \perp$, or (2) $\mathbf{P}[i] \notin \mathrm{HP}_S$. The difficulty is that we have no way to efficiently — meaning, without having our password-guessing adversary make more queries than we want — tell which of the two cases holds. Our strategy is to consider $M \leftarrow \mathsf{SE.Dec}(\mathrm{KT}[i, S], I, C, A)$ (line 12), where key $\mathrm{KT}[i, S]$, unless it is already defined, is freshly chosen at line 11. If (1) happens, line 15 will correctly return false. If (2) happens then $\mathrm{HT}[\mathbf{P}[i], S]$ is undefined and the intent is that it takes value $\mathrm{KT}[i, S]$, the setting of win and what is returned done accordingly. Expending password guesses for line 13 will be avoided by bounding, via the authenticity of SE, the probability that $M \neq \perp$.

Now we turn to $\mathrm{H}(P, S)$ queries. Lines 18,19 handle the case that S is one of the $S_{i,j}$. We note that the distinctness of the latter salts, imposed by INIT in Figure 16, ensures that the choice of j is unique and thus line 18 is unambiguous. Now if $P = \mathbf{P}[i]$ and $\mathrm{KT}[i, S] \neq \perp$, we would like to set $\mathrm{HT}[P, S]$ to $\mathrm{KT}[i, S]$. But testing the condition to do this again would cost too many password-guessing queries. Instead, lines 20–23 check whether the default value of $\mathrm{HT}[P, S]$ chosen at line 17 is consistent, with regard to VERIFY replies, with $\mathrm{KT}[i, S]$. If NO, bad is set, and $\mathrm{HT}[P, S]$ is set to $\mathrm{KT}[i, S]$. If YES then $\mathrm{HT}[P, S]$ is left unchanged. This allows us to avoid a number of password-guessing queries proportional to q_h. The subtle thing is that at this point, $\mathrm{HT}[\mathbf{P}[i], S]$ and $\mathrm{KT}[i, S]$ would both be defined and likely different, so that we have two keys contending for the role of the base key corresponding to $S, \mathbf{P}[i]$. However, game G_2 ensures that this creates no discrepancy in the adversary's view. (Replies to VERIFY queries stay consistent with $\mathrm{HT}[\mathbf{P}[i], S]$ if the latter is defined, and otherwise with $\mathrm{KT}[i, S]$.) This completes our explanation of Eq. (20) and we now need to bound $\Pr[\mathrm{G}_2(A)]$.

Games $\mathrm{G}_2, \mathrm{G}_3$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [14] we have

$$\begin{aligned}\Pr[\mathrm{G}_2(A)] &= \Pr[\mathrm{G}_3(A)] + \Pr[\mathrm{G}_2(A)] - \Pr[\mathrm{G}_3(A)] \\ &\leq \Pr[\mathrm{G}_3(A)] + \Pr[\mathrm{G}_3(A) \text{ sets } \mathsf{bad}] \ .\end{aligned}$$

We now bound the two terms above. The flag win that G_3 returns is only set in boxed code, which is excluded in G_3, so $\Pr[\mathrm{G}_3(A)] = 0$.

It remains to bound $\Pr[\mathrm{G}_3(A) \text{ sets } \mathsf{bad}]$. This task is simplified via game G_4 of Figure 18. We claim that

$$\Pr[\mathrm{G}_3(A) \text{ sets } \mathsf{bad}] \leq \Pr[\mathrm{G}_4(A) \text{ sets } \mathsf{bad}] \ . \tag{21}$$

Let us explain Eq. (21). Game G_4 starts from G_3, making simplifications due to the boxed code being absent in G_3. The "If" at line 11 of G_4 drops the "$\mathbf{P}[i] \notin \mathrm{HP}_S$" condition of line 13 (Figure 17) of G_3, which can only increase the probability of setting bad, consistent with Eq. (21). In G_3, table

Game G_4

VERIFY(i, S, I, C, A):
1 If $(\mathrm{MT}[i, S, I, C, A] \neq \perp)$ then return $\perp$
2 If $(\mathrm{FT}[i, S] \neq \perp)$ then
3 $M \leftarrow \mathsf{SE.Dec}(\mathrm{FT}[i, S], I, C, A)$; If $(M \neq \perp)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
4 Return false
5 $\mathcal{R} \leftarrow \{\, P \in \mathrm{HP}_S : \mathsf{SE.Dec}(\mathrm{HT}[P, S], I, C, A) \neq \perp \,\}$
6 If $(1 \leq |\mathcal{R}| < \gamma)$ then
7 If $(\mathbf{P}[i] \in \mathcal{R})$ then $\mathsf{bad} \leftarrow \mathsf{true}$
8 If $(|\mathcal{R}| \geq \gamma)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
9 If $(\mathrm{KT}[i, S] = \perp)$ then $\mathrm{KT}[i, S] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$
10 $M \leftarrow \mathsf{SE.Dec}(\mathrm{KT}[i, S], I, C, A)$
11 If $(M \neq \perp)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
12 $\mathrm{CC}_S \leftarrow \mathrm{CC}_S \cup \{(i, I, C, A)\}$; Return false

H(P, S):
13 If $(\mathrm{HT}[P, S] \neq \perp)$ then return $\mathrm{HT}[P, S]$
14 $\mathrm{HP}_S \leftarrow \mathrm{HP}_S \cup \{P\}$; $\mathrm{HT}[P, S] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$; $i \leftarrow \mathrm{Find1}(P, \mathbf{P})$
15 If $(i \neq 0)$ then $j \leftarrow \mathrm{Find1}(S, (S_{i,1}, \ldots, S_{i,q_{s,i}}))$
16 If $(i \neq 0$ and $j \neq 0)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
17 For all $(i, I, C, A) \in \mathrm{CC}_S$ do
18 If $(\mathsf{SE.Dec}(\mathrm{HT}[P, S], I, C, A) \neq \perp)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
19 Return $\mathrm{HT}[P, S]$

FIN:
20 Return false

Figure 18: VERIFY, H and FIN oracles for game G_4 for the proof of Theorem 6.3. The other oracles are in Figure 16.

---

entry $\mathrm{BT}[i, S, I, C, A]$ would always be false. Game G_4 thus does not define it, and simplifies lines 20–23 to lines 17,18, including dropping the line 23 "$P = \mathbf{P}[i]$" test, which can again only increase the probability of setting bad. VERIFY always returns false, as per G_3. We are concerned only with G_3's setting of bad, not with what the game returns, so we have FIN always return false. This completes our explanation of Eq. (21).

It remains to bound $\Pr[\mathrm{G}_4(A)$ sets $\mathsf{bad}]$. For this, we design adversaries A_{pg}, $A_{\mathrm{krob\$}}$, A_{auth} and $A_{\text{auth-c}}$ such that:

$$\Pr[\mathrm{G}_4(A) \text{ sets } \mathsf{bad} \text{ at lines 7 or 16}] \leq \mathbf{Adv}^{\mathrm{pg}}_{\mathsf{PD},u}(A_{\mathrm{pg}}) \tag{22}$$

$$\leq \mathbf{GP}_{\mathsf{PD}}(\mathsf{zt}(q_s) \cdot q_h + (\gamma - 1) \cdot q_v) \tag{23}$$

$$\Pr[\mathrm{G}_4(A) \text{ sets } \mathsf{bad} \text{ at line 8}] \leq \mathbf{Adv}^{\mathrm{krob\$}}_{\mathsf{SE},q_h,\gamma}(A_{\mathrm{krob\$}}) \tag{24}$$

$$\Pr[\mathrm{G}_4(A) \text{ sets } \mathsf{bad} \text{ at lines 3 or 11}] \leq \mathbf{Adv}^{\mathrm{auth}}_{\mathsf{SE},u}(A_{\mathrm{auth}}) \tag{25}$$

$$\Pr[\mathrm{G}_4(A) \text{ sets } \mathsf{bad} \text{ at line 18}] \leq \mathbf{Adv}^{\text{auth-c}}_{\mathsf{SE},u}(A_{\text{auth-c}})\ . \tag{26}$$

Putting all the above together yields Eq. (6). We proceed to the adversary constructions.

