A New Framework for Password-Based Authenticated Key Exchange

Adam Groce[*] Jonathan Katz[*]

Abstract

Protocols for password-based authenticated key exchange (PAKE) allow two users who share only a short, low-entropy password to agree on a cryptographically strong session key. The challenge in designing such protocols is that they must be immune to *off-line dictionary attacks* in which an eavesdropping adversary exhaustively enumerates the dictionary of likely passwords in an attempt to match a password to the set of observed transcripts.

To date, few general frameworks for constructing PAKE protocols in the standard model are known. Here, we abstract and generalize a protocol proposed by Jiang and Gong to give a new methodology for realizing PAKE without random oracles, in the common reference string model. In addition to giving a new approach to the problem, the resulting framework offers several advantages over prior work.

1 Introduction

Protocols for password-based authenticated key exchange (PAKE) enable two parties who share a short, low-entropy password to agree on a cryptographically strong session key. The difficulty in this setting is to design protocols preventing *off-line dictionary attacks* whereby an eavesdropping adversary exhaustively enumerates passwords, attempting to match the correct password to observed protocol executions. Roughly, a PAKE protocol is "secure" if off-line attacks are of no use and the best attack is an on-line dictionary attack whereby the adversary tries to impersonate the honest user with each possible password. This is the best that can be hoped for in the password-only setting; more importantly, on-line attacks can be detected and defended against.

PAKE protocols are fascinating from a theoretical perspective, as they can be viewed as a means of "bootstrapping" a common cryptographic key from the (essentially) minimal setup assumption of a short, shared secret. PAKE protocols are also important in practice, since passwords are perhaps the most common and widely-used means of authentication.

There is, by now, a substantial body of research focused on the design of PAKE protocols. Early work [14] (see also [15]) considered a "hybrid" model where users share public keys in addition to a password; we are concerned in this paper with the more challenging "password-only" setting. Bellovin and Merritt [6] initiated research in this direction, and presented a PAKE protocol with heuristic arguments for its security. It was not until several years later that formal models for PAKE were developed [3, 7, 13], and provably secure PAKE protocols were shown in the random oracle/ideal cipher models [3, 7, 21].

[*]Dept. of Computer Science, University of Maryland. Email: {agroce, jkatz}@cs.umd.edu. Research supported by NSF grant #0627306.

To date, there are only a few general approaches for constructing PAKE protocols in the *standard model* (i.e., without random oracles). Goldreich and Lindell [13] constructed the first such PAKE protocol in the so-called "plain model" where there is no additional setup. Unfortunately, their protocol is inefficient in terms of communication, computation, and round complexity; furthermore, it does not tolerate concurrent executions by the same party. Nguyen and Vadhan [22] show some simplifications and efficiency improvements to the Goldreich-Lindell protocol, but at the expense of achieving a qualitatively weaker notion of security. The results of Barak et al. [2] also imply a protocol for password-based key exchange, albeit in the common reference string model. None of the above approaches appear to yield anything close to a practical instantiation.

Katz, Ostrovsky, and Yung (KOY) [18] demonstrated the first *efficient* PAKE protocol with a proof of security in the standard model. Their protocol was later abstracted by Gennaro and Lindell (GL) [12], who gave a general framework that encompasses the original KOY protocol as a special case. These protocols are secure even under concurrent executions by the same party, but they require a *common reference string* (CRS). While this may be less appealing than the "plain model," reliance on a CRS does not appear to be a serious drawback in practice for the deployment of PAKE, where common parameters can be hard-coded into an implementation of the protocol.

Surprisingly, the KOY/GL framework remains the only *general* framework for constructing *efficient* PAKE protocols in *the standard model*, and it is fair to say that almost subsequent work on efficient PAKE in the standard model [12, 8, 17, 11, 1, 19] can be viewed as extending and building on the KOY/GL framework. The one exception is a paper by Jiang and Gong [16] that shows an efficient PAKE protocol in the standard model (assuming a common reference string) based on the decisional Diffie-Hellman assumption. Our work is to theirs as the work of Gennaro-Lindell [12] is to that of Katz-Ostrovsky-Yung [18]; namely, we present a (new) framework for PAKE that is obtained by suitably abstracting and generalizing the Jiang-Gong protocol. In so doing, we gain the same benefits as in the previous case: i.e., we get a simple-to-describe, generic protocol with a clean and intuitive proof of security, and derive (as corollaries to our work) new variants of the Jiang-Gong protocol based on different cryptographic assumptions.

More compellingly, as compared to PAKE protocols built using the KOY/GL framework we obtain several advantages, described now:

Weaker assumptions. From a foundational point of view, the new framework relies on potentially *weaker* assumptions than the KOY/GL framework. Specifically, we require (1) a CCA-secure encryption scheme as well as (2) a *CPA*-secure encryption scheme with an associated smooth projective hash function [10]. In contrast, the framework from [12] requires[1] a *CCA*-secure encryption scheme with an associated smooth projective hash function, something not known to follow from the previous assumptions.[2]

In particular, our results imply a more efficient — not to mention simpler — construction of PAKE from lattice-based assumptions as compared to the recent work of [19]. (This is because most of the complexity in [19] arises from the need to construct a lattice-based CCA-secure encryption scheme with an associated smooth projective hash function.)

