One-Round Password-Based Authenticated Key Exchange

Jonathan Katz* Vinod Vaikuntanathan†

Abstract

We show a general framework for constructing password-based authenticated key exchange protocols with *optimal* round complexity — one message per party, sent simultaneously — in the standard model, assuming the existence of a common reference string. When our framework is instantiated using bilinear-map cryptosystems, the resulting protocol is also (reasonably) efficient. Somewhat surprisingly, our framework can be adapted to give protocols (still in the standard model) that are *universally composable* while still using only one (simultaneous) round.

*Dept. of Computer Science, University of Maryland. Work done while visiting IBM. Email: jkatz@cs.umd.edu.
Research supported by NSF grant #0627306 and NSF CAREER award #0447075.
†IBM Research. vinodv@alum.mit.edu.

1 Password-Based Authenticated Key Exchange

Protocols for *authenticated key exchange* enable two parties to generate a shared, cryptographically strong key while communicating over an insecure network under the complete control of an adversary. Such protocols are among the most widely used and fundamental cryptographic primitives; indeed, agreement on a shared key is necessary before "higher-level" tasks such as encryption and message authentication become possible.

Parties must share *some* information in order for authenticated key exchange to be possible. It is well known that shared cryptographic keys — either in the form of public keys or a long, uniformly random symmetric key — suffice, and several protocols in this model (building on the classic Diffie-Hellman protocol [22], which protects only against an eavesdropping adversary and provides no authentication at all) are known [8, 5, 6, 3, 43, 18, 19, 36, 37].

Password-based protocols allow users to "bootstrap" even a *very weak* (e.g., short) shared secret into a (much longer) cryptographic key. The canonical application here is authentication using *passwords*, though protocols developed in this context can be useful even when the shared secret has high min-entropy (but is not uniform) [13]. The security guaranteed by password-based protocols (roughly speaking) is that if the password is chosen uniformly[1] from a dictionary of size D then an adversary who initiates Q "on-line" attacks — i.e., who actively interferes in Q sessions — has "advantage" at most Q/D. (This is inherent, as an adversary can always carry out Q impersonation attempts and succeed with this probability.) In particular, "off-line" dictionary attacks where an adversary enumerates all passwords from the (presumably small) dictionary of potential passwords, and tries to match observed protocol executions to each one, are of no use.

Early work [27, 30] considered a "hybrid" setting where users share public keys in addition to a password. In the setting where *only* a password is shared, Bellovin and Merritt [7] proposed the first protocols for password-based authenticated key exchange (PAK) with heuristic arguments for their security. Several years later, provably secure PAK protocols were constructed [4, 14, 38] in the random oracle/ideal cipher models, and many improvements and generalizations of these protocols are known. In contrast, only a handful of PAK protocols are known in the so-called "standard model" (i.e., without random oracles):

General assumptions: Goldreich and Lindell [26] gave the first PAK protocol in the standard model. Subsequent work of Barak et al. [2] shows a general feasibility result for computation over unauthenticated networks which implies a solution for PAK as a special case. These approaches gives the only PAK protocols for the plain model where there is no setup. (Nguyen and Vadhan [40] show efficiency improvements to the Goldreich-Lindell protocol, but achieve a weaker notion of security.) Unfortunately, all these approaches are completely impractical in terms of communication, computation, and round complexity. Moreover, they do not tolerate concurrent executions by the same party (unless additional setup is assumed).

Efficient protocols: Katz, Ostrovsky, and Yung [34] demonstrated the first *efficient* PAK protocol with a proof of security based on standard assumptions; extensions and improvements of their protocol were given in [25, 17, 33, 24, 35]. Different constructions of efficient PAK protocols in the CRS model are given in [32, 28]. In contrast to the work of Goldreich and Lindell, these approaches are secure even under concurrent executions by the same party. On the

[1] Although the usual presentation of PAK assumes a uniform password, all known protocols work with passwords chosen from an arbitrary (efficiently sampleable) distribution.

other hand, they require a *common reference string* (CRS). (Use of a CRS in cryptographic protocols has a long history that can be traced back to [10].) While less appealing than the "plain model", using a CRS is not a serious drawback in the context of PAK where the CRS can be hard-coded into an implementation of the protocol. We note also that reliance on a CRS (or some other setup) is inherent for achieving *universally composable* PAK [17].

Round/message complexity of existing protocols. We distinguish between *rounds* and *messages*. Differing somewhat from the usual convention in the two-party setting (but matching the usual convention in the multi-party setting), we let a round consist of one message sent by each party simultaneously; note that in a one-round protocol each honest party's message (if any) cannot depend on the other party's message. We stress, however, that even for one-round protocols the adversary is always assumed to be *rushing*; i.e., the adversary may wait to receive the other party's first-round message before sending its own.

Determining the optimal round complexity of key-exchange protocols is of both theoretical and practical interest, and has been studied in various settings. The original Diffie-Hellman protocol [22], which provides security against a passive eavesdropper, can be run in one round; one-round authenticated key exchange based on shared public/symmetric keys is also possible [31, 41]. One-round protocols for PAK are also known (e.g., [4]), albeit in the random oracle model. All known PAK protocols based on standard assumptions, though, require three or more rounds. We remark that the protocols in [32, 28] achieve *explicit* authentication in three rounds (whereas the protocols of [34, 25, 24, 35] achieve only *implicit* authentication in three rounds, and require an additional round for explicit authentication), but the round complexity of these protocols cannot be reduced even if only implicit authentication is desired.

1.1 Our Results

We show a new framework for constructing *one-round* PAK protocols in the standard model (assuming a CRS), where each party may send their message *simultaneously*. (Once again, we stress that our security model allows for a "rushing" adversary who waits to see the message sent by a party before sending its response.) Our protocols achieve implicit authentication but can be extended to give explicit authentication using one additional round; it is not hard to see that explicit authentication is impossible in one round without stronger setup assumptions (e.g., a global clock).

Our framework relies on non-interactive zero-knowledge proofs (NIZK) and so, in general, may be computationally inefficient. When instantiating our framework using bilinear maps, however, we obtain a reasonably efficient solution (e.g., communicating a constant number of group elements).

Moreover, and somewhat surprisingly, we can extend our framework to give a universally composable PAK protocol [16] *without increasing the round complexity at all* (and still without relying on random oracles). In contrast, the work of [17] shows a method (used also by [28]) for obtaining universal composability that requires additional messages/rounds. Abdalla et al. [1] show a universally composable PAK protocol, proven secure in the random oracle model, that requires three rounds. To the best of our knowledge, no prior universally composable protocol (whether in the random oracle model or not) can be run in only one round.

1.2 Our Techniques

At a basic level, we rely on smooth projective hash functions [20], as used in [25] (and implicitly in [34]); see Section 2.2 for a definition. The basic structure of previous protocols [34, 25], omitting many important details, is as follows:

First round: The client sends an encryption C of the password pw.