Adversary A_{pg} is playing game $\mathbf{G}^{\mathrm{pg}}_{\mathsf{PD},u}$ (Figure 3). It runs A, responding to its oracle queries as

Adversary $A_{\rm pg}$

INIT:
1 For $i = 1, \ldots, u$ and $j = 1, \ldots, q_{s,i}$ do
2 $\quad S_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}} \setminus \mathrm{OS}$; $K_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$; $\mathrm{OS} \leftarrow \mathrm{OS} \cup \{S_{i,j}\}$; $\mathrm{FT}[i, S_{i,j}] \leftarrow K_{i,j}$

VERIFY(i, S, I, C, A):
3 If $(\mathrm{MT}[i, S, I, C, A] \neq \bot)$ then return $\bot$
4 If $(\mathrm{FT}[i, S] \neq \bot)$ then return false
5 $\mathcal{R} \leftarrow \{\ P \in \mathrm{HP}_S : \mathsf{SE.Dec}(\mathrm{HT}[P, S], I, C, A) \neq \bot\ \}$
6 If $(1 \leq |\mathcal{R}| < \gamma)$ then
7 $\quad$ For all $P \in \mathcal{R}$ do $\mathbf{G}^{\rm pg}_{\mathsf{PD},u}$.TEST$(i, P)$
8 Return false

H(P, S):
9 If $(\mathrm{HT}[P, S] \neq \bot)$ then return $\mathrm{HT}[P, S]$
10 $\mathrm{HP}_S \leftarrow \mathrm{HP}_S \cup \{P\}$; $\mathrm{HT}[P, S] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$; $j \leftarrow \mathrm{Find1}(S, (S_{i,1}, \ldots, S_{i,q_{s,i}}))$
11 If $(j \neq 0)$ then $\mathbf{G}^{\rm pg}_{\mathsf{PD},u}$.TEST$(j, P)$
12 Return $\mathrm{HT}[P, S]$

Figure 19: How adversary $A_{\rm pg}$, for the proof of Theorem 6.3, simulates A's oracles. ENC, SALT responses are as in Figure 16.

---

shown in Figure 19. The role of the $\mathbf{P}$ chosen in G_4.INIT (line 1 of Figure 16) is played by the one chosen at line 1 of $\mathbf{G}^{\rm pg}_{\mathsf{PD},u}$. At lines 7,11 of Figure 19, $A_{\rm pg}$ calls the TEST oracle provided by the $\mathbf{G}^{\rm pg}_{\mathsf{PD},u}$ game it is playing, and this query is successful (sets win in $\mathbf{G}^{\rm pg}_{\mathsf{PD},u}$) whenever $\mathrm{G}_4(A)$ sets bad at lines 7,16 of Figure 18, justifying Eq. (22). The number of TEST queries from line 7 is at most $(\gamma - 1) \cdot q_v$. This is the crucial improvement, showing how defining the set $\mathcal{R}$, and expending password-guessing queries only when it has size less than γ, pays off in reducing the number of TEST queries of $A_{\rm pg}$. The number of TEST queries from line 11 is 0 if $q_s = 0$ and is otherwise at most q_h, which, put succinctly, is at most $\mathsf{zt}(q_s) \cdot q_h$. This justifies Eq. (23).

Adversary $A_{\rm krob\$}$ is playing game $\mathbf{G}^{\rm krob\$}_{\mathsf{SE},q_h,\gamma}$ (Figure 6). It runs A, responding to the latter's oracle queries as shown in Figure 20. At line 1, $A_{\rm krob\$}$ calls its own INIT oracle to get random keys $K_1, \ldots, K_{q_h}$. At line 10, it responds to H queries using its target keys $K_1, \ldots, K_{q_h}$, so that these keys play the role of the $\mathrm{HT}[P, S]$ values. At line 7, it calls the FIN oracle of its $\mathbf{G}^{\rm krob\$}_{\mathsf{SE},q_h,\gamma}$ game. It wins whenever $\mathrm{G}_4(A)$ sets bad at line 8 of Figure 18, justifying Eq. (24).

Adversary $A_{\rm auth}$ is playing game $\mathbf{G}^{\rm auth}_{\mathsf{SE},q_s+q_v}$ (Figure 4). It runs A, responding to oracle queries as shown in Figure 21. Counter c represents a user index. Table entry $\mathrm{UT}[i, S_{i,j}]$ is the index of a user in game $\mathbf{G}^{\rm auth}_{\mathsf{SE},q_s+q_v}$ whose key will play the role of $K_{i,j}$. Line 4 is a call to $A_{\rm auth}$'s own ENC oracle for user $\mathrm{UT}[i, S_{i,s(i)}]$. Lines 7,10 are calls to $A_{\rm auth}$'s own VERIFY oracle, for users $\mathrm{UT}[i, S]$ and $\mathrm{VT}[i, S]$ (respectively), and they succeed if bad is set at lines 3 or 10 (respectively) of G_4 (Figure 18), justifying Eq. (25). Now, as per the theorem statement, for $\mathrm{y} \in \{\mathrm{b}, \mathrm{a}\}$, we need to check that if $A \in \mathcal{A}_{\rm y}$ then $A_{\rm auth} \in \mathcal{A}_{\rm y}$. The case $\mathrm{y} = \mathrm{a}$ is clear, but the case $\mathrm{y} = \mathrm{b}$ uses the fact that the salts $S_{i,j}$ are all distinct; otherwise, A repeating a nonce across two such salts (allowed) would make $A_{\rm auth}$ repeat a nonce for i (not allowed). Finally, A was assumed sequential, and $A_{\rm auth}$ makes its ENC, VERIFY queries in the same order as A, and hence is also sequential.

Adversary $A_{\text{krob\$}}$

INIT:
1 $(K_1, \ldots, K_{q_h}) \xleftarrow{\$} \mathbf{G}^{\text{krob\$}}_{\mathsf{SE},q_h,\gamma}.\text{INIT}$; $c \leftarrow 0$
2 For $i = 1, \ldots, u$ and $j = 1, \ldots, q_{s,i}$ do
3 $S_{i,j} \xleftarrow{\$} \{0,1\}^{\mathsf{sl}} \setminus \text{OS}$; $K_{i,j} \xleftarrow{\$} \{0,1\}^{\mathsf{kl}}$; $\text{OS} \leftarrow \text{OS} \cup \{S_{i,j}\}$; $\text{FT}[i, S_{i,j}] \leftarrow K_{i,j}$

VERIFY(i, S, I, C, A):
4 If $(\text{MT}[i, S, I, C, A] \neq \bot)$ then return $\bot$
5 If $(\text{FT}[i, S] \neq \bot)$ then return false
6 $\mathcal{R} \leftarrow \{\, P \in \text{HP}_S : \mathsf{SE.Dec}(\text{HT}[P, S], I, C, A) \neq \bot \,\}$
7 If $(|\mathcal{R}| \geq \gamma)$ then $\mathbf{G}^{\text{krob\$}}_{\mathsf{SE},q_h,\gamma}.\text{FIN}(I, C, A)$
8 Return false

H(P, S):
9 If $(\text{HT}[P, S] \neq \bot)$ then return $\text{HT}[P, S]$
10 $\text{HP}_S \leftarrow \text{HP}_S \cup \{P\}$; $c \leftarrow c + 1$; $\text{HT}[P, S] \leftarrow K_c$
11 Return $\text{HT}[P, S]$

Figure 20: How adversary $A_{\text{krob\$}}$, for the proof of Theorem 6.3, simulates A's oracles. ENC, SALT responses are as in Figure 16.

Adversary A_{auth}

INIT:
1 $c \leftarrow 0$
2 For $i = 1, \ldots, u$ and $j = 1, \ldots, q_{s,i}$ do
3 $S_{i,j} \xleftarrow{\$} \{0,1\}^{\mathsf{sl}} \setminus \text{OS}$; $\text{OS} \leftarrow \text{OS} \cup \{S_{i,j}\}$; $c \leftarrow c + 1$; $\text{UT}[i, S_{i,j}] \leftarrow c$

ENC(i, N, M, A):
4 $C \leftarrow \mathbf{G}^{\text{auth}}_{\mathsf{SE},q_s+q_v}.\text{ENC}(\text{UT}[i, S_{i,s(i)}], N, M, A)$
5 $\text{MT}[i, S_{i,s(i)}, \mathsf{SE.NI}(N), C, A] \leftarrow M$; Return C

VERIFY(i, S, I, C, A):
6 If $(\text{MT}[i, S, I, C, A] \neq \bot)$ then return $\bot$
7 If $(\text{UT}[i, S] \neq \bot)$ then $V \leftarrow \mathbf{G}^{\text{auth}}_{\mathsf{SE},q_s+q_v}.\text{VERIFY}(\text{UT}[i, S], I, C, A)$
8 Else
9 If $(\text{VT}[i, S] = \bot)$ then $c \leftarrow c + 1$; $\text{VT}[i, S] \leftarrow c$
10 $b \leftarrow \mathbf{G}^{\text{auth}}_{\mathsf{SE},q_s+q_v}.\text{VERIFY}(\text{VT}[i, S], I, C, A)$
11 Return false

Figure 21: How adversary A_{auth}, for the proof of Theorem 6.3, simulates A's oracles. SALT, H responses are as in Figure 15.