Better efficiency. The above directly translates into better efficiency for protocols constructed using the new framework: first, because CCA-secure encryption is used in only one round (as

[1]Technically speaking, they require non-malleable, non-interactive *commitment* with an associated smooth projective hash function. However, all known constructions of this primitive are in fact CCA-secure encryption schemes.

[2]Cramer and Shoup [10] show that a CPA-secure encryption scheme Π with a smooth projective hash function implies a CCA-secure scheme Π', but there is no guarantee that Π' will *itself* admit a smooth projective hash function.

compared to the protocols of [18, 12] which use CCA-secure encryption in two rounds); second, since the CCA-secure encryption scheme used need not admit a smooth projective hash function. (E.g., restricting our attention to the decisional Diffie-Hellman assumption, our framework can use the Kurosawa-Desmedt [20] scheme instead of Cramer-Shoup encryption [10].) The new framework also avoids using digital signatures (though the recent work of Gennaro [11] shows how this can be avoided when using the KOY/GL framework as well).

Mutual authentication. The framework yields PAKE protocols achieving *mutual authentication* in only three rounds. In contrast, the KOY protocol and its extensions require four rounds in order to achieve mutual authentication. (This advantage was already noted in [16].)

1.1 Outline of the Paper

We review definitions for PAKE and smooth projective hashing in Sections 2.1 and 2.2, respectively; these are fairly standard and can be skipped by readers already familiar with these notions. In Section 3 we describe the new framework for PAKE; we prove security of this framework in Section 4.

2 Definitions

2.1 Password-Based Authenticated Key Exchange

We present the definition of Bellare, Pointcheval, and Rogaway [3], based on prior work of [4, 5]. The treatment here is lifted almost verbatim from [18], except that here we also define mutual authentication. We denote the security parameter by n.

Participants, passwords, and initialization. Prior to any execution of the protocol there is an initialization phase during which public parameters are established. We assume a fixed set User of protocol participants (also called principals or users). For every distinct $U, U' \in \mathsf{User}$, we assume U and U' share a password $\pi_{U,U'}$. We make the simplifying assumption that each $\pi_{U,U'}$ is chosen independently and uniformly at random from the set $\{1, \ldots, D_n\}$ for some integer D_n that may depend on n. (Our proof of security extends to more general cases.)

Execution of the protocol. In the real world, a protocol determines how principals behave in response to input from their environment. In the formal model, these inputs are provided by the adversary. Each principal can execute the protocol multiple times (possibly concurrently) with different partners; this is modeled by allowing each principal to have an unlimited number of *instances* with which to execute the protocol. We denote instance i of user U as Π_U^i. Each instance may be used only once. The adversary is given oracle access to these different instances; furthermore, each instance maintains (local) state which is updated during the course of the experiment. In particular, each instance Π_U^i has associated with it the following variables:

- sid_U^i, pid_U^i, and sk_U^i denote the *session id*, *partner id*, and *session key* for an instance, respectively. The session id is simply a way to keep track of different executions; we let sid_U^i be the (ordered) concatenation of all messages sent and received by Π_U^i. The partner id denotes the user with whom Π_U^i believes it is interacting. (Note that pid_U^i can never equal U.)
- acc_U^i and term_U^i are boolean variables denoting whether a given instance has accepted or terminated, respectively.

The adversary's interaction with the principals (more specifically, with the various instances) is modeled via access to *oracles* that we describe now:

- $\mathsf{Send}(U, i, M)$ — This sends message M to instance Π_U^i. This instance runs according to the protocol specification, updating state as appropriate. The output of Π_U^i (i.e., the message sent by the instance) is given to the adversary.
- $\mathsf{Execute}(U, i, U', j)$ — If Π_U^i and $\Pi_{U'}^j$ have not yet been used, this oracle executes the protocol between these instances and gives the transcript of this execution to the adversary. This oracle call represents passive eavesdropping of a protocol execution.
- $\mathsf{Reveal}(U, i)$ — This outputs the session key sk_U^i, modeling leakage of session keys due to, e.g., improper erasure of session keys after use, compromise of a host computer, or cryptanalysis.
- $\mathsf{Test}(U, i)$ — This oracle does not model any real-world capability of the adversary, but is instead used to define security. A random bit b is chosen; if $b = 1$ the adversary is given sk_U^i, and if $b = 0$ the adversary is given a session key chosen uniformly from the appropriate space.

Partnering. Let $U, U' \in \mathsf{User}$. Instances Π_U^i and $\Pi_{U'}^j$ are *partnered* if: (1) $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j \neq \text{NULL}$; and (2) $\mathsf{pid}_U^i = U'$ and $\mathsf{pid}_{U'}^j = U$.

Correctness. To be viable, a key-exchange protocol must satisfy the following notion of correctness: if Π_U^i and $\Pi_{U'}^j$ are partnered then $\mathsf{acc}_U^i = \mathsf{acc}_{U'}^j = \text{TRUE}$ and $\mathsf{sk}_U^i = \mathsf{sk}_{U'}^j$, i.e., they both accept and conclude with the same session key.

Advantage of the adversary. Informally, the adversary can succeed in two ways: (1) if it guesses the bit b used by the Test oracle, or (2) if it causes an instance to accept without there being a corresponding partner. Defining this formally requires dealing with several technicalities.