Second round: The server sends an encryption C' of pw, and a projected key $s' = \alpha(k', C, pw)$.

Third round: The client sends a projected key $s = \alpha(k, C', pw)$.

The client computes the session key as $H_k(C', pw) \cdot H_{s'}(C, pw, r)$, and the server computes the session key as $H_s(C', pw, r') \cdot H_{k'}(C, pw)$. (Here, r, r' is the randomness used to compute C, C', respectively.) Properties of the smooth projective hash function ensure that these are equal.

Two difficulties must be overcome in order to collapse a protocol of the above form to one round:

- In the smooth projective hash functions used in prior work, the "projection function" α was *adaptive*, and depended on both the hash key k and the element being hashed (i.e., (C, pw) in the above example). This leads to protocols requiring three rounds just to ensure correctness.

  Here we show a construction of CCA-secure encryption schemes with associated smooth projective hash functions whose projection function is *non-adaptive*, and depends only on the hash key k. This allows us to obtain the *functionality* of PAK in a single round, by having the client send $(\alpha(k), C)$ and the server send $(\alpha(k'), C')$ simultaneously.

- The above addresses correctness, but says nothing about security of the protocol. The technical difficulty here is that an honestly generated client message $\mathsf{msg} = (s, C)$ might be forwarded by an adversary to *multiple* server instances (and vice versa), and we need to guarantee that the session keys computed in all these instances look random and independent to the adversary. (This issue does not arise in prior work because, roughly speaking, messages are *bound* to a single session by virtue of a signature key sent in the first round [34, 25] or a MAC derived from the shared session key [24]. Neither approach is viable if we want the entire protocol to take place in a single round.)

  Due to the above difficulty, the proof of security is the most technically challenging part of our work. Our proof relies on a technical lemma related to re-use of hash keys and elements that may be of independent interest.

Additional ideas are needed to obtain a *universally composable* protocol without increasing the number of rounds. We refer the reader to Section 4.1 for an overview of the techniques used there.

1.3 Outline of the Paper

In Section 2 we present a standard definition of security for PAK due to Bellare et al. [4]. We also review there the notion of smooth projective hashing, and prove an important technical lemma regarding its usage. In Section 3.1 we describe our basic framework for constructing one-round PAK protocols, and prove security of this approach according to the definition of [4]. We discuss in Section 3.2 two instantiations of our framework: one based on the decisional Diffie-Hellman assumption, and a second, more efficient instantiation based on bilinear maps. In Section 4 we describe an extension of our framework that yields one-round, universally composable password-based authenticated key-exchange protocols.

2 Definitions and Background

Throughout, we denote the security parameter by n.

2.1 Password-Based Authenticated Key Exchange

Here we present a definition of security for PAK due to Bellare, Pointcheval, and Rogaway [4], based on prior work of [5, 6]. The text here is taken almost verbatim from [34].

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 $pw_{U,U'}$. We make the simplifying assumption that each $pw_{U,U'}$ is chosen independently and uniformly at random from the set $[D] \stackrel{\text{def}}{=} \{1, \ldots, D\}$ for some integer D. (Our proof of security extends to more general cases, and we explicitly consider arbitrary password distributions when we move to the setting of universal composability.)

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* which we describe now:

- $\mathsf{Send}(U, i, \mathsf{msg})$ — This sends message msg 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.

  The adversary can "prompt" instance Π_U^i to initiate the protocol with partner U' by querying $\mathsf{Send}(U, i, U')$. In response to this query, instance Π_U^i outputs the first message of the protocol.
- $\mathsf{Execute}(U, i, U', j)$ — 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 succeeds if it can guess the bit b used by the Test oracle. To formally define the adversary's success, 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 allow the adversary to succeed only if its Test query is made to a fresh instance; this is necessary for any reasonable definition of security.

An adversary $\mathcal{A}$ *succeeds* if it 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). We denote this event by Succ. The *advantage* of adversary $\mathcal{A}$ in attacking protocol Π is given by $\mathsf{Adv}_{\mathcal{A},\Pi}(k) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{Succ}] - 1$, where the probability is taken over the random coins used by the adversary and the random coins used 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 both the following are true at the time of the Test query: (1) at some point, the adversary queried $\mathsf{Send}(U,i,*)$; and (2) at some point, the adversary queried $\mathsf{Reveal}(U,i)$ or $\mathsf{Test}(U,i)$. 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 and for all* PPT *adversaries $\mathcal{A}$ making at most $Q(n)$ on-line attacks, it holds that* $\mathsf{Adv}_{\mathcal{A},\Pi}(n) \leq Q(n)/D + \mathsf{negl}(n)$.

2.2 Smooth Projective Hash Functions

We provide a self-contained definitional treatment of smooth projective hash functions. These were introduced by Cramer and Shoup [20], and our discussion here is based on that of Gennaro and Lindell [25]. Rather than aiming for utmost generality, we tailor the definitions to our application.

Hard subset membership problems. Fix some integer D. Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA-secure labeled encryption scheme (cf. Appendix A.1). For a given public key pk, we let C_{pk} denote the set of pairs of valid labels and valid ciphertexts with respect to pk, and require that this set be efficiently recognizable. For a given public key pk, define sets X and $\{L_{pw}\}_{pw \in [D]}$ as follows:

1. $X \stackrel{\text{def}}{=} \{(\mathsf{label}, C, pw)\}$, where $(\mathsf{label}, C) \in C_{pk}$ and $pw \in \{1, \ldots, D\}$.

2. $L_{pw} \stackrel{\text{def}}{=} \{(\mathsf{label}, \mathsf{Enc}_{pk}(\mathsf{label}, pw), pw)\}$, where $\mathsf{label} \in \{0,1\}^*$. Let $L = \bigcup_{pw=1}^{D} L_i$.

Note that $L \subset X$. It follows from CCA security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ that the following is negligible for any polynomial-time adversary $\mathcal{A}$:

$$\Big|\Pr\Big[(pk, sk) \leftarrow \mathsf{Gen}(1^n); (\mathsf{label}, pw) \leftarrow \mathcal{A}(pk); C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw) : \mathcal{A}^{\mathsf{Dec}_{sk}(\cdot,\cdot)}(C) = 1\Big]$$
$$- \Pr\Big[(pk, sk) \leftarrow \mathsf{Gen}(1^n); (\mathsf{label}, pw) \leftarrow \mathcal{A}(pk); C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, 0) : \mathcal{A}^{\mathsf{Dec}_{sk}(\cdot,\cdot)}(C) = 1\Big]\Big|, \quad (1)$$

where $\mathcal{A}$ is disallowed from querying (label, C) to its decryption oracle.