Adversary $A_{\text{auth-c}}$ is playing game $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},q_h}$ (Figure 5). It runs A, responding to oracle queries as shown in Figure 22. To each pair (P, S) where S is not one of the $S_{i,j}$, table VT associates a user index $\text{VT}[P, S] \in [1..q_h]$ (line 8 of Figure 22). At line 9, $A_{\text{auth-c}}$ calls the VERIFY oracle provided by the $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},q_h}$ game it is playing. The delicate issue is that $A_{\text{auth-c}}$ now needs to return $\text{HT}[P, S]$ to A in response to the H query, but this is a challenge key, not available to the adversary in the usual authenticity game. This is where exposures enter. At line 10, $A_{\text{auth-c}}$ exposes the key of

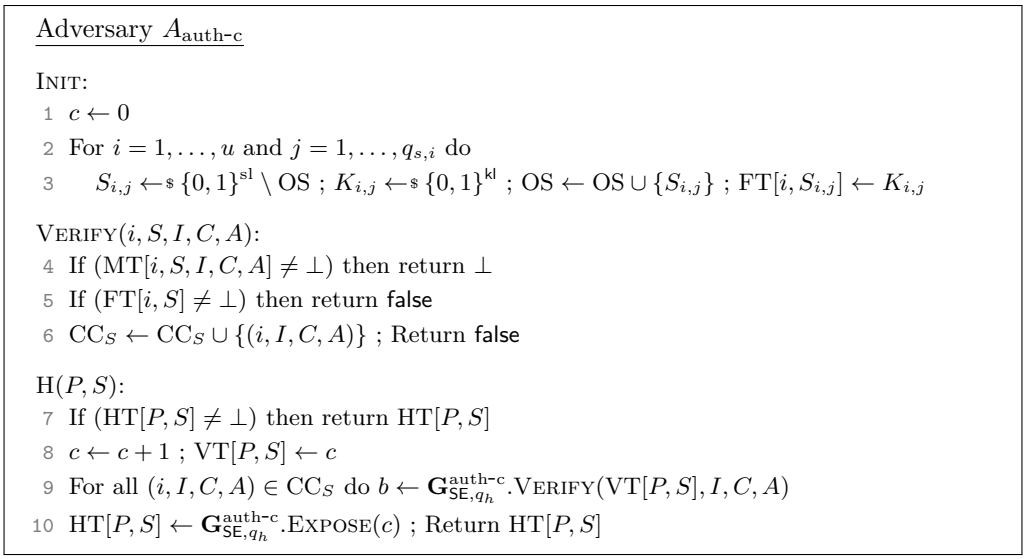

Figure 22: How adversary $A_{\text{auth-c}}$, for the proof of Theorem 6.3, simulates A's oracles. ENC, SALT responses are as in Figure 16.

---

the appropriate user and returns it to A as the oracle response. But the VERIFY queries at line 9 precede the EXPOSE query at line 10, so $A_{\text{auth-c}}$ sets win in $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},q_h}$ whenever $\text{G}_4(A)$ sets bad at line 18. This justifies Eq. (26). Note that $A_{\text{auth-c}}$ makes no queries to its $\mathbf{G}^{\text{auth-c}}_{\mathsf{SE},q_h}$.ENC oracle; replies to A's ENC queries are computed under the keys $K_{i,j}$ that $A_{\text{auth-c}}$ picked at line 3. This justifies Eq. (26), the final step in proving Eq. (6). ▌

D Proof of Theorem 7.1

Proof of Theorem 7.1: Figure 23 simultaneously specifies games $\text{G}_0, \text{G}_1, \text{G}_2$; a line annotated with a game name (eg. lines 2,3) is included only in the indicated game. We assume A respects requirements, allowing us to omit writing required conditions. In the case y = b we assume un is never set to false and accordingly drop it and associated variables. Game $\mathbf{G}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}$ is extended to the ROM by addition of a procedure H that implements a random oracle with range $\mathcal{R}$ as per the theorem statement.

We recall that by definition, the advantage $\mathbf{Adv}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A)$ may be written, equivalently, as $2\Pr[\mathbf{G}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A)] - 1$, or as $\Pr[\mathbf{G}^{\text{pind\$}}_{d=1}(A)] - \Pr[\mathbf{G}^{\text{pind\$}}_{d=0}(A)]$, where the notation $\mathbf{G}^{\text{pind\$}}_{d=d'}$ denotes the PIND\$ game (with parameters $\mathsf{FPBE}, \mathsf{PD}, u$) when operating with challenge bit d'. We refer to the latter advantage definition for this proof.

We first claim that G_1 is equivalent to $\mathbf{G}^{\text{pind\$}}_{d=1}$, and that G_0 is equivalent to $\mathbf{G}^{\text{pind\$}}_{d=0}$. Game G_1 returns a real ciphertext in line 2 and computes keys appropriately using the PBKDF in line 6, as expected for the PIND\$ game with challenge bit 1. Game G_0 returns a random ciphertext in line 3, as expected when the challenge bit is 0, and has no need to keep track of symmetric keys. Applying the definition of advantage,

$$\mathbf{Adv}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) = \Pr[\text{G}_1(A)] - \Pr[\text{G}_0(A)] \ .$$

Games G_0, G_1, G_2

INIT:
1 $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$

ENC(i, N, M, A):
2 $C \leftarrow \mathsf{SE.Enc}(K_{i,s(i)}, N, M, A)$ // Games $\mathrm{G}_1, \mathrm{G}_2$
3 $C \leftarrow\!\!\$\ \{0,1\}^{\mathsf{FPBE.cl}(|M|)}$ // Game G_0
4 Return C

SALT(i):
5 $s(i) \leftarrow s(i) + 1$
6 $S_{i,s(i)} \leftarrow\!\!\$\ \mathsf{FPBE.SS}$; $K_{i,s(i)} \leftarrow \mathsf{F}[\mathrm{H}](\mathbf{P}[i], S_{i,s(i)})$ // Game G_1
7 $S_{i,s(i)} \leftarrow\!\!\$\ \mathsf{FPBE.SS}$; $K_{i,s(i)} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{SE.kl}}$ // Game G_2
8 $S_{i,s(i)} \leftarrow\!\!\$\ \mathsf{FPBE.SS}$ // Game G_0
9 Return $S_{i,s(i)}$

H(X):
10 If $\mathrm{HT}[X] = \perp$ then $\mathrm{HT}[X] \leftarrow\!\!\$\ \mathcal{R}$
11 Return $\mathrm{HT}[X]$

FIN(d'):
12 Return $(d' = 1)$

Figure 23: Games for proof of Theorem 7.1.

We can trivially extend this expression to

$$\mathbf{Adv}^{\mathrm{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) = (\Pr[\mathrm{G}_1(A)] - \Pr[\mathrm{G}_2(A)]) + (\Pr[\mathrm{G}_2(A)] - \Pr[\mathrm{G}_0(A)]) \ .$$

We will design adversaries $A_{\mathsf{F}}, A_{\mathsf{SE}}$ such that

$$\Pr[\mathrm{G}_1(A)] - \Pr[\mathrm{G}_2(A)] \leq \mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \tag{27}$$

$$\Pr[\mathrm{G}_2(A)] - \Pr[\mathrm{G}_0(A)] \leq \mathbf{Adv}^{\mathrm{ind\$}}_{\mathsf{SE},q_s}(A_{\mathsf{SE}}) \ . \tag{28}$$

We first describe adversary A_{F}. Recall that in the kd game, described in Figure 10, the RIO oracle responses are applications of F with challenge bit 1, or are random with challenge bit 0. A_{F} can thus run A, responding to oracle queries as specified in either G_1 or G_2. Concretely, key and salt selection is implemented by doing $(S_i, K_{i,s(i)}) \leftarrow \mathrm{RIO}(i)$. When RIO returns the result of applying F, this is exactly line 6. If RIO returns random keys, this is line 7. A_{F} responds to A's encryption queries using the appropriate key in line 2. When A guesses a bit d' and terminates, A_{F} outputs d' as well.