We first define a notion of *freshness*. An instance Π_U^i is *fresh* unless one of the following is true at the conclusion of the experiment: (1) at some point, the adversary queried $\mathsf{Reveal}(U, i)$; or (2) at some point the adversary queried $\mathsf{Reveal}(U', j)$, where $\Pi_{U'}^j$ and Π_U^i are partnered.

We also define a notion of *semi-partnering*. Instances Π_U^i and $\Pi_{U'}^j$ are *semi-partners* if they are partners, or if the following holds: (1) the (non-NULL) session ids sid_U^i and $\mathsf{sid}_{U'}^j$ agree except possibly for the final message, and $\mathsf{pid}_U^i = U'$ and $\mathsf{pid}_{U'}^j = U$. This relaxed definition is needed to rule out the trivial attack where an adversary forwards all protocol messages except the final one.

An adversary $\mathcal{A}$ *succeeds* if either:

1. $\mathcal{A}$ makes a single query $\mathsf{Test}(U, i)$ to a fresh instance Π_U^i, and outputs a bit b' with $b' = b$ (recall that b is the bit chosen by the Test oracle).
2. At the end of the experiment, there is an instance Π_U^i that accepts but is not semi-partnered with any other instance.

We denote this event by Succ. The *advantage* of adversary $\mathcal{A}$ in attacking protocol Π is given by $\mathsf{Adv}_{\mathcal{A},\Pi}(k) \overset{\text{def}}{=} 2 \cdot \Pr[\mathsf{Succ}] - 1$, where the probability is taken over the random coins used by the adversary and during the course of the experiment (including the initialization phase).

It remains to define a secure protocol. A probabilistic polynomial-time (PPT) adversary can always succeed with probability 1 by trying all passwords one-by-one; this is possible since the size of the password dictionary is small. Informally, a protocol is secure if this is the best an adversary can do. Formally, an instance Π_U^i represents an *on-line attack* if, at some point, the adversary queried $\mathsf{Send}(U, i, *)$. The number of on-line attacks represents a bound on the number of passwords the adversary could have tested in an on-line fashion.

Definition 1 Protocol Π is a secure protocol for password-based authenticated key exchange if, for all dictionary sizes $\{D_n\}$ and for all PPT adversaries $\mathcal{A}$ making at most $Q(n)$ on-line attacks, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\mathsf{Adv}_{\mathcal{A},\Pi}(n) \leq Q(n)/D_n + \mathsf{negl}(n)$. $\diamondsuit$

2.2 Smooth Projective Hashing

Smooth projective hash functions were introduced by Cramer and Shoup [9]; we follow (and adapt) the treatment of Gennaro and Lindell [12], who extend the original definition. Rather than aiming for utmost generality, we tailor the definitions to our eventual application.

Fix a CPA-secure public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and an efficiently recognizable message space $\mathcal{D}$ (that will correspond to the dictionary of possible passwords in our application to PAKE). We assume the encryption scheme defines a notion of *ciphertext validity* such that (1) validity of a ciphertext (with respect to pk) can be determined efficiently using pk alone, and (2) all honestly generated ciphertexts are valid.

For the rest of the discussion, fix a key pair (pk, sk) as output by $\mathsf{Gen}(1^n)$ and let $\mathcal{C}$ denote the set of valid ciphertexts with respect to pk. Define sets X, $\{L_m\}_{m\in\mathcal{D}}$, and L as follows. First, set

$$X = \{(C, m) \mid C \in \mathcal{C};\ m \in \mathcal{D}\}.$$

For $m \in \mathcal{D}$ let $L_m = \{(C, m) \mid \mathsf{Dec}_{sk}(C) = m\} \subset X$; i.e., L_m is the set of ciphertext/message pairs. Define $L = \bigcup_{m\in\mathcal{D}} L_m$. Note that for any C there is at most one $m \in \mathcal{D}$ for which $(C, m) \in L$.

Smooth projective hash functions. A *smooth projective hash function* is a collection of keyed functions $\{H_k : X \to \{0,1\}^n\}_{k\in K}$, along with a *projection function* $\alpha : K \times \mathcal{C} \to S$, satisfying notions of *correctness* and *smoothness*:

Correctness: If $x = (C, m) \in L$ then the value of $H_k(x)$ is determined by $\alpha(k, C)$ and x (in a sense we will make precise below).

Smoothness: If $x \in X \setminus L$ then the value of $H_k(x)$ is statistically close to uniform given $\alpha(k, C)$ and x (assuming k is chosen uniformly in K).

Formally, a smooth projective hash function is defined by a sampling algorithm that, given pk, outputs $(K, \mathcal{H} = \{H_k : X \to \{0,1\}^n\}_{k\in K}, S,\ \alpha : K \times \mathcal{C} \to S)$ such that:

1. There are efficient algorithms for (1) sampling a uniform $k \in K$, (2) computing $H_k(x)$ for all $k \in K$ and $x \in X$, and (3) computing $\alpha(k, C)$ for all $k \in K$ and $C \in \mathcal{C}$.

2. For $x = (C, m) \in L$, the value of $H_k(x)$ is determined by $\alpha(k, C)$. Specifically, there is an efficient algorithm H' that takes as input $s = \alpha(k, C)$ and $\bar{x} = (C, m, r)$ (where r is such that $C = \mathsf{Enc}_{pk}(m; r)$) and satisfies $H'(s, \bar{x}) = H_k(x)$.