Smooth projective hash functions. Fix pk and sets $X, \{L_i\}$ as above. A *smooth projective hash function* $\mathcal{H} = \{H_k\}_{k \in K}$ is a keyed function mapping elements in X to elements in some group G, along with a *projection function* $\alpha : K \to S$. Informally, if $x \in L$ then the value of $H_k(x)$ is uniquely determined by $s = \alpha(k)$ and x, whereas if $x \in X \setminus L$ then the value of $H_k(x)$ is statistically close to uniform given $\alpha(k)$ and x (assuming k was chosen uniformly in K). A smooth projective hash function is formally defined by a sampling algorithm that, given pk, outputs $(K, G, \mathcal{H} = \{H_k : X \to G\}_{k \in K}, S, \alpha : K \to S)$ 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 $\alpha(k)$ for all $k \in K$.

2. For all $(\mathsf{label}, C, pw) \in L$, the value of $H_k(\mathsf{label}, C, pw)$ is uniquely determined by $\alpha(k)$. Moreover, there is an efficient algorithm that takes as input $s = \alpha(k)$ and $(\mathsf{label}, C, pw, r)$ for which $C = \mathsf{Enc}_{pk}(\mathsf{label}, pw; r)$, and outputs $H_k(\mathsf{label}, C, pw)$. (In other words, when $(\mathsf{label}, C, pw) \in L$ then $H_k(\mathsf{label}, C, pw)$ can be computed in two ways: either using k itself, or using $\alpha(k)$ and the randomness used to generate C.)

3. For any (even unbounded) function $f : S \to X \setminus L$, the following distributions have statistical difference negligible in n:

$$\big\{k \leftarrow K; s = \alpha(k) : \big(s, H_k(f(s))\big)\big\} \quad \text{and} \quad \{k \leftarrow K; s = \alpha(k); g \leftarrow G : (s, g)\}. \quad (2)$$

We stress that in the above we modify the definition from [25] in two ways: first, α is *non-adaptive*, and depends on k only (rather than both k and x); second, we require Equation (2) to hold even for *adaptive* choice of $f(s) \notin L$.

2.2.1 A Technical Lemma

We now prove a technical lemma regarding smooth projective hash functions. Somewhat informally, Gennaro and Lindell [25] showed that, for randomly generated pk and any label, pw, the distribution

$$\big\{k \leftarrow K; s = \alpha(k); C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw) : \big(s, C, H_k(\mathsf{label}, C, pw)\big)\big\}$$

is computationally indistinguishable from the distribution

$$\{k \leftarrow K; s = \alpha(k); C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw); g \leftarrow G : (s, C, g)\}.$$

(Note this holds even though $H_k(\mathsf{label}, C, pw)$ is uniquely determined by s and C.) Here we show that this continues to hold even if hash keys and ciphertexts are *re-used* multiple times. That is, at a high level (ignoring labels and technical details), we show that the distribution

$$\Big\{k_1, \ldots, k_\ell \leftarrow K; \forall i : s_i = \alpha(k_i); C_1, \ldots, C_\ell \leftarrow \mathsf{Enc}_{pk}(pw) : \Big(\{s_i\}, \{C_i\}, \{H_{k_i}(C_j, pw)\}_{i,j=1}^{\ell}\Big)\Big\}$$

is computationally indistinguishable from the distribution

$$\left\{k_1, \ldots, k_\ell \leftarrow K; \forall i : s_i = \alpha(k_i); C_1, \ldots, C_\ell \leftarrow \mathsf{Enc}_{pk}(pw); g_{i,j} \leftarrow G : \left(\{s_i\}, \{C_i\}, \{g_{i,j}\}_{i,j=1}^{\ell}\right)\right\}.$$

Formally, fix a function $\ell = \ell(n)$, let $\mathcal{A}$ be an adversary, and let $b \in \{0,1\}$. Consider the following experiment Expt_b:

1. Compute $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and let $(K, G, \mathcal{H} = \{H_k : X \to G\}_{k \in K}, S, \alpha : K \to S)$ be an associated smooth projective hash function for pk. Given pk to $\mathcal{A}$.
2. Sample $k_1, \ldots, k_\ell \leftarrow K$, and let $s_i = \alpha(k_i)$ for all i. Give $s_1, \ldots, s_\ell$ to $\mathcal{A}$.
3. $\mathcal{A}$ may adaptively query a (modified) encryption oracle that takes as input (label, pw) and outputs a ciphertext $C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw)$ along with
   (a) If $b = 0$, the values $H_{k_i}(\mathsf{label}, C, pw)$ for $i = 1$ to ℓ.
   (b) If $b = 1$, random values $g_1, \ldots, g_\ell \leftarrow G$.
4. $\mathcal{A}$ can also query a decryption oracle $\mathsf{Dec}_{sk}(\cdot, \cdot)$ at any point, except that it may not query any pair (label, C) where C was obtained from the encryption oracle on query (label, pw).
5. At the end of the experiment, $\mathcal{A}$ outputs a bit b'. We say $\mathcal{A}$ *succeeds* if $b' = b$.

A proof of the following appears in Appendix C.1:

Lemma 1 *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a CCA-secure labeled encryption scheme. For any polynomial* ℓ *and polynomial-time* $\mathcal{A}$*, we have* $\Pr[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(n)$.

3 A One-Round PAK Protocol

3.1 The Framework and Proof of Security

Our protocol uses a CCA-secure labeled public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and a smooth projective hash function as described in Section 2.2.

Public parameters. The public parameters consist of a public key pk generated by $\mathsf{Gen}(1^n)$. No one need know or store the associated secret key. (For the specific instantiations given in Section 3.2, a public key can be derived from a common random string.) Let $(K, G, \mathcal{H} = \{H_k : X \to G\}_{k \in K}, S, \alpha : K \to S)$ be a smooth projective hash function for pk.

Protocol execution. Consider an execution of the protocol between users U and $U' \neq U$ holding a shared password pw. Our protocol is symmetric, and so we describe the execution from the point of view of U; see also Figure 1.

First, U chooses random hash key $k \leftarrow K$ and computes $s := \alpha(k)$. It then sets $\mathsf{label} := (U, U', s)$ and computes the ciphertext $C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw)$. It sends the message (s, C).

Upon receiving the message (s', C'), user U does the following. If C' is not a valid ciphertext or $s' \notin S$, then U simply rejects. Otherwise, U sets $\mathsf{label}' := (U', U, s')$ and computes

$$\mathsf{sk}_U := H_k(\mathsf{label}', C', pw) \cdot H_{k'}(\mathsf{label}, C, pw).$$

Note that U can compute $H_k(\mathsf{label}', C', pw)$ since it knows k, and can compute $H_{k'}(\mathsf{label}, C, pw)$ using $s' = \alpha(k')$ and the randomness it used to generate C. Correctness follows immediately from the definition of smooth projective hashing.

Public parameters: pk

User U		User U'
$k \leftarrow K;\ s := \alpha(k)$		$k' \leftarrow K;\ s' := \alpha(k')$
$\mathsf{label} := (U, U', s)$		$\mathsf{label} := (U', U, s')$
$C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw)$	$\xrightarrow{\ s, C\ }$	$C' \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}', pw)$
	$\xleftarrow{\ s', C'\ }$	
$\mathsf{label}' := (U', U, s')$		$\mathsf{label} := (U, U', s)$
$sk_U := H_k(\mathsf{label}', C', pw) \cdot H_{k'}(\mathsf{label}, C, pw)$		$sk_{U'} := H_k(\mathsf{label}', C', pw) \cdot H_{k'}(\mathsf{label}, C, pw)$

Figure 1: A one-round protocol for password-based authenticated key exchange.