As we did above with PIND\$ advantage, we can write the kd advantage of A_{F} as $\Pr[\mathbf{G}^{\mathrm{kd}}_{d=1}(A_{\mathsf{F}})] - \Pr[\mathbf{G}^{\mathrm{kd}}_{d=0}(A_{\mathsf{F}})]$, where the game parameters remain $\mathsf{F}, \mathsf{PD}, u$. Since A_{F} outputs the same d' as A, we have $\Pr[\mathbf{G}^{\mathrm{kd}}_{d=1}(A_{\mathsf{F}})] = \Pr[\mathrm{G}_1(A)]$ and $\Pr[\mathbf{G}^{\mathrm{kd}}_{d=0}(A_{\mathsf{F}})] = \Pr[\mathrm{G}_2(A)]$, which in combination with the definition of kd advantage, justifies Eq. (27).

Next we turn to describing adversary A_{SE}. A_{SE} initializes a user counter $v \leftarrow 0$, then runs A. When A makes a SALT(i) query, A_{SE} responds by doing:

$$s(i) \leftarrow s(i) + 1 \ ; \ v \leftarrow v + 1 \ ; \ S_i \leftarrow\!\!\$\ \mathsf{FPBE.SS} \ ; \ k_{i,s(i)} \leftarrow v$$

Return S_i // Response returned to A

When A makes an $\textsc{Enc}(i, N, M, A)$ query, A_{SE} responds by doing:

$C \leftarrow \mathbf{G}^{\text{ind\$}}_{\mathsf{SE},q_s}.\textsc{Enc}(k_{i,s(i)}, N, M, A)$ // Call own Enc oracle for user $k_{i,s(i)}$
Return C // Response returned to A

When A guesses a bit d' and ends execution, A_{SE} guesses that same bit. We claim that A_{SE} satisfies Eq. (28). The q_s parameter in the $\mathbf{Adv}^{\text{ind\$}}_{\mathsf{SE},q_s}(A_{\mathsf{SE}})$ term is explained as follows: a user in the $\mathbf{G}^{\text{ind\$}}_{\mathsf{SE},q_s}$ game corresponds to a new, uniformly random key whenever a Salt query occurs in the $\mathbf{G}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}$ game. Thus the number of keys in the $\mathbf{G}^{\text{ind\$}}_{\mathsf{SE},q_s}$ game is the number of Salt queries, q_s. To explain the rest, the oracle responses returned by A_{SE} conform to the game G_2 when A_{SE}'s challenge bit is 1, and A_{SE} correctly guesses 1 whenever A does. The oracle responses conform to game G_0 (random ciphertexts) when A_{SE}'s challenge bit is 0, and again A_{SE} guesses the same bit as A. Thus $\Pr[\mathbf{G}^{\text{ind\$}}_{d=1}(A_{\mathsf{SE}})] = \Pr[\mathrm{G}_2(A)]$ and $\Pr[\mathbf{G}^{\text{ind\$}}_{d=0}(A_{\mathsf{SE}})] = \Pr[\mathrm{G}_0(A)]$, where the IND\$ game parameters are SE, q_s. The definition of IND\$ advantage directly implies Eq. (28).

The theorem statement immediately follows from Eqs. (27), (28). We have

$$\begin{aligned}\mathbf{Adv}^{\text{pind\$}}_{\mathsf{FPBE},\mathsf{PD},u}(A) &= (\Pr[\mathrm{G}_1(A)] - \Pr[\mathrm{G}_2(A)]) + (\Pr[\mathrm{G}_2(A)] - \Pr[\mathrm{G}_0(A)]) \\ &\leq \mathbf{Adv}^{\text{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) + \mathbf{Adv}^{\text{ind\$}}_{\mathsf{SE},q_s}(A_{\mathsf{SE}}) \ .\end{aligned}$$

We additionally remark that A_{SE} preserves the adversary class of A, meaning that for $\mathrm{y} \in \{\mathrm{b}, \mathrm{a}\}$, $A \in \mathcal{A}_{\mathrm{y}}$ implies that $A_{\mathsf{SE}} \in \mathcal{A}_{\mathrm{y}}$. Recall that the user indices used by A and A_{SE} differ as follows: for a user i for which A makes queries of the form $\textsc{Enc}(i, N, M, A)$, A_{SE} may make queries to multiple SE users, corresponding to the FPBE resalts for user i. A_{SE} forwards the same nonce and other inputs, meaning that A_{SE} only repeats a nonce for a particular user if A does so for a particular user and salt. This proves the full statement of Theorem 7.1. ▮

E Proof of Theorem 7.2

Proof of Theorem 7.2: Game $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$ is extended to the ROM by addition of a procedure H that implements a random oracle with range $\mathcal{R}$ as per the theorem statement. With this as starting point, we refer to games $\mathrm{G}_0, \mathrm{G}_1$ in Figure 24. A line annotated with a game name (eg. lines 5,6) is included only in the indicated game. The notation $\mathsf{F}[\mathrm{H}]$ at lines 5,11 indicates that F may be a ROM PBKDF, calling H as an oracle. We are assuming A is sequential, so all its Enc and Salt queries are made before any of its Verify queries. We assume A respects requirements, allowing us to drop writing required conditions. In the case $\mathrm{y} = \mathrm{b}$ we assume un is never set to false and accordingly drop it and associated variables.

We claim that G_0 is equivalent to $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$, meaning that

$$\mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A) = \Pr[\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A)] = \Pr[\mathrm{G}_0(A)] \ .$$

Indeed G_0 maintains that $K_{i,s(i)} = \mathsf{F}[\mathrm{H}](\mathbf{P}[i], S_i) = \mathrm{KT}[i, S_i]$ and is using the right keys at all times.

Game G_1 switches the keys for the base scheme SE to random (lines 6,12). A subtle point is what happens if a salt S_i happens to equal a prior one for user i. Then $K_{i,s(i)}$ is nonetheless fresh, but $\mathrm{KT}[i, S_i]$ at line 13 in G_1 can get redefined to a new value. (The latter does not happen in G_0,

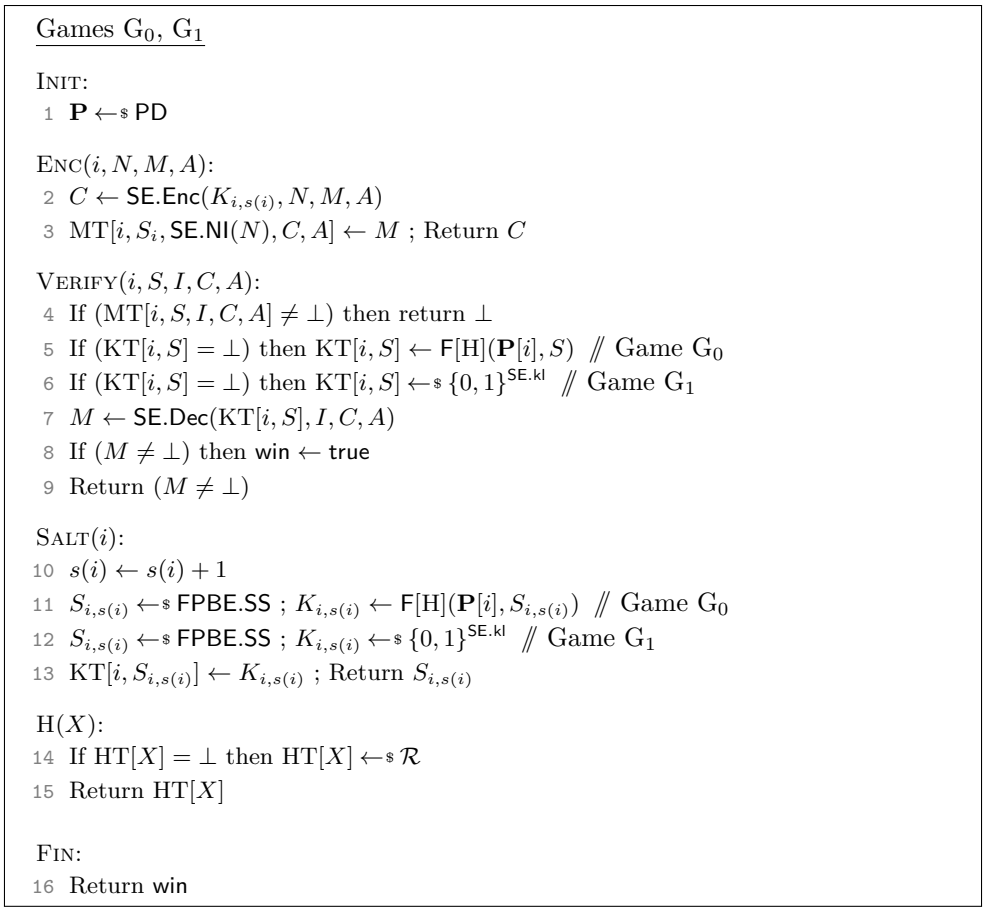

Figure 24: Games for proof of Theorem 7.2.

where $\mathrm{KT}[i, S_i]$, once defined, won't change, because F is deterministic.) However, $\mathrm{KT}[\cdot, \cdot]$ is only used in VERIFY, and since the adversary is sequential, responses will only reflect the latest $\mathrm{KT}[\cdot, \cdot]$ values and stay consistent with those. In particular, salt collisions create no "bad" event in the current context; instead this is covered through the definition of kd-security of F (salt collisions are allowed in RIO in Figure 10) and a salt-collision term shows up in Theorem 7.3.