3. For any $x = (C, m) \in X \setminus L$, the distributions

$$\big\{k \leftarrow K; s = \alpha(k, C) : \big(s, H_k(x)\big)\big\} \text{ and } \{k \leftarrow K; s = \alpha(k, C); v \leftarrow \{0,1\}^n : (s, v)\}$$

have statistical difference negligible in n.

3 A New Framework for PAKE

We now describe the new framework for PAKE, obtained as a generalization and abstraction of the specific protocol by Jiang and Gong [16]. In our construction, we use the following primitives:

- A CPA-secure public-key encryption scheme $\Sigma' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ that has an associated smooth projective hash function as defined in Section 2.2.
- A *labeled* [23] CCA-secure public-key encryption scheme $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

We now describe the protocol.

Initialization. Our protocol relies on a common reference string (CRS) consisting of public keys pk, pk' for Σ and Σ', respectively, along with parameters $(K, \mathcal{H} = \{H_k : X \to \{0,1\}^n\}_{k \in K}, S, \alpha : K \times \mathcal{C} \to S)$ for a smooth projective hash function associated with pk'. We stress that, as in all other work in the CRS model, no participants need to know the secret keys associated with the public keys in the CRS. Furthermore, depending on the exact public-key encryption schemes used it is possible that the necessary public keys could be generated via a common *random* string.

$$\begin{array}{rcl}
\multicolumn{3}{c}{\text{CRS: } pk, pk'} \\
\underline{\Pi_U^i} & & \underline{\Pi_{U'}^j} \\
r \leftarrow \{0,1\}^* & & \\
C' := \mathsf{Enc}'_{pk'}(\pi; r) & \xrightarrow{U \,\|\, C'} & \\
 & & k \leftarrow K \\
 & & s := \alpha(k, C') \\
 & & r_j \| \tau_j \| \mathsf{sk}_j := H_k(C', \pi) \\
 & & \mathsf{label} := U \| C' \| U' \| s \\
r_i \| \tau_i \| \mathsf{sk}_i := H'(s, C', \pi, r) & \xleftarrow{U' \,\| s \,\| C} & C := \mathsf{Enc}^{\mathsf{label}}_{pk}(\pi; r_j) \\
\mathsf{label} := U \| C' \| U' \| s & & \\
\hat{C} := \mathsf{Enc}^{\mathsf{label}}_{pk}(\pi; r_i) & & \\
\text{if } C \neq \hat{C}\text{, abort} & & \\
 & \xrightarrow{\tau_i} & \text{if } \tau_i \neq \tau_j\text{, abort} \\
\text{output } \mathsf{sk}_i & & \text{output } \mathsf{sk}_j
\end{array}$$

Figure 1: An honest execution of the protocol. π denotes the shared password $\pi_{U,U'}$.

Protocol execution. A high-level depiction of the protocol is given in Figure 1. When a client instance Π_U^i wants to authenticate to the server instance $\Pi_{U'}^j$, the client first chooses a random tape r and then computes an encryption $C' := \mathsf{Enc}'_{pk'}(\pi; r)$ of the shared password π. The client then sends $U \| C'$ to the server.

Upon receiving the message $U \| C'$, the server proceeds as follows. It chooses a random hash key $k \leftarrow K$ and computes the projection key $s := \alpha(k, C')$. It then computes the hash $H_k(C', \pi)$ using the ciphertext C' it received in the first message and the password π that it shares with U. The result is parsed as a sequence of three bit-strings $r_j, \tau_j, \mathsf{sk}_j$, where τ_j and sk_j have length at least n, and r_j is sufficiently long to be used as the random tape for an encryption using Enc. The server then sets $\mathsf{label} := U\|C'\|U'\|s$ and generates an encryption $C := \mathsf{Enc}_{pk}^{\mathsf{label}}(\pi; r_j)$ of the shared password π, using the label label and the randomness r_j that it previously computed. Finally, P_j sends the message $U'\|s\|C$ back to the client.

Upon receiving the message $U'\|s\|C$, the client computes the hash using the projected key s and the randomness it used to generate the ciphertext C' in the first round; that is, P_i computes $r_i\|\tau_i\|\mathsf{sk}_i := H'(s, C', \pi, r)$. It sets $\mathsf{label} := U\|C'\|U'\|s$ and computes the ciphertext $\hat{C} := \mathsf{Enc}_{pk}^{\mathsf{label}}(\pi; r_i)$. If $C = \hat{C}$ the server has successfully authenticated to the client, and the client then accepts, sends τ_i to the server, and outputs the session key sk_i. If $C \neq \hat{C}$ then the client aborts.

When the server receives the client's final message τ_i, it checks that $\tau_i = \tau_j$ and aborts if that is not the case. Otherwise the client has successfully authenticated to the server, and the server accepts and outputs the session key sk_j.

Correctness is easily verified. If both parties are honest and there is no adversarial interference, then $H'(s, C', \pi, r) = H_k(C', \pi)$ and so it holds that $r_i = r_j$, $\tau_i = \tau_j$, and $\mathsf{sk}_i = \mathsf{sk}_j$. It follows that both parties will accept and output the same session key.

A concrete instantiation. By letting Σ' be the El Gamal encryption scheme (which is well-known to have a smooth projective hash function), and Σ be the Cramer-Shoup encryption scheme (though more efficient alternatives are possible), we recover the Jiang-Gong protocol. Without any optimization, this is about 25% faster than the KOY protocol (besides fewer exponentiations, a signature computation is avoided), and roughly 33% more communication efficient.