Theorem 1 *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a CCA-secure labeled encryption scheme and* $(K, G, \mathcal{H} = \{H_k : X \to G\}_{k \in K}, S, \alpha : K \to S)$ *is a smooth projective hash function, then the protocol in Figure 1 is a secure protocol for password-based authenticated key exchange.*

Proof Let Π denote the protocol in Figure 1, and fix a polynomial-time adversary $\mathcal{A}$ attacking Π. We construct a sequence of experiments $\mathsf{Expt}_0, \ldots, \mathsf{Expt}_5$, with the original experiment corresponding to Expt_0. Let $\mathsf{Adv}_{\mathcal{A},i}(n)$ denote the advantage of $\mathcal{A}$ in experiment Expt_i. To prove the desired bound on $\mathsf{Adv}_{\mathcal{A},\Pi}(n) = \mathsf{Adv}_{\mathcal{A},0}(n)$, we bound the effect of each change in the experiment on the advantage of $\mathcal{A}$, and then show that $\mathsf{Adv}_{\mathcal{A},5}(n) \leq Q(n)/D$ (where, recall, $Q(n)$ denotes the number of on-line attacks made by $\mathcal{A}$, and D denotes the dictionary size).

Experiment Expt_1: Here we change the way $\mathsf{Execute}$ queries are answered. Specifically, the ciphertexts C, C' sent by the two parties U, U' are computed as encryptions of 0 instead of being computed as encryptions of the correct password $pw_{U,U'}$. (Recall that the space of legal passwords is $\{1, \ldots, D\}$, and so 0 is never a valid password.) The (common) session key is computed as

$$\mathsf{sk}_U := \mathsf{sk}_{U'} := H_k(\mathsf{label}', C', pw) \cdot H_{k'}(\mathsf{label}, C, pw),$$

where both values are computed using the (known) keys k, k'. A proof of the following is immediate:

Claim 1 *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is semantically secure,* $|\mathsf{Adv}_{\mathcal{A},0}(n) - \mathsf{Adv}_{\mathcal{A},1}(n)|$ *is negligible.*

Experiment Expt_2: Here, we again change the way $\mathsf{Execute}$ queries are answered. Now, the (common) session key $\mathsf{sk}_U = \mathsf{sk}_{U'}$ is chosen uniformly from G.

Claim 2 $|\mathsf{Adv}_{\mathcal{A},1}(n) - \mathsf{Adv}_{\mathcal{A},2}(n)|$ *is negligible.*

Proof The claim follows from the properties of the smooth projective hash function. Consider a single call to the $\mathsf{Execute}$ oracle (in either Expt_1 or Expt_2), where the transcript given to the adversary is (s, C, s', C') with $C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, 0)$ and $C' \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}', 0)$. In Expt_1 the session keys are computed as

$$\mathsf{sk}_U := \mathsf{sk}_{U'} := H_k(\mathsf{label}', C', pw) \cdot H_{k'}(\mathsf{label}, C, pw),$$

where $pw = pw_{U,U'}$ is the password shared by U and U'. Since $(\mathsf{label}', C', pw)$ is *not* in L, it follows (cf. Equation (2)) that $(s, H_k(\mathsf{label}', C', pw))$ is statistically close to (s, g), where g is a uniform element in G. Similarly, $(s', H_{k'}(\mathsf{label}, C, pw))$ is statistically close to (s', g'). This means that $\mathsf{sk}_U = \mathsf{sk}_{U'}$ is statistically close to a uniform element in G, even conditioned on the given transcript. Since this is how $\mathsf{sk}_U, \mathsf{sk}_{U'}$ are chosen in Expt_2, the claim follows. ■

Before continuing, we distinguish between two possible types of Send oracle queries. We let $\mathsf{Send}_0(U, i, U')$ denote a "prompt" query that causes instance Π_U^i of user U to initiate the protocol with user U'. In response to a Send_0 query, the adversary is given the message sent by U to U'. This query also has the effect of setting $\mathsf{pid}_U^i = U'$.

The second type of Send query, $\mathsf{Send}_1(U, i, \mathsf{msg})$, denotes the event where $\mathcal{A}$ sends the message msg to instance Π_U^i. In response to this query, a session key sk_U^i is computed. (Nothing is output in response to this query, but the value of the computed session key can be learned via a subsequent Reveal or Test query.) For a query $\mathsf{Send}_1(U, i, \mathsf{msg})$ with $\mathsf{pid}_U^i = U'$, we say msg is *previously used* if it was output by a previous oracle query $\mathsf{Send}_0(U', \star, U)$. In any other case, we say msg is *adversarially generated.*

Experiment Expt_3: We first modify the experiment so that, when the public parameters pk are generated, the simulator stores the associated secret key sk. (This is just a syntactic change.) We then modify the way queries to the Send_1 oracle are handled. Specifically, in response to the query $\mathsf{Send}_1(U, i, \mathsf{msg})$ where $\mathsf{msg} = (s', C')$, we distinguish the following three cases (in all the following, let $\mathsf{pid}_U^i = U'$ and set $\mathsf{label}' := (U', U, s')$):

1. If msg is adversarially generated, then compute $pw' := \mathsf{Dec}_{sk}(\mathsf{label}', C')$. Then:
   (a) If $pw' = pw_{U,U'}$, the simulator declares that $\mathcal{A}$ succeeds and terminates the experiment.
   (b) If $pw' \neq pw_{U,U'}$, the simulator chooses sk_U^i uniformly from G.
2. If msg is previously used, then in particular the simulator knows a value k' such that $s' = \alpha(k')$. The simulator computes $\mathsf{sk}_U^i := H_k(\mathsf{label}', C', pw) \cdot H_{k'}(\mathsf{label}, C, pw)$, but using k' to compute $H_{k'}(\mathsf{label}, C, pw)$ (rather than using the randomness used to generate C, as done in Expt_2).

Invalid messages are treated as before, and no session key is computed.

Claim 3 $\mathsf{Adv}_{\mathcal{A},2}(n) \leq \mathsf{Adv}_{\mathcal{A},3}(n) + \mathsf{negl}(n)$.