Proceeding, we trivially have

$$\Pr[\mathrm{G}_0(A)] = \Pr[\mathrm{G}_1(A)] + \Pr[\mathrm{G}_0(A)] - \Pr[\mathrm{G}_1(A)] \ .$$

We will design adversaries $A_{\mathsf{F}}, A_{\mathsf{SE}}$ so that

$$\Pr[\mathrm{G}_0(A)] - \Pr[\mathrm{G}_1(A)] \leq \mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \tag{29}$$

$$\Pr[\mathrm{G}_1(A)] \leq \mathbf{Adv}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_v}(A_{\mathsf{SE}}) \ . \tag{30}$$

Adversary A_{F} runs A. When A makes a SALT(i) query, A_{F} responds as in game G_1 except that line 12 is replaced with $(S_{i,s(i)}, K_{i,s(i)}) \leftarrow \mathrm{RIO}(i)$, meaning the salt and key are obtained from A_{F}'s RIO oracle rather than being chosen directly. Now ENC queries can be answered as shown in G_1. For VERIFY queries, line 6 is replaced with $\mathrm{KT}[i, S] \leftarrow \mathrm{CIO}(i, S)$, and H queries are answered via A_{F}'s own H oracle. (Game $\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}$ here is in the ROM, providing oracle H because F uses it.)

When A terminates, A_{F} returns 1 if $\mathsf{win} = \mathsf{true}$ and 0 otherwise. If d denotes the challenge bit in game $\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}$ then

$$\Pr[\,\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \,|\, d = 1\,] = \Pr[\mathrm{G}_0(A)]$$
$$\Pr[\,\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \,|\, d = 0\,] = \Pr[\mathrm{G}_1(A)]\ .$$

Subtracting yields Eq. (29).

Next we describe adversary A_{SE}. A_{SE} initializes a counter $v \leftarrow 0$, representing a user index. It now runs A. When A makes a $\textsc{Salt}(i)$ query, it does the following:

$s(i) \leftarrow s(i) + 1$; $v \leftarrow v + 1$; $S_i \leftarrow\!\!\$\ \mathsf{FPBE.SS}$
$k_{i,s(i)} \leftarrow v$; $\mathrm{kT}[i, S_{i,s(i)}] \leftarrow v$ // Set these to user indices, not keys
Return S_i // Response returned to A

When A makes an $\textsc{Enc}(i, N, M, A)$ query, A_{SE} does the following:

$C \leftarrow \mathbf{G}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_v}.\textsc{Enc}(k_{i,s(i)}, N, M, A)$ // Call own Enc oracle for user $k_{i,s(i)}$
$\mathrm{MT}[i, S_i, \mathsf{SE.NI}(N), C, A] \leftarrow M$; Return C // Response returned to A

When A makes a $\textsc{Verify}(i, S, I, C, A)$ query, A_{SE} does the following:

If $(\mathrm{MT}[i, S, I, C, A] \neq \perp)$ then return $\perp$
If $(\mathrm{kT}[i, S] = \perp)$ then $v \leftarrow v + 1$; $\mathrm{kT}[i, S] \leftarrow v$
$e \leftarrow \mathbf{G}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_v}.\textsc{Verify}(\mathrm{kT}[i, S], I, C, A)$ // Call own Verify oracle for $\mathrm{kT}[i, S]$
Return e // Response returned to A

When A makes a $\mathrm{H}(X)$ query, A_{SE} does the following:

If $\mathrm{HT}[X] = \perp$ then $\mathrm{HT}[X] \leftarrow\!\!\$\ \mathcal{R}$
Return $\mathrm{HT}[X]$ // Response returned to A

Note that game $\mathbf{G}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_v}$ is not in the ROM and provides no H oracle; adversary A_{SE} is simply simulating it for A on its own. If the execution of A with G_1 sets win, then win will also be set in the execution of A_{SE} with $\mathbf{G}^{\mathrm{auth}}_{\mathsf{SE},q_s+q_v}$, which justifies Eq. (30). The count of $q_s + q_v$ for the number of users for SE arises as an upper bound for the counter v.

We also need to check that if $A \in \mathcal{A}_{\mathrm{b}}$ (a nonce is not repeated for a given user and salt instance) then $A_{\mathsf{SE}} \in \mathcal{A}_{\mathrm{b}}$ (a nonce is not repeated for a given user). This is true because we have taken care that every pair $(i, s(i))$ corresponds to a different user for A_{SE}. (This is true even if there are salt collisions.) A_{SE} is sequential because it makes all Enc, Verify queries in the same order as A. ▮

F Proof of Theorem 7.3

Proof of Theorem 7.3: In the ROM, game $\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}$ adds a procedure H for the random oracle. Below we will often, without explicit mention, exploit the assumption, from the definition of a password distribution, that $\mathbf{P}[1], \ldots, \mathbf{P}[u]$ are distinct. Similarly, we will exploit throughout the

Game G_0 / $\boxed{\mathrm{G}_1}$

INIT:
1 $d \leftarrow\!\!\$\ \{0,1\}$; $\mathbf{P} \leftarrow\!\!\$\ \mathsf{PD}$
2 For $i = 1, \ldots, u$ and $j = 1, \ldots, q_{r,i}$ do
3 $\quad S_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}}$
4 $\quad$ If $(d = 1)$ then $K_{i,j} \leftarrow \mathrm{H}(\mathbf{P}[i], S_{i,j})$ else $K_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$
5 $\quad$ If $(S_{i,j} \in \mathrm{OS})$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{S_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{sl}} \setminus \mathrm{OS}}$
6 $\quad \mathrm{OS} \leftarrow \mathrm{OS} \cup \{S_{i,j}\}$; $\mathrm{FT}[i, S_{i,j}] \leftarrow K_{i,j}$

RIO(i):
7 $s(i) \leftarrow s(i) + 1$; Return $(S_{i,s(i)}, K_{i,s(i)})$

CIO(i, S):
8 If $(\mathrm{FT}[i, S] \neq \perp)$ then return $\mathrm{FT}[i, S]$
9 $K_1 \leftarrow \mathrm{H}(\mathbf{P}[i], S)$; $K_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$; $\mathrm{FT}[i, S] \leftarrow K_d$; Return K_d

H(P, S):
10 If $(\mathrm{HT}[P, S] \neq \perp)$ then return $\mathrm{HT}[P, S]$
11 $\mathrm{HT}[P, S] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}}$; Return $\mathrm{HT}[P, S]$

FIN(d'):
12 Return $(d' = d)$

Figure 25: Games $\mathrm{G}_0, \mathrm{G}_1$ for proof of Theorem 7.3, where G_1 includes the boxed code and G_0 does not.

---

assumption that A_{F} is sequential, meaning it makes all its q_r RIO queries before any of its CIO queries. The algorithms Find1, Find2, used in some games and the constructed adversary below, were defined in Section 3.