4 Proof of Security

This section is devoted to a proof of the following theorem:

Theorem 1 *If Σ' is a CPA-secure public-key encryption scheme with associated smooth projective hash function, and Σ is a CCA-secure public-key encryption scheme, then the protocol in Figure 1 is a secure password-based authenticated key-exchange protocol.*

Proof Fix a PPT adversary $\mathcal{A}$ attacking the protocol. We use a hybrid argument to bound the advantage of $\mathcal{A}$. Let Γ_0 represent the initial experiment, in which $\mathcal{A}$ interacts with the real protocol as defined in the previous section. In the rest of the proof, we will define a sequence of experiments $\Gamma_1, \ldots$, and denote the advantage of adversary $\mathcal{A}$ in experiment Γ_i as:

$$\mathsf{Adv}_i(n) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathcal{A} \text{ succeeds in } \Gamma_i] - 1. \tag{1}$$

Our proof will bound the difference between the adversary's advantage in successive experiments, and then bound the adversary's advantage in the final experiment; this will give the desired bound on $\mathsf{Adv}_0(n)$, the adversary's advantage when attacking the real protocol.

Experiment Γ_1. In Γ_1 we modify the way $\mathsf{Execute}$ queries are handled. Namely, in response to a query $\mathsf{Execute}(U, i, U', j)$ we now compute $C' \leftarrow \mathsf{Enc}'_{pk'}(\pi_0)$, where π_0 represents some password not in the dictionary. The remainder of the transcript is computed the same way, and the (common) session key for instances Π_U^i and $\Pi_{U'}^j$ is set to be equal to the session key sk_j computed by the server (cf. Figure 1).

Lemma 1 $|\mathsf{Adv}_0(n) - \mathsf{Adv}_1(n)| \leq \mathsf{negl}(n)$.

Proof This follows in a straightforward way from the CPA-security of encryption scheme Σ'. Construct a PPT adversary $\mathcal{B}$ attacking Σ' as follows: given public key pk', the adversary $\mathcal{B}$ simulates the entire experiment for $\mathcal{A}$ including choosing random passwords for each pair of parties. In response to $\mathsf{Execute}(U, i, U', j)$ queries, $\mathcal{B}$ queries its own "challenge" oracle using as its pair of messages the real password $\pi_{U,U'}$ and the fake password π_0; when it receives in return a ciphertext C' it includes this in the transcript that it returns to $\mathcal{A}$. Note that $\mathcal{B}$ can compute correct sessions keys $\mathsf{sk}_U^i = \mathsf{sk}_{U'}^j$ since the actions of instance $\Pi_{U'}^j$ are simulated exactly as in the real protocol (and so, in particular, $\mathcal{B}$ can compute $\mathsf{sk}_{U'}^j$ exactly as an honest player in the real protocol would). At the end of the experiment, $\mathcal{B}$ determines whether $\mathcal{A}$ succeeded or not.

The distinguishing advantage of $\mathcal{B}$ is exactly $|\mathsf{Adv}_0(n) - \mathsf{Adv}_1(n)|$. CPA-security of Σ' yields the lemma. ■

Experiment Γ_2. Here we modify the response to a query $\mathsf{Execute}(U, i, U', j)$ as follows. The first message of the transcript is $U \| C'$, where C' is an encryption of π_0 as in Γ_1. Then $k \leftarrow K$ and $s := \alpha(k, C')$ are generated as before. Now, however, we simply choose $r_j \| \tau_j \| \mathsf{sk}_j$ as a random string of the appropriate length. The ciphertext C is computed as in the real protocol, and the message $U' \| s \| C$ is added to the transcript. The final message of the protocol is $\tau_i = \tau_j$, and the session keys $\mathsf{sk}_U^i, \mathsf{sk}_{U'}^j$ are set equal to sk_j (which, recall, was chosen at random).

Lemma 2 $|\mathsf{Adv}_2(n) - \mathsf{Adv}_1(n)| \leq \mathsf{negl}(n)$.

Proof This follows from the properties of the smooth projective hash function for Σ'. To see this note that when answering $\mathsf{Execute}$ queries in Γ_1 the hash function $H_k(\cdot)$ is always applied to a pair $(C', \pi) \notin L$, and therefore the output is statistically close to uniform conditioned on the projection key s. Furthermore, in both Γ_1 and Γ_2 the values $r_i, \tau_i, \mathsf{sk}_i$ used by the client are equal to the values $r_j, \tau_j, \mathsf{sk}_j$ computed by the server. ■

Experiment Γ_3. In experiment Γ_3 we again change how $\mathsf{Execute}$ queries are handled. Namely, we compute the ciphertext C sent in the second round as $C \leftarrow \mathsf{Enc}_{pk}^{\mathsf{label}}(\pi_0)$. (We also remove the check performed by the client, and always have the client accept and output the same session key as the server.)

Lemma 3 $|\mathsf{Adv}_3(n) - \mathsf{Adv}_2(n)| \leq \mathsf{negl}(n)$.