Proof Consider the three possible cases described above. The change in Case 2 does not affect the computed value sk_U^i. The change in Case 1(b) introduces a negligible statistical difference between Expt_3 and Expt_2 (the analysis is as in Claim 2, except that we now specifically use the fact that Equation (2) holds even under adaptive choice of $(\mathsf{label}', C', pw) \notin L$). Finally, the change in Case 1(a) can only increase the advantage of $\mathcal{A}$. ■

Experiment Expt_4: Once again we change how Send_1 queries are handled. In response to query $\mathsf{Send}_1(U, i, \mathsf{msg})$ where $\mathsf{msg} = (s', C')$ is previously used, let $\mathsf{pid}_U^i = U'$ and proceed as follows:

- If there exists an instance $\Pi_{U'}^j$ partnered with Π_U^i (i.e., such that $\mathsf{sid}_{U'}^j$, the transcript of the protocol for instance $\Pi_{U'}^j$, is equal to sid_U^i), then set $\mathsf{sk}_U^i := \mathsf{sk}_{U'}^j$.
- Otherwise, choose sk_U^i uniformly from G.

Claim 4 *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is CCA-secure,* $|\mathsf{Adv}_{\mathcal{A},3}(n) - \mathsf{Adv}_{\mathcal{A},4}(n)|$ *is negligible.*

Proof The proof relies on Lemma 1 (cf. Section 2.2.1). Let ℓ be a polynomial upper bound on the number of Send queries issued by $\mathcal{A}$, and consider the following adversary $\mathcal{S}$ interacting in the experiment defined in Lemma 1:

1. $\mathcal{S}$ is given pk, and $s_1, \ldots, s_\ell$.
2. $\mathcal{S}$ chooses random passwords $pw_{U,U'}$ for all pairs of parties U, U', and runs $\mathcal{A}$ on input pk.
3. $\mathcal{S}$ responds to $\mathsf{Execute}$ queries as in Expt_2 by generating a transcript where C, C' are encryptions of 0, and where the (matching) session keys are chosen uniformly at random.
4. For all i, adversary $\mathcal{S}$ responds to the ith Send_0 query $\mathsf{Send}_0(U, \star, U')$ as follows: Set $\mathsf{label} := (U, U', s_i)$. Submit $(\mathsf{label}, pw_{U,U'})$ to the encryption oracle, and receive in return a ciphertext C_i along with values $h_{1,i}, \ldots, h_{\ell,i}$. Give to $\mathcal{A}$ the message (s_i, C_i).
5. $\mathcal{S}$ responds to a query $\mathsf{Send}_1(U, j, \mathsf{msg})$, where $\mathsf{msg} = (s', C')$, as follows: If there exists an instance $\Pi^k_{U'}$ partnered with Π^j_U, then set $\mathsf{sk}^j_U := \mathsf{sk}^k_{U'}$. Otherwise, let $\mathsf{pid}^j_U = U'$, set $\mathsf{label}' := (U', U, s')$, and say the query $\mathsf{Send}_0(U, j, U')$ (i.e., the Send query that initiated instance Π^j_U) was the ith Send_0 query made by $\mathcal{A}$, and resulted in the response (s_i, C_i). We now distinguish several cases based on $\mathsf{msg} = (s', C')$:
   (a) If msg is invalid (i.e., C' is an invalid ciphertext, or $s' \notin S$), no session key is computed.
   (b) If msg is previously used, then (by definition) it was output by some previous query $\mathsf{Send}_0(U', \star, U)$. Say this was the rth Send_0 query made by $\mathcal{A}$, and so $\mathsf{msg} = (s_r, C_r)$. Then $\mathcal{S}$ computes $\mathsf{sk}^j_U := h_{i,r} \cdot h_{r,i}$.
   (c) If msg is adversarially generated, then $\mathcal{S}$ submits (label', C') to its decryption oracle and receives in return a value pw. If $pw \neq pw_{U,U'}$ then sk^j_U is chosen uniformly from G. If $pw = pw_{U,U'}$ then $\mathcal{S}$ declares that $\mathcal{A}$ succeeds and terminates the experiment.
6. At the end of the experiment, $\mathcal{S}$ outputs 1 if and only if $\mathcal{A}$ succeeds.

Referring to the game defined in Section 2.2.1, note that if $b = 0$ then the view of $\mathcal{A}$ in the above execution with $\mathcal{S}$ is identical to the view of $\mathcal{A}$ in Expt_3. This is true since when $b = 0$ it holds in step 5(b), above, that $h_{i,r} = H_{k_i}(\mathsf{label}', C_r, pw_{U,U'})$ and $h_{r,i} = H_{k_r}(\mathsf{label}, C_i, pw_{U,U'})$, where $s_i = \alpha(k_i)$, $s_r = \alpha(k_r)$, and C_i, C_r are encryptions of $pw_{U,U'}$.

On the other hand, when $b = 1$ we claim that the view of $\mathcal{A}$ in the above execution with $\mathcal{S}$ is identical to the view of $\mathcal{A}$ in Expt_4. To see this, recall that when $b = 1$ all the values $\{h_{i,j}\}$ received by $\mathcal{S}$ are chosen uniformly and independently from G. We need to show that this yields a uniform and independent distribution on all the session keys computed in step 5(b). Consider a particular session key sk^j_U computed as in step 5(b). The only other possible time the value $h_{i,r}$ can be used in the experiment is if $\mathcal{A}$ queries $\mathsf{Send}_1(U', k, (s_i, C_i))$ to the instance $\Pi^k_{U'}$ which sent (s_r, C_r). But then $\Pi^k_{U'}$ and Π^j_U are partnered, and so the session key will be set equal to sk^j_U (exactly as in Expt_4). Since $h_{i,r}$ is random and used only once to compute a session key in step 5(b), we conclude that (when $b = 1$) any session keys computed in that step are uniform and independent in G.

The claim now follows from Lemma 1. ■

Experiment Expt_5: Here, we change how Send_0 queries are handled. Now, in response to a query $\mathsf{Send}_0(U, i, U')$, we compute s as usual but let C be an encryption of 0. The following claim is immediate from CCA-security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

Claim 5 *If* (Gen, Enc, Dec) *is CCA-secure,* $|\mathsf{Adv}_{\mathcal{A},4}(n) - \mathsf{Adv}_{\mathcal{A},5}(n)|$ *is negligible.*

In Expt_5, the view of $\mathcal{A}$ is independent of any of the user's passwords until it sends an adversarially generated message that corresponds to an encryption of the correct password (at which point $\mathcal{A}$ succeeds, as described in Expt_3). It therefore holds that $\mathsf{Adv}_{\mathcal{A},5}(n) \leq Q(n)/D$. Claims 1–5 thus imply that $\mathsf{Adv}_{\mathcal{A},0}(n) \leq Q(n)/D + \mathsf{negl}(n)$, completing the proof of the theorem. ■

3.2 Instantiating the Building Blocks

We now discuss two possible instantiations of the building blocks required by the protocol of the previous section. Our first instantiation is based on the decisional Diffie-Hellman (DDH) assumption and (generic) simulation-sound non-interactive zero-knowledge (NIZK) proofs. (It could also be based on the quadratic residuosity assumption or the Paillier assumption, as in [25]. We omit further details.) Our second, more efficient construction is based on the decisional linear assumption [12] in groups with a bilinear map.