Consider the games of Figure 25, where G_1 includes the boxed code and G_0 does not. We have let $q_{r,i}$ be the number of RIO(i) queries, so that $q_r = q_{r,1} + \cdots + q_{r,u}$. Let $S_{i,j}$ denote the input that RIO(i) would pick the j-th time it is called. The games start by picking these values up front in INIT, together with values $K_{i,j}$ that RIO would return as function outputs. However, while G_0 picks the $S_{i,j}$ at random, G_1 ensures that they are distinct. Down the line, this will allow password-guessing adversary A_{pg} to use the input to uniquely identify the user in an RIO query and thereby minimize the number of TEST queries it makes. For now, we just note that game G_0 is equivalent to game $\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}$, so

$$\mathbf{Adv}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) = 2\Pr[\mathbf{G}^{\mathrm{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}})] - 1 = 2\Pr[\mathrm{G}_0(A_{\mathsf{F}})] - 1 \ .$$

Trivially we have

$$2\Pr[\mathrm{G}_0(A_{\mathsf{F}})] - 1 = 2\Pr[\mathrm{G}_1(A_{\mathsf{F}})] - 1 + 2(\Pr[\mathrm{G}_0(A_{\mathsf{F}})] - \Pr[\mathrm{G}_1(A_{\mathsf{F}})]) \ . \tag{31}$$

Games $\mathrm{G}_0, \mathrm{G}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [14] we have

$$\Pr[\mathrm{G}_0(A_{\mathsf{F}})] - \Pr[\mathrm{G}_1(A_{\mathsf{F}})] \leq \Pr[\mathrm{G}_0(A_{\mathsf{F}}) \text{ sets } \mathsf{bad}] \ .$$

Flag bad is set when two of the q_r inputs collide, so

$$\Pr[\mathrm{G}_0(A_{\mathsf{F}}) \text{ sets } \mathsf{bad}] \leq \frac{q_r(q_r - 1)}{2^{\mathsf{sl}+1}} \ .$$

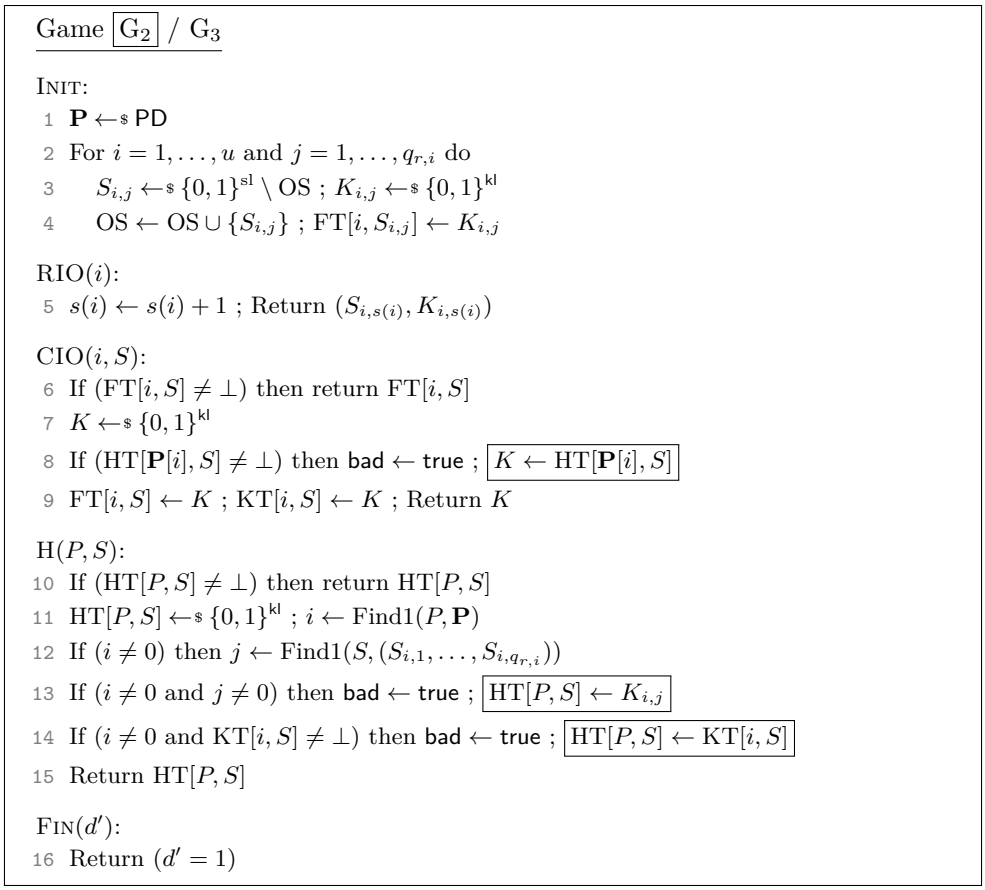

Figure 26: Games $\mathrm{G}_2, \mathrm{G}_3$ for proof of Theorem 7.3, where G_2 includes the boxed code and G_3 does not.

Returning to Eq. (31), we now want to bound the advantage of A_{F} in G_1, namely the quantity

$$2\Pr[\mathrm{G}_1(A_{\mathsf{F}})] - 1 = \Pr[\,\mathrm{G}_1(A_{\mathsf{F}}) \,|\, d = 1\,] - (1 - \Pr[\,\mathrm{G}_1(A_{\mathsf{F}}) \,|\, d = 0\,])\ ,$$

where d is the challenge bit from line 1 of Figure 25. Towards this, games $\mathrm{G}_2, \mathrm{G}_3$ in Figure 26 have been designed so that they are identical-until-bad and also

$$\Pr[\,\mathrm{G}_1(A_{\mathsf{F}}) \,|\, d = 1\,] = \Pr[\mathrm{G}_2(A_{\mathsf{F}})] \tag{32}$$

$$1 - \Pr[\,\mathrm{G}_1(A_{\mathsf{F}}) \,|\, d = 0\,] = \Pr[\mathrm{G}_3(A_{\mathsf{F}})]\ . \tag{33}$$

We now justify the two equations above. Unlike in G_1, oracle INIT in $\mathrm{G}_2, \mathrm{G}_3$ picks no challenge bit d, and, correspondingly, FIN(d') is changed to return true iff $d' = 1$. As in G_1, oracle INIT continues to pick distinct inputs $S_{i,j}$, but always optimistically picks the $K_{i,j}$ to be random. In G_3 it stays that way, as per the $d = 0$ case of G_1. As per the $d = 1$ case of G_1, however, G_2, via the inclusion of the boxed code at line 13, ensures that $K_{i,j} = \mathrm{H}(\mathbf{P}[i], S_{i,j})$. This shows that responses to RIO queries adhere to Eqs. (32) and (33). The boxed code at lines 8 and 14, included in G_2 but not G_3, ensures the same for CIO queries.

Games $\mathrm{G}_2, \mathrm{G}_3$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [14] we have

$$\Pr[\mathrm{G}_2(A_{\mathsf{F}})] - \Pr[\mathrm{G}_3(A_{\mathsf{F}})] \leq \Pr[\mathrm{G}_3(A_{\mathsf{F}}) \text{ sets } \mathsf{bad}]\ .$$

Game G_4

INIT:
1 $\mathbf{P} \xleftarrow{\$} \mathsf{PD}$
2 For $i = 1, \ldots, u$ and $j = 1, \ldots, q_{r,i}$ do
3 $\quad S_{i,j} \xleftarrow{\$} \{0,1\}^{\mathsf{sl}} \setminus \mathrm{OS}$; $K_{i,j} \xleftarrow{\$} \{0,1\}^{\mathsf{kl}}$
4 $\quad \mathrm{OS} \leftarrow \mathrm{OS} \cup \{S_{i,j}\}$; $\mathrm{FT}[i, S_{i,j}] \leftarrow K_{i,j}$

RIO(i):
5 $s(i) \leftarrow s(i) + 1$; Return $(S_{i,s(i)}, K_{i,s(i)})$

CIO(i, S):
6 If $(\mathrm{FT}[i, S] \neq \perp)$ then return $\mathrm{FT}[i, S]$
7 $\mathrm{CS} \leftarrow \mathrm{CS} \cup \{S\}$; $\mathrm{CU}_S \leftarrow \mathrm{CU}_S \cup \{i\}$
8 $K \xleftarrow{\$} \{0,1\}^{\mathsf{kl}}$; $\mathrm{FT}[i, S] \leftarrow K$; Return K

H(P, S):
9 If $(\mathrm{HT}[P, S] \neq \perp)$ then return $\mathrm{HT}[P, S]$
10 $\mathrm{HS} \leftarrow \mathrm{HS} \cup \{S\}$; $\mathrm{HP}_S \leftarrow \mathrm{HP}_S \cup \{P\}$
11 $\mathrm{HT}[P, S] \xleftarrow{\$} \{0,1\}^{\mathsf{kl}}$; Return $\mathrm{HT}[P, S]$

FIN(d'):
12 For all $S \in \mathrm{HS}$ and $P \in \mathrm{HP}_S$ do
13 $\quad i \leftarrow \mathrm{Find2}(S, (S_{1,1}, \ldots, S_{1,q_{r,1}}), \ldots, (S_{u,1}, \ldots, S_{u,q_{r,u}}))$
14 $\quad$ If $(i \neq 0$ and $P = \mathbf{P}[i])$ then $\mathsf{bad} \leftarrow \mathsf{true}$
15 For all $S \in \mathrm{CS} \cap \mathrm{HS}$ and $i \in \mathrm{CU}_S$ do
16 $\quad$ If $(\mathbf{P}[i] \in \mathrm{HP}_S)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
17 Return bad

Figure 27: Game G_4 for proof of Theorem 7.3.