Proof The lemma holds based on the CCA-security of Σ. (In fact, all we rely on here is security of Σ against chosen-plaintext attacks.) The key observation is that in experiment Γ_2, the ciphertext C is encrypted using *truly random* coins r_j. Thus, we can construct a PPT adversary $\mathcal{B}$ attacking Σ as follows: given public key pk, adversary $\mathcal{B}$ simulates the entire experiment for $\mathcal{A}$. In response to $\mathsf{Execute}(U, i, U', j)$ queries, $\mathcal{B}$ queries its own "challenge" oracle using as its pair of messages the real password $\pi_{U,U'}$ and the fake password π_0; when it receives in return a ciphertext C it includes this in the second message of the transcript that it returns to $\mathcal{A}$. Session keys are chosen at random.

It is immediate that the distinguishing advantage of $\mathcal{B}$ is exactly $|\mathsf{Adv}_3(n) - \mathsf{Adv}_2(n)|$. CPA-security of Σ' yields the lemma. ■

Note that in experiment Γ_3, the actions taken in response to an $\mathsf{Execute}$ query are independent of the actual passwords of any of the parties, and all session keys generated are random.

Experiment Γ_4. In this experiment we will begin to modify the Send oracle. For notational convenience, we let $\mathsf{Send}_0(U, i, U')$ denote a "prompt" message that causes the client instance Π_U^i to initiate the protocol with server U'; let $\mathsf{Send}_1(U', j, U\|C')$ denote sending the first message of the protocol to server instance $\Pi_{U'}^j$; let $\mathsf{Send}_2(U, i, U'\|s\|C)$ denote sending the second message of the protocol to client instance Π_U^i; and let $\mathsf{Send}_3(U', j, \tau)$ denote sending the final message of the protocol to server instance $\Pi_{U'}^j$.

In Γ_4 we now record the secret keys sk, sk' when the public keys in the CRS are generated. Furthermore, in response to the query $\mathsf{Send}_2(U, i, U'\|s\|C)$ we proceed as follows:

- If $\mathsf{pid}_U^i \neq U'$ then Π_U^i aborts as it would in Γ_3. From here on, we assume this is not the case.
- Let $U\|C'$ denote the initial message sent by Π_U^i (i.e., $U\|C'$ is the message that was output in response to the query $\mathsf{Send}_0(U, i, U')$). Then:
  - If $U'\|s\|C$ was output by a previous query $\mathsf{Send}_1(U', \star, U\|C')$ then we say that the message $U'\|s\|C$ is *previously-used* and the experiment continues as in Γ_3.
  - If $U'\|s\|C$ is not previously-used, then we set $\mathsf{label} := U\|C'\|U'\|s$ and compute $\pi := \mathsf{Dec}_{sk}^{\mathsf{label}}(C)$. If π is equal to the password $\pi_{U,U'}$ shared by U and U' then the adversary is immediately declared successful and the experiment ends. Otherwise, Π_U^i rejects (and outputs no session key, nor sends the final message of the protocol).

Lemma 4 *$\mathsf{Adv}_3(n) \leq \mathsf{Adv}_4(n)$.*

Proof The only situation in which Γ_4 proceeds differently from Γ_3 occurs when $U'\|s\|C$ is not previously-used but decrypts to the correct password; in this case the adversary is immediately declared successful, so its advantage can only increase. ■

Experiment Γ_5. In experiment Γ_5 we modify the way Send_0 and Send_2 queries are handled. In response to a query $\mathsf{Send}_0(U, i, U')$ we now compute $C' \leftarrow \mathsf{Enc}'_{pk'}(\pi_0)$, where (as before) π_0 denotes a dummy password that is not in the dictionary. When responding to a query $\mathsf{Send}_2(U, i, U'\|s\|C)$, we proceed as follows:

- If $\mathsf{pid}_U^i \neq U'$ we reject as always. From here on, we simply assume this does not occur.
- If $U'\|s\|C$ is previously-used (as defined in experiment Γ_4), then it was output in response to some previous query $\mathsf{Send}_1(U', j, U\|C')$; let $r_j, \tau_j, \mathsf{sk}_j$ be the internal variables used by the server instance $\Pi_{U'}^j$. Then to respond to the current Send_2 query we set $\tau_i := \tau_j$ (and send τ_i as the final message of the protocol), and set the session key for instance Π_U^i to $\mathsf{sk}_U^i := \mathsf{sk}_j$.
- If $U'\|s\|C$ is not previously-used, then we respond as in Γ_4: namely, we set $\mathsf{label} := U\|C'\|U'\|s$ and compute $\pi := \mathsf{Dec}_{sk}^{\mathsf{label}}(C)$. If π is equal to the password $\pi_{U,U'}$ shared by U and U', the adversary is declared successful and the experiment ends. Otherwise, Π_U^i rejects (and outputs no session key, nor sends the final message of the protocol).

Lemma 5 *$|\mathsf{Adv}_5(n) - \mathsf{Adv}_4(n)| \leq \mathsf{negl}(n)$.*

Proof First consider an intermediate experiment Γ_4', where the Send_2 oracle is modified as described above, but Send_0 still computes C' exactly as in Γ_4. It is not hard to see that Γ_4' is simply a syntactic rewriting of Γ_4, and so the adversary's advantage remains unchanged.