3.2.1 A Construction Based on the DDH Assumption

We first describe the encryption scheme and then the smooth projective hash function.

A CCA-secure encryption scheme. We apply the Naor-Yung/Sahai paradigm [39, 42] to the El Gamal encryption scheme. Briefly, the public key defines a group $\mathbb{G}$ of prime order p along with generators $g_1, h_1, g_2, h_2 \in \mathbb{G}$. The public key also contains a common random string crs for a (one-time) simulation-sound NIZK proof system [42]; refer to Appendix A.2 for a definition.

Fixing $\mathbb{G}$, let $\mathsf{ElGamal}_{g,h}(m)$ denote an El Gamal encryption of $m \in \mathbb{Z}_p$ with respect to (g, h); namely, $\mathsf{ElGamal}_{g,h}(m) \to (g^r,\ h^r \cdot g^m)$, where $r \in \mathbb{Z}_p$ is chosen uniformly at random. (Note: we put m in the exponent which means that efficient recovery of m is not possible. For our purposes all that is needed is to recover g^m, which can be done in the usual way.) To encrypt a message $m \in \mathbb{Z}_p$ in our CCA-secure scheme, the sender outputs the ciphertext $(\mathsf{ElGamal}_{g_1,h_1}(m),\ \mathsf{ElGamal}_{g_2,h_2}(m),\ \pi)$, where π is a simulation-sound NIZK proof that the same m is encrypted in both cases. Labels can be incorporated by including the label in the proof π; we omit the standard details.

Decryption of the ciphertext $(c_1, d_1, c_2, d_2, \pi)$ rejects if $c_1, d_1, c_2, d_2 \notin \mathbb{G}$ or if the proof π is invalid. (Note that the space of valid label/ciphertext pairs is efficiently recognizable without the secret key.) If the ciphertext is valid, then one of the two component ciphertexts is decrypted and the resulting message is output. The results of [42] show that this yields a CCA-secure (labeled) encryption scheme based on the DDH assumption and simulation-sound NIZK.

A smooth projective hash function. Fix $\mathbb{G}$ and a public key $pk = (g_1, h_1, g_2, h_2, \mathsf{crs})$ as above, and define sets X and $\{L_i\}$ as described in Section 2.2. Define a smooth projective hash function as follows. The set of keys K consists of all four-tuples of elements in $\mathbb{Z}_p$. Given a valid label/ciphertext pair $(\mathsf{label}, C = (c_1, d_1, c_2, d_2, \pi))$ and key $k = (x_1, y_1, x_2, y_2)$, the hash function is defined as:

$$H_{(x_1,y_1,x_2,y_2)}\big(\mathsf{label}, (c_1, d_1, c_2, d_2, \pi), pw\big) = c_1^{x_1} \cdot (d_1/g_1^{pw})^{y_1} \cdot c_2^{x_2} \cdot (d_2/g_2^{pw})^{y_2}\,.$$

(Thus, the range of H is the group $\mathbb{G}$ itself.) The projection function α is defined as:

$$\alpha(x_1, y_1, x_2, y_2) = g_1^{x_1} \cdot h_1^{y_1} \cdot g_2^{x_2} \cdot h_2^{y_2}.$$

A proof of the following is standard, and is given for completeness in Appendix C.2.

Lemma 2 $(K, \mathbb{G}, \mathcal{H} = \{H_k\}_{k \in K}, S, \alpha)$ *as defined above is a smooth projective hash function for the hard subset membership problem* $(X, \{L_i\})$.

3.2.2 A Construction Based on the Decisional Linear Assumption

We now present a more efficient construction based on bilinear maps. The efficiency advantage is obtained by using a *specific* simulation-sound NIZK proof system, built using techniques adapted from [29, 15]. Our construction here relies on the *decisional linear assumption* as introduced by Boneh et al. [12]; we refer the reader there for a precise statement of the assumption.

A CPA-secure encryption scheme. We start by describing a semantically secure encryption scheme, due to Boneh et al. [12], based on the decisional linear assumption; we then convert this into a CCA-secure encryption scheme via the same paradigm as above, but using an efficient simulation-sound NIZK proof system. The bilinear map itself is used only in the construction of the simulation-sound NIZK.

Fix groups $\mathbb{G}, \mathbb{G}_T$ of prime order p, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The public key is $pk = (f, g, h) \in \mathbb{G}^3$, and the secret key is (α, β) such that $f = h^{1/\alpha}$ and $g = h^{1/\beta}$. A message $m \in \mathbb{Z}_p$ is encrypted by choosing random $r, s \in \mathbb{Z}_p$ and computing the ciphertext $(f^r, g^s, h^{r+s} f^m)$. Given a ciphertext (c_1, c_2, c_3), we can recover f^m as $c_3/c_1^\alpha c_2^\beta$.

A simulation-sound NIZK proof of plaintext equality. We can construct a (one-time) simulation-sound NIZK proof of plaintext equality for the encryption scheme described above using the techniques of [29, 15]. Details of the construction (which, while not entirely straightforward, are not the focus of this work) are given in Appendix B.

A CCA-secure encryption scheme. We obtain a CCA-secure encryption scheme by using the Naor-Yung/Sahai paradigm, as described previously. (The following discussion relies on the results of Appendix B.) The public key consists of group elements (f_1, g_1, f_2, g_2, h) used for encryption, in addition to any group elements needed for the CRS of the simulation-sound NIZK proof. Encryption of m, as described in Appendix B, is done by choosing $r_1, s_1, r_2, s_2 \in \mathbb{Z}_p$ and computing the ciphertext

$$(f_1^{r_1},\ g_1^{s_1},\ h^{r_1+s_1} f_1^m,\ f_2^{r_2},\ g_2^{s_2},\ h^{r_2+s_2} f_1^m,\ \pi),$$

where π denotes a simulation-sound NIZK proof that the same m was encrypted both times. (Once again, the space of valid label/ciphertext pairs is efficiently recognizable without the secret key.) It follows from [39, 42] that this yields a CCA-secure scheme under the decisional linear assumption. Ciphertexts consist of 66 group elements altogether.

A smooth projective hash function. Fix $\mathbb{G}, \mathbb{G}_T$, and a public-key $pk = (f_1, g_1, f_2, g_2, h)$ as above, and define sets X and $\{L_i\}$ as in Section 2.2. We define a smooth projective hash function as follows. The set of keys K is the set of six-tuples of elements in $\mathbb{Z}_p$. Given a valid label/ciphertext pair $(\mathsf{label}, C = (c_1, d_1, e_1, c_2, d_2, e_2, \pi))$ and a key $k = (x_1, y_1, z_1, x_2, y_2, z_2) \in \mathbb{Z}_p^6$, the hash function is defined as

$$H_{(x_1,y_1,z_1,x_2,y_2,z_2)}(\mathsf{label}, C, pw) = c_1^{x_1} \cdot d_1^{y_1} \cdot (e_1/f_1^{pw})^{z_1} \cdot c_2^{x_2} \cdot d_2^{y_2} \cdot (e_2/f_2^{pw})^{z_2} .$$

(The range of H is $\mathbb{G}$ itself.) The projection function $\alpha : K \to \mathbb{G}^4$ is defined as:

$$\alpha(x_1, y_1, z_1, x_2, y_2, z_2) = (f_1^{x_1} h^{z_1},\ g_1^{y_1} h^{z_1},\ f_2^{x_2} h^{z_2},\ g_2^{y_2} h^{z_2}) .$$

In Appendix C.3 we show:

Lemma 3 $(K, \mathbb{G}, \mathcal{H} = \{H_k\}_{k \in K}, S, \alpha)$ *as defined above is a smooth projective hash function for the hard subset membership problem* $(X, \{L_i\})$.