Now we turn to bounding $\Pr[\mathrm{G}_3(A_{\mathsf{F}})$ sets $\mathsf{bad}]$. We claim that

$$\Pr[\mathrm{G}_3(A_{\mathsf{F}}) \text{ sets } \mathsf{bad}] \leq \Pr[\mathrm{G}_4(A_{\mathsf{F}})] \;, \tag{34}$$

where game G_4 is in Figure 27. Since the setting of bad in G_3 does not affect what oracles return, G_4 moves the setting of bad to FIN. Through line 7, the set CS holds all inputs S that were queried to CIO, and for each such S, set CU_S holds all users for which i, S was queried to CIO. Sets $\mathrm{HS}, \mathrm{HS}_S$ are analogously defined through line 10. These are used to set bad in FIN. Game G_4 sets bad at line 14 (respectively 16) whenever G_3 would have set it at line 13 (respectively 8 or 14), justifying Eq. (34). The distinctness of the $S_{i,j}$ inputs is important here to ensure that i at line 13 is unique.

Next we design A_{pg} so that

$$\Pr[\mathrm{G}_4(A_{\mathsf{F}})] \leq \mathbf{Adv}^{\mathrm{pg}}_{\mathsf{PD},u}(A_{\mathrm{pg}}) \;. \tag{35}$$

Adversary A_{pg} begins by executing lines 2–4 of G_4 in Figure 27. (It skips line 1. The role of $\mathbf{P}$ will be played by the one chosen in its game $\mathbf{G}^{\mathrm{pg}}_{\mathsf{PD},u}$.) It then runs A_{F}, simulating its $\mathrm{RIO}, \mathrm{CIO}, \mathrm{H}$ oracles as shown in game G_4. When A_{F} terminates (with an output d' that A_{pg} ignores), A_{pg} does the following:

1. For all $S \in \mathrm{HS}$ and $P \in \mathrm{HP}_S$ do

2. $i \leftarrow \text{Find2}(S, (S_{1,1}, \ldots, S_{1,q_{r,1}}), \ldots, (S_{u,1}, \ldots, S_{u,q_{r,u}}))$
3. If $(i \neq 0)$ then $\textsc{Test}(i, P)$
4. For all $S \in \text{CS} \cap \text{HS}$ and $i \in \text{CU}_S$ do
5. For all $P \in \text{HP}_S$ do $\textsc{Test}(i, P)$

If G_4 sets bad then some $\textsc{Test}$ query of A_{pg} will set win in game $\mathbf{G}^{\text{pg}}_{\mathsf{PD},u}$, justifying Eq. (35).

Putting things together, we have

$$\begin{aligned}\mathbf{Adv}^{\text{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) &= 2\Pr[\text{G}_0(A_{\mathsf{F}})] - 1 \leq 2\Pr[\text{G}_1(A_{\mathsf{F}})] - 1 + 2 \cdot \frac{q_r(q_r-1)}{2^{\text{sl}+1}} \\ &\leq \mathbf{Adv}^{\text{pg}}_{\mathsf{PD},u}(A_{\text{pg}}) + \frac{q_r(q_r-1)}{2^{\text{sl}}} \ .\end{aligned}$$

To obtain Eq. (11), we show that A_{pg}'s series of $\textsc{Test}$ queries is bounded by the parameters (q, q_p, q_w) defined in the theorem statement, so that $\mathbf{Adv}^{\text{kd}}_{\mathsf{F},\mathsf{PD},u}(A_{\mathsf{F}}) \leq \mathbf{GP}_{\mathsf{PD}}(q, q_p, q_w)$. We begin by considering q, the total number of unique $\textsc{Test}$ queries.

In the above code, the total number of $\textsc{Test}$ queries emanating from line 3 is at most q_h. The number from line 5 is

$$\sum_{S \in \text{CS} \cap \text{HS}} \sum_{i \in \text{CU}_S} |\text{HP}_S| = \sum_{S \in \text{CS} \cap \text{HS}} |\text{CU}_S| \cdot |\text{HP}_S| \ . \tag{36}$$

We can bound this in two ways. First, $|\text{CU}_S| \leq u$ so the sum is at most $u \cdot \sum_{S \in \text{HS}} |\text{HP}_S| = uq_h$. Second, $|\text{HP}_S| \leq q_h$ so the sum is at most $q_h \cdot \sum_{S \in \text{CS}} |\text{CU}_S| = q_h q_c$. Thus the number of $\textsc{Test}$ queries arising from line 5 is $\min(q_c, u) \cdot q_h$. Additionally there are q_h $\textsc{Test}$ queries if $q_r > 0$, and none if $q_r = 0$. So overall, A_{pg} makes at most $q = \mathsf{zt}(q_r) \cdot q_h + \min(q_c, u) \cdot q_h$ $\textsc{Test}$ queries, as claimed. Now, the number of distinct passwords guessed among the $\textsc{Test}$ queries (the q_p guessing probability parameter) is at most q_h. And the number of distinct users included among the $\textsc{Test}$ queries (the q_w parameter) is at most $\min(q_r + q_c, u)$.

We note that the reason CIO queries are more expensive in terms of $\textsc{Test}$ queries is that A_{pg} does not have a way to know for which user to test a particular P, forcing it to try all the ones in CU_S. We avoided this for RIO queries by making the $S_{i,j}$ inputs distinct, so that the input would uniquely identify the user. This is how we have made RIO queries cheaper in terms of $\textsc{Test}$ queries. ▮

G Proofs of attack propositions

Proof of Proposition 8.1: Adversary A queries $S_i \leftarrow \textsc{Salt}(i)$ for $i = 1, \ldots, u$. Then it picks a nonce N, an ℓ-bit message M and associated data A. It lets $C_i \leftarrow \textsc{Enc}(i, N, M, A)$ for $i = 1, \ldots, u$. It then runs A_{pg}. When the latter makes query $\textsc{Test}(i, P)$, it does the following:

$K \leftarrow \text{H}(P, S_i)$; $C \leftarrow \mathsf{SE.Enc}(K, N, M, A)$
If $(C = C_i)$ then $\textsc{Fin}(1)$

If the execution of A_{pg} terminates without the "If" above ever returning true, then A ends with $\textsc{Fin}(0)$. If A_{pg} makes q_h $\textsc{Test}$ queries then A makes q_h H queries and u $\textsc{Salt}, \textsc{Enc}$ queries, as specified in Proposition 8.1.

Next we consider the advantage of A. If any of $A_{\rm pg}$'s $\textsc{Test}(i,P)$ queries is correct, meaning that $\mathbf{P}[i]=P$, then A's $C=C_i$ check will return true. In the $d=1$ setting this means that

$$\Pr[\mathbf{G}^{\rm pind\$}_{\mathsf{FPBE},\mathsf{PD},u}(A) \mid d=1] \geq \mathbf{Adv}^{\rm pg}_{\mathsf{PD},u}(A_{\rm pg})\ . \tag{37}$$

In the $d=0$ setting, the ciphertexts are random (independent of the password guesses). A will only return $\textsc{Fin}(1)$ if a random ciphertext happens to decrypt to the correct message for at least one of the q_h tries. For one try, this probability is $\frac{1}{2^{\mathsf{SE.cl}(\ell)}}$. Over all tries this means

$$\Pr[\mathbf{G}^{\rm pind\$}_{\mathsf{FPBE},\mathsf{PD},u}(A) \mid d=0] \geq 1-\frac{q_h}{2^{\mathsf{SE.cl}(\ell)}}\ . \tag{38}$$

Combining Eqs. (37), (38) gives the advantage for A. We note that A does not misuse nonces and thus can be considered in the class $\mathcal{A}_y$ for either of $\mathrm{y}\in\{\mathrm{b},\mathrm{a}\}$. ▮

Partitioning-oracle attack. Before presenting the proof of Proposition 8.2, which uses the partitioning-oracle attack, we briefly summarize its context. As introduced in [29], the basic partitioning-oracle attack targets one specific user. The attack utilizes a function MakeSplittingCT which does the following: On inputs $K_1,\ldots,K_n\in\{0,1\}^{\mathsf{SE.kl}}$ for $n>0$, MakeSplittingCT returns $(I\in\mathsf{SE.NIS}, C\in\{0,1\}^*, A\in\mathsf{SE.AS})$ such that $\mathsf{SE.Dec}(K_i,I,C,A)\neq\perp$ for all $1\leq i\leq n$. Importantly, this violates key-robustness with advantage 1, for an arbitrary number n of symmetric keys.