We next show that the adversary's advantage can change by only a negligible amount in moving from Γ_4' to Γ_5. This follows from the CPA-security of Σ'. Namely, we construct an adversary $\mathcal{B}$ who, given public key pk, simulates the entire experiment for $\mathcal{A}$. This includes generation of the CRS, which $\mathcal{B}$ does by generating $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ on its own and letting the CRS be (pk, pk'). In response to Send_0 queries, $\mathcal{B}$ queries its own "challenge" oracle using as its pair of messages the real password $\pi_{U,U'}$ and the dummy password π_0; when it receives in return a ciphertext C' it outputs the message $U\|C'$ to $\mathcal{A}$. Note that $\mathcal{B}$ can still respond to Send_2 queries since knowledge of the randomness used to generate C' is no longer used (in either Γ_4' or Γ_5). At the end of the experiment, $\mathcal{B}$ determines whether $\mathcal{A}$ succeeded or not.

The distinguishing advantage of $\mathcal{B}$ is exactly $|\mathsf{Adv}_5(n) - \mathsf{Adv}_4'(n)|$. CPA-security of Σ' yields the lemma. ■

Experiment Γ_6. In experiment Γ_6 we introduce a simple modification to the way Send_1 oracle calls are handled. When the adversary queries $\mathsf{Send}_1(U', j, U\|C')$, we now compute $\pi := \mathsf{Dec}'_{sk'}(C')$ (using the secret key sk' that was stored at the time the CRS was generated) and check if π is equal to the password $\pi_{U,U'}$ shared by U and U'. If so, we immediately declare the adversary successful and end the experiment. Otherwise, the experiment continues as previously. All this does is introduce a new way for the adversary to succeed, and so $\mathsf{Adv}_5(n) \leq \mathsf{Adv}_6(n)$.

It may at first appear odd that we allow the adversary to succeed in this way, since Σ' may be completely malleable. Recall, however, that in Γ_5/Γ_6 all ciphertexts C' output in response to Send_0 queries are in fact encryptions of dummy passwords; thus, the condition introduced here will not occur "trivially".

Experiment Γ_7. In experiment Γ_7 we again modify the behavior of the Send_1 oracle. In response to a query $\mathsf{Send}_1(U', j, U\|C')$ we check whether $\mathsf{Dec}'_{sk'}(C')$ is equal to $\pi_{U,U'}$ as in experiment Γ_6. If so, the adversary is declared to succeed as before. If not, however, we now choose r_j, τ_j, and sk_j uniformly at random (rather than computing these values as the output of $H_k(C', \pi)$), and then continue as before. In particular, if there is a subsequent Send_3 query using the correct value of τ_j then the server instance $\Pi^j_{U'}$ accepts and outputs the session key $\mathsf{sk}^j_{U'} := \mathsf{sk}_j$.

Lemma 6 $|\mathsf{Adv}_7(n) - \mathsf{Adv}_6(n)| \leq \mathsf{negl}(n)$.

Proof This follows from the properties of the smooth projective hash function for Σ'. Consider a query $\mathsf{Send}_1(U', j, U\|C')$ where $\mathsf{Dec}'_{sk'}(C') \neq \pi_{U,U'}$. In Γ_6, we compute $r_j\|\tau_j\|\mathsf{sk}_j := H_k(C', \pi_{U,U'})$, whereas in Γ_7 we choose r_j, τ_j, and sk_j uniformly at random. Since $(C', \pi_{U,U'}) \notin L$, however, these are statistically close given the fact that the adversary only ever sees the projected key $s := \alpha(k, C')$. ■

The key point to observe about experiment Γ_7 is that every oracle-generated second-round message contains a ciphertext C that is an encryption of the correct password *using truly random coins.*

Experiment Γ_8. For the final experiment, we again modify the response to Send_1 queries; specifically, the ciphertext C is now computed as $C \leftarrow \mathsf{Enc}^{\mathsf{label}}_{pk}(\pi_0)$, where π_0 (as always) denotes a dummy password not in the dictionary.

Lemma 7 $|\mathsf{Adv}_8(n) - \mathsf{Adv}_7(n)| \leq \mathsf{negl}(n)$.

Proof The proof relies on the CCA-security of Σ. Construct a PPT adversary $\mathcal{B}$ attacking Σ as follows: given public key pk, the adversary $\mathcal{B}$ simulates the entire experiment for $\mathcal{A}$. In response to Send_1 queries, $\mathcal{B}$ queries its own "challenge" oracle using as its pair of messages the real password $\pi_{U,U'}$ and the fake password π_0; when it receives in return a ciphertext C, it includes this ciphertext in the message that it outputs to $\mathcal{A}$. To fully simulate the experiment, $\mathcal{B}$ also has to check whether $\mathcal{A}$ succeeds in the course of making a Send_1 or Send_2 query. The former case is easy to handle, since $\mathcal{B}$ itself knows the secret key sk' corresponding to the public key pk' in the CRS; $\mathcal{B}$ can therefore decrypt the necessary ciphertexts on its own. The latter case is more subtle, as here $\mathcal{B}$ will have to use its decryption oracle in order to determine whether $\mathcal{A}$ succeeds or not. It can be verified, however, that $\mathcal{B}$ never has to request decryption of a label/ciphertext pair that it received from its own challenge oracle (this follows from the way we defined "previously-used").

The distinguishing advantage of $\mathcal{B}$ is exactly $|\mathsf{Adv}_8(n) - \mathsf{Adv}_7(n)|$. CCA-security of Σ yields the lemma. ■

Bounding the advantage in Γ_8. Consider the different ways for the adversary to succeed in Γ_8:

1. $\mathsf{Send}_1(U', j, U|C')$ is queried, where $\mathsf{Dec}_{sk'}(C') = \pi_{U,U'}$.