4 A One-Round, Universally-Composable PAK Protocol

A definition of security for password-based authenticated key exchange in the universal composability (UC) framework [16] was given by Canetti et al. [17]. We review the UC framework in Appendix A.3, and include there the password-based key-exchange functionality $\mathcal{F}_{\mathsf{pwKE}}$ from [17]. (We also let $\hat{\mathcal{F}}_{\mathsf{pwKE}}$ denote the multi-session extension of $\mathcal{F}_{\mathsf{pwKE}}$.) This definition guarantees a strong, simulation-based notion of security that, in particular, guarantees that security is maintained even when multiple protocols are run concurrently in an arbitrary network. For the specific case of password-based key exchange, the definition also has the advantage of *automatically* handling arbitrary (efficiently sampleable) distributions on passwords, and even correlations between passwords of different users. We refer to [17] for a more complete discussion.

4.1 Overview of the Construction

We do not know how to prove that the protocol from Section 3.1 is universally composable. The main difficulty is that the definition of PAK in the UC framework requires simulation *even if the adversary guesses the correct password.* (In contrast, in the proof of security in Section 3.1 we simply "give up" in case this ever occurs.) To see the problem more clearly, consider what happens in the UC setting when the simulator sends the first message of the protocol to the adversary, before the simulator knows the correct password. The simulator must send *some* ciphertext C as part of the first message, and this "commits" the simulator to some password pw. When the adversary sends the reply, the simulator can extract the adversary's "password guess" pw' and submit this guess to the ideal functionality. If this turns out to be the correct password, however, the simulator is stuck: it needs to compute a session key that matches the session key the adversary would compute, but the simulator is (information-theoretically!) unable to do so because it sent an incorrect ciphertext in the first message.

In [17], this was resolved (roughly) by having one party send a "pre-commitment" to the password, and then running a regular PAK protocol along with a proof that the password being used in the protocol is the same as the password to which it "pre-committed". (The proof is set up in such a way that the simulator can equivocate this proof, but the adversary cannot.) This requires at least one additional round.

We take a different approach that does not affect the round complexity at all. Roughly, we modify the protocol from Figure 1 by having each party include as part of its message an encryption C_1 of its hash key k, along with a proof that C_1 encrypts a value k for which $\alpha(k) = s$. Now, even if the simulator is wrong in its guess of the password it will still be able to compute a session key by extracting this hash key from the adversary's message. A full description of the protocol is given in the following section.

While we do not describe in detail any instantiation of the components, we remark that it should be possible to use the same techniques as in Appendix B to construct (reasonably) efficient realizations of the necessary components using bilinear maps. We leave this for future work.

Public Parameters: $(pk_1, pk_2, \mathsf{crs})$

User U		User U'
$k \leftarrow K;\ s := \alpha(k)$		$k' \leftarrow K;\ s' := \alpha(k')$
$C_1 \leftarrow \mathsf{Enc}_{pk_1}(k)$		$C'_1 \leftarrow \mathsf{Enc}_{pk_1}(k')$
$\pi := \mathcal{P}_{\mathsf{crs}}((s, C_1) \in L^*)$		$\pi' := \mathcal{P}_{\mathsf{crs}}((s', C'_1) \in L^*)$
$\mathsf{label} := (\mathsf{ssid}, U, U', s, C_1, \pi)$		$\mathsf{label}' := (\mathsf{ssid}, U', U, s', C'_1, \pi')$
$C_2 \leftarrow \mathsf{Enc}_{pk_2}(\mathsf{label}, pw)$	$\xrightarrow{s, C_1, \pi, C_2}$	$C'_2 \leftarrow \mathsf{Enc}_{pk_2}(\mathsf{label}', pw)$
	$\xleftarrow{s', C'_1, \pi', C'_2}$	
$\mathsf{label}' := (\mathsf{ssid}, U', U, s', C'_1, \pi')$		$\mathsf{label} := (\mathsf{ssid}, U, U', s, C_1, \pi)$
$sk_U := H_k(\mathsf{label}', C'_2, pw)$		$sk_{U'} := H_k(\mathsf{label}', C'_2, pw)$
$\cdot H_{k'}(\mathsf{label}, C_2, pw)$		$\cdot H_{k'}(\mathsf{label}, C_2, pw)$

Figure 2: A universally composable protocol for password-based authenticated key exchange.

4.2 Description of the Protocol

In addition to the building blocks used in Section 3.1, here we also rely on an unbounded simulation-sound NIZK proof system $(\mathsf{CRSGen}, \mathcal{P}, \mathcal{V})$ for a language L^* defined below; see Appendix A.2.

Public parameters. The public parameters consist of two public keys pk_1 and pk_2 generated by $\mathsf{Gen}(1^n)$ and a common random string crs for the simulation-sound NIZK proof system. Let $(K, G, \mathcal{H} = \{H_k : X \to G\}_{k \in K}, S, \alpha : K \to S)$ be a smooth projective hash function for pk_2.

Protocol execution. Consider an execution of the protocol between users U and $U' \neq U$ holding a shared password pw and a common session identifier ssid. (The ssid is an artefact of the UC framework, and it is guaranteed that (1) parties communicating with each other begin holding matching ssids, and (2) each ssid is used only once. We remark that the presence of these ssids is not essential to our proof of security, though it does make the proof somewhat simpler.) Our protocol is symmetric, and so we describe the execution from the point of view of U; see Figure 2.

First, U chooses a random hash key $k \leftarrow K$ and computes $s := \alpha(k)$. It then computes an encryption of k, namely $C_1 \leftarrow \mathsf{Enc}_{pk_1}(k)$. Define a language L^* as follows.

$$L^* \stackrel{\text{def}}{=} \{(s, C_1) : \exists k \in K \text{ and } \omega \text{ s.t } s = \alpha(k) \text{ and } C_1 = \mathsf{Enc}_{pk_1}(k; \omega)\}.$$

U computes an NIZK proof π that $(C_1, s) \in L^*$, using crs. It then sets $\mathsf{label} := (\mathsf{ssid}, U, U', s, C_1, \pi)$ and computes the ciphertext $C_2 \leftarrow \mathsf{Enc}_{pk_2}(\mathsf{label}, pw)$. The message it sends is (s, C_1, π, C_2).