The description of partitioning oracles in the following proof differs slightly, though the algorithm remains essentially the same, in order to provide comparison to our theorems. First, it is multi-user, considering u users rather than one. Second, it considers the more general γ-way robustness. Finally, we aim to break authenticity rather than mount a full password-recovery attack, as [29] does. Like [29], we focus on a setting with zero $\textsc{Enc}$ queries and access to a $\textsc{Verify}$ oracle. We now proceed to the proof.

Proof of Proposition 8.2: Adversary $A_{\rm po}$ begins by running $A_{\rm pg}$ to obtain its sequence $(i_1,g_1),\ldots,(i_q,g_q)$ of guesses. Since the $\textsc{Test}$ oracle sends no response, all of $A_{\rm pg}$'s queries can be made and recorded up front. Let $q=n(1)+\cdots+n(u)$, so that $n(i)$ is the number of $\textsc{Test}$ queries to user $i\in[1..u]$. $A_{\rm po}$ then re-indexes the guesses so that $g_{i,j}$ is the j-th guess to user i, where $0\leq j\leq n(i)-1$ and $1\leq i\leq u$. It fixes a salt $S\in\mathsf{FPBE.SS}$ and lets $K_{i,j}\leftarrow \mathrm{H}(S,g_{i,j})$. Note that in the proposition statement, we assume that $A_{\rm pg}$ makes (q,q_h,q_w) $\textsc{Test}$ queries, and in particular this covers q_h distinct password guesses. Thus $A_{\rm po}$ need only query H q_h times, since the salt remains fixed.

In the proposition statement we assume that robustness is completely violated, meaning that an adversary $A_{\rm krob\$}$ violates γ-way robustness with advantage 1 for arbitrarily large γ. This is the assumption in [29], but for now, suppose only that $\gamma\geq 2$ and that we are given $A_{\rm krob\$}$ achieving some advantage. Let $q(i)=\lceil n(i)/\gamma\rceil$ for $1\leq i\leq u$. Let $q_v=q(1)+\cdots+q(u)$ and $U=\{\,i\in[1..u]: n(i)\geq\gamma\,\}$.

Now for each i, $A_{\rm po}$ splits the sequence $K_{i,0},\ldots,K_{i,n(i)-1}$ into $q(i)$ sub-sequences, where the first $q(i)-1$ have size γ and the last has size at most γ. The robustness adversary is run on each of the first $q(i)-1$ sub-sequences. This is detailed below:

For all $i\in U$ do
  For $\ell=1,\ldots,q(i)-1$ do
    $(I,C,A)\leftarrow A_{\rm krob\$}(K_{i,\gamma(\ell-1)},\ldots,K_{i,\gamma\ell-1})$; $\textsc{Verify}(i,S,I,C,A)$

If the $q(i)$-th sub-sequence has size γ, the loop continues for one more iteration and $A_{\rm po}$ does:

$$(I, C, A) \leftarrow A_{\text{krob\$}}(K_{i,\gamma(q(i)-1)}, \ldots, K_{i,\gamma q(i)-1}) \ ; \ \textsc{Verify}(i, S, I, C, A)$$

Otherwise, if the $q(i)$-th sub-sequence has size smaller than γ, the difficulty is that the robustness adversary $A_{\text{krob\$}}$ runs in a setting with γ keys. In this case, A_{po} chooses remaining keys uniformly at random, so that the $q(i)$-th subsequence has γ keys and the above line can be executed. We claim that this does not change A_{po}'s advantage lower bound, and in fact can only *increase* the chance that a VERIFY query correctly decrypts. We also note about $A_{\text{krob\$}}$ that the γ keys given as input are always random, as expected in the INIT procedure of the γ-way robustness game (as defined in Figure 6), either from being chosen randomly or as the output of the random oracle H.

At this point, A_{po} has made its q_h H queries and q_v VERIFY queries; the ceiling in $q(i) = \lceil n(i)/\gamma \rceil$ ensures that the number of VERIFY queries accounts for the second case above. We proceed to explaining Eq. (13). In order for A_{po} to make a VERIFY query that sets win to true in its $\mathbf{G}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}$ game, two conditions must hold: first, one of the keys $K_{i,j}$ is such that $\mathrm{H}(\mathbf{P}[i], S) = K_{i,j}$ for user i; and second, $A_{\text{krob\$}}$ "succeeds" on this particular key. That is, in that iteration, $A_{\text{krob\$}}$ returns (I, C, A) such that $\textsc{Verify}(i, S, I, C, A)$ wins and thus $\mathsf{SE.Dec}(K_{i,j}, I, C, A) \neq \perp$.

To express the requirement that both a password guess be correct, and that $A_{\text{krob\$}}$ find a robustness-violating ciphertext for that particular guess, the following is a lower bound, multiplying the required probabilities:

$$\mathbf{Adv}^{\text{pauth}}_{\mathsf{FPBE},\mathsf{PD},u}(A_{\text{po}}) \geq \mathbf{Adv}^{\text{pg}}_{\mathsf{PD},u}(A_{\text{pg}}) \cdot \prod_{i \in U} \mathbf{Adv}^{\text{krob\$}}_{\mathsf{SE},\gamma,\gamma}(A_{\text{krob\$}})^{q(i)} \ . \tag{39}$$

While Eq. (39) is a general statement, it is most useful in the setting of the proposition, when $A_{\text{krob\$}}$ violates γ-way robustness with advantage 1 for arbitrarily large γ. In this case, the righthand robustness advantage product is simply 1, and we achieve Eq. (13). We again note that A_{po} makes q_h H queries, and because of the robustness assumption, at most one VERIFY query per user. Given that A_{pg} makes (q, q_h, q_w) TEST queries, A_{po} thus makes $\min(q_w, u)$ VERIFY queries, proving the remainder of the proposition statement. We also note that A_{po} does not misuse nonces and thus can be considered in the class $\mathcal{A}_y$ for either of $\mathrm{y} \in \{\mathrm{b}, \mathrm{a}\}$. ∎

H Proof of Proposition 9.1

Proof of Proposition 9.1: Given adversary A, we begin with the construction of A_{SE}. Adversary A_{SE} runs A to get $(P_1, S_1, N_1, A_1, M_1), \ldots, (P_\gamma, S_\gamma, N_\gamma, A_\gamma, M_\gamma)$, then outputs $(\mathsf{F}(P_1, S_1), N_1, A_1, M_1), \ldots, (\mathsf{F}(P_\gamma, S_\gamma), N_\gamma, A_\gamma, M_\gamma)$. If all of A's outputs encrypt to the same ciphertext, then so will A_{SE}'s outputs; this follows from the fact that $\mathsf{FPBE} = \mathbf{DtE}[\mathsf{SE}, \mathsf{F}]$ derives symmetric keys by doing exactly $\mathsf{F}(P, S)$. However, it is not the case that the distinctness of $P_1, \ldots, P_\gamma$ (if $\ell = 1$) or the distinctness of $(P_1, S_1), \ldots, (P_\gamma, S_\gamma)$ (if $\ell = 2$) implies the distinctness of $\mathsf{F}(P_1, S_1), \ldots, \mathsf{F}(P_\gamma, S_\gamma)$. At this point, we can say that

$$\mathbf{Adv}^{\text{pcmt-}\ell}_{\mathsf{FPBE},\gamma}(A) \leq \mathbf{Adv}^{\text{cmt-}\ell'}_{\mathsf{SE},\gamma}(A_{\mathsf{SE}}) + \Pr[\mathsf{F}(P_1, S_1), \ldots, \mathsf{F}(P_\gamma, S_\gamma) \text{ are not distinct}] \ .$$

Note that the above expression holds for all three cases claimed in the theorem: namely $\ell = 1$ to $\ell' = 1$ (trivially), $\ell = 2$ to $\ell' = 1$, and $\ell = 5$ to $\ell' = 4$.

Next we turn to A_{F}. A_{F} runs A to get $(P_1, S_1, N_1, A_1, M_1), \ldots, (P_\gamma, S_\gamma, N_\gamma, A_\gamma, M_\gamma)$. Then A_{F} scans through the γ tuples to find i, j such that $i \neq j$ and $\mathsf{F}(P_i, S_i) = \mathsf{F}(P_j, S_j)$. If A_{F} finds such a pair, it outputs $(P_i, S_i), (P_j, S_j)$, and otherwise outputs $\perp$. Thus A_{F} wins the cr game if and only

if A's outputs are such that $\mathsf{F}(P_1, S_1), \ldots, \mathsf{F}(P_\gamma, S_\gamma)$ are not distinct. This, along with the above equation, proves Eq. (14). ▮