2. $\mathsf{Send}_2(U, i, U'\|s\|C)$ is queried, where $U'\|s\|C$ is not previously-used and $\mathsf{Dec}_{sk}^{\mathsf{label}}(C) = \pi_{U,U'}$ for label computed as discussed in experiment Γ_4.

3. The adversary successfully guesses the bit used by the Test oracle.

4. $\mathsf{Send}_3(U', j, \tau)$ is queried, where $\tau = \tau_j$ but the value τ was not output by any instance partnered with $\Pi_{U'}^j$.

Case 4 occurs with only negligible probability, since τ_j is a uniform n-bit string that is independent of the adversary's view if it was not output by any instance partnered with $\Pi_{U'}^j$.

Let $\mathsf{PwdGuess}$ be the event that case 1 or 2 occurs. Since the adversary's view is independent of all passwords until one of these cases occurs, we have $\Pr[\mathsf{PwdGuess}] \leq Q(n)/D_n$. Conditioned on $\mathsf{PwdGuess}$ not occurring, the adversary can succeed only in case 3. But then all session keys defined throughout the experiment are chosen uniformly and independently at random (except for the fact that partnered instances are given identical session keys), and so the probability of success in this case is exactly 1/2. Ignoring case 4 (which we have already argued occurs with only negligible probability), then, we have

$$\begin{aligned}
\Pr[\mathsf{Success}] &= \Pr[\mathsf{Success} \wedge \mathsf{PwdGuess}] + \Pr[\mathsf{Success} \wedge \overline{\mathsf{PwdGuess}}] \\
&\leq \Pr[\mathsf{PwdGuess}] + \Pr[\mathsf{Success} \mid \overline{\mathsf{PwdGuess}}] \cdot (1 - \Pr[\mathsf{PwdGuess}]) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \Pr[\mathsf{PwdGuess}] \\
&\leq \frac{1}{2} + \frac{Q(n)}{2 \cdot D_n},
\end{aligned}$$

and so $\mathsf{Adv}_8(n) \leq Q(n)/D_n$. Lemmas 1–7 imply that $\mathsf{Adv}_0(n) \leq Q(n)/D_n + \mathsf{negl}(n)$ as desired. ■

References

[1] M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. In *Advances in Cryptology — Crypto 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, 2009.

[2] B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin. Secure computation without authentication. In *Advances in Cryptology — Crypto 2005*, volume 3621 of *LNCS*, pages 361–377. Springer, 2005.

[3] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology — Eurocrypt 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, 2000.

[4] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology — Crypto '93*, volume 773 of *LNCS*, pages 232–249. Springer, 1994.

[5] M. Bellare and P. Rogaway. Provably secure session key distribution: The three party case. In *27th Annual ACM Symposium on Theory of Computing (STOC)*, pages 57–66. ACM Press, 1995.

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

[7] V. Boyko, P. D. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Advances in Cryptology — Eurocrypt 2000*, volume 1807 of *LNCS*, pages 156–171. Springer, 2000.

[8] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, 2005.

[9] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology — Eurocrypt 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, 2002.

[10] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[11] R. Gennaro. Faster and shorter password-authenticated key exchange. In *5th Theory of Cryptography Conference — TCC 2008*, volume 4948 of *LNCS*, pages 589–606. Springer, 2008.

[12] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. *ACM Trans. Information and System Security*, 9(2):181–234, 2006.

[13] O. Goldreich and Y. Lindell. Session-key generation using human passwords only. *Journal of Cryptology*, 19(3):241–340, 2006.

[14] L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE J. Selected Areas in Communications*, 11(5):648–656, 1993.

[15] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. *ACM Trans. Information and System Security*, 2(3):230–268, 1999.

[16] S. Jiang and G. Gong. Password based key exchange with mutual authentication. In *11th Annual International Workshop on Selected Areas in Cryptography (SAC)*, volume 3357 of *LNCS*, pages 267–279. Springer, 2004.

[17] J. Katz, P. D. MacKenzie, G. Taban, and V. D. Gligor. Two-server password-only authenticated key exchange. In *3rd International Conference on Applied Cryptography and Network Security (ACNS)*, volume 3531 of *LNCS*, pages 1–16. Springer, 2005.

[18] J. Katz, R. Ostrovsky, and M. Yung. Efficient and secure authenticated key exchange using weak passwords. *Journal of the ACM*, 57(1):78–116, 2009.

[19] J. Katz and V. Vaikuntanathan. Password-based authenticated key exchange based on lattices. In *Advances in Cryptology — Asiacrypt 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, 2009.

[20] K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *Advances in Cryptology — Crypto 2004*, volume 3152 of *LNCS*, pages 426–442. Springer, 2004.

[21] P. D. MacKenzie, S. Patel, and R. Swaminathan. Password-authenticated key exchange based on RSA. In *Advances in Cryptology — Asiacrypt 2000*, volume 1976 of *LNCS*, pages 599–613. Springer, 2000.

[22] M.-H. Nguyen and S. Vadhan. Simpler session-key generation from short random passwords. *Journal of Cryptology*, 21(1):52–96, 2008.

[23] V. Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. Available at `http://eprint.iacr.org/`.