Upon receiving the message (s', C'_1, π', C'_2), user U does the following. If the message is invalid (i.e., if verification of π' fails, or C'_2 is not a valid ciphertext, or $s' \notin S$), then U simply rejects. Otherwise, U sets $\mathsf{label}' := (\mathsf{ssid}, U', U, s', C'_1, \pi')$ and computes $\mathsf{sk}_U := H_k(\mathsf{label}', C'_2, pw) \cdot H_{k'}(\mathsf{label}, C_2, pw)$. Note U can compute $H_k(\mathsf{label}', C'_2, pw)$ since it knows k, and can compute $H_{k'}(\mathsf{label}, C_2, pw)$ using $s' = \alpha(k')$ and the randomness used to generate C_2. Correctness follows from the definition of smooth projective hashing. A proof of the following is given in Appendix C.4.

Theorem 2 *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a CCA-secure encryption scheme,* $(\mathsf{CRSGen}, \mathcal{P}, \mathcal{V})$ *is an unbounded simulation-sound NIZK proof system, and* $(K, G, \mathcal{H} = \{H_k : X \to G\}_{k \in K}, S, \alpha : K \to S)$ *is a smooth projective hash family, then the protocol in Figure 2 securely realizes* $\hat{\mathcal{F}}_{\mathsf{pwKE}}$ *in the* $\mathcal{F}_{\mathsf{crs}}$*-hybrid model.*

References

[1] M. Abdalla, D. Catalano, C. Chevalier, and D. Pointcheval. Efficient two-party password-based key exchange protocols in the UC framework. In T. Malkin, editor, *Cryptographers' Track — RSA 2008*, LNCS, pages 335–351. Springer, Apr. 2008.

[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, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 419–428. ACM Press, 1998.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic design of two-party authentication protocols. *IEEE J. on Selected Areas in Communications*, 11(5):679–693, 1993.

[9] M. Blum, A. De Santis, S. Micali, and G. Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[10] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 103–112. ACM Press, 1988.

[11] M. Blum, P. Feldman, and S. Micali. Proving security against chosen cyphertext attacks. In *Advances in Cryptology — Crypto '88*, volume 403 of *LNCS*, pages 256–268. Springer, 1990.

[12] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[13] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of *LNCS*, pages 147–163. Springer, 2005.

[14] 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.

[15] J. Camenisch, N. Chandran, and V. Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In *Advances in Cryptology — Eurocrypt 2009*, volume 5479 of *LNCS*, pages 351–368. Springer, 2009. Available at `http://eprint.iacr.org/2008/375`.

[16] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE, 2001.

[17] 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.

[18] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology — Eurocrypt 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, 2001.

[19] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology — Eurocrypt 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, 2002.

[20] 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.

[21] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology — Crypto 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, 2001.

[22] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[23] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[24] 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.

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

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

[27] 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.

[28] A. Groce and J. Katz. A new framework for efficient password-based authenticated key exchange. In *17th ACM Conf. on Computer and Communications Security (ACM CCCS 2010)*, pages ???–??? ACM Press, 2010.

[29] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology — Eurocrypt 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

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

[31] I. R. Jeong, J. Katz, and D. H. Lee. One-round protocols for two-party authenticated key exchange. In *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security (ACNS)*, volume 3089 of *LNCS*, pages 220–232. Springer, 2004.

[32] 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.

[33] 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.

[34] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Advances in Cryptology — Eurocrypt 2001*, volume 2045 of *LNCS*, pages 475–494. Springer, 2001.

[35] J. Katz and V. Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In *Advances in Cryptology — Asiacrypt 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, 2009.

[36] H. Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *Advances in Cryptology — Crypto 2003*, volume 2729 of *LNCS*, pages 400–425. Springer, 2003.

[37] H. Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. In *Advances in Cryptology — Crypto 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, 2005.

[38] 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.

[39] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 33–43. ACM Press, 1989.

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

[41] T. Okamoto. Authenticated key exchange and key encapsulation in the standard model (invited talk). In *Advances in Cryptology — Asiacrypt 2007*, volume 4833 of *LNCS*, pages 474–484. Springer, 2007.

[42] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 543–553. IEEE, 1999.

[43] V. Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 1999. Available at `http://eprint.iacr.org/1999/012`.

A Additional Definitions

A.1 Labeled CCA-Secure Encryption

We use the standard notion of chosen-ciphertext security for public-key encryption, though adapted to support the inclusion of *labels* when generating ciphertexts.

Definition 2 *A* public-key encryption scheme *supporting labels is a tuple of* PPT *algorithms* (Gen, Enc, Dec) *such that:*

- *The* key generation algorithm Gen *takes as input a security parameter* 1^n *and returns a public key* pk *and a secret key* sk.
- *The* encryption algorithm Enc *takes as input a public key* pk, *a label* label, *and a message* m. *It returns a ciphertext* $C \leftarrow \mathbb{E}_{pk}(\mathsf{label}, m)$.
- *The* decryption algorithm Dec *takes as input a secret key* sk, *a label* label, *and a ciphertext* C. *It returns a message* m *or a distinguished symbol* $\perp$. *We write this as* $m = \mathcal{D}_{sk}(\mathsf{label}, C)$.

We require that for all pk, sk *output by* $gen(1^n)$, *any* $\mathsf{label} \in \{0,1\}^*$, *all* m *in the (implicit) message space, and any* C *output by* $\mathsf{Enc}_{pk}(\mathsf{label}, m)$ *we have* $\mathsf{Dec}_{sk}(\mathsf{label}, C) = m$.

Our definition of security against chosen-ciphertext attacks is standard except for our inclusion of labels. In the following, we define a left-or-right encryption oracle $\mathbb{E}_{pk,b}(\cdot,\cdot,\cdot)$ (where $b \in \{0,1\}$) as follows:

$$\mathsf{Enc}_{pk,b}(\mathsf{label}, m_0, m_1) \stackrel{\text{def}}{=} \mathsf{Enc}_{pk}(\mathsf{label}, m_b).$$

Definition 3 *A public-key encryption scheme* (Gen, Enc, Dec) *is* secure against adaptive chosen-ciphertext attacks *(*CCA-secure*) if the following is negligible for all* PPT *algorithms* A*:*

$$\left| 2 \cdot \Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^n); b \leftarrow \{0,1\} : A^{\mathsf{Enc}_{pk,b}(\cdot,\cdot,\cdot), \mathsf{Dec}_{sk}(\cdot,\cdot)}(1^n, pk) = b] - 1 \right|,$$

where A*'s queries are restricted as follows: if* A *makes a query* $\mathbb{E}_{pk,b}(\mathsf{label}, m_0, m_1)$ *then* $|m_0| = |m_1|$*; furthermore, if* A *receives ciphertext* C *in response to this query, then* A *cannot later query* $\mathsf{Dec}_{sk}(\mathsf{label}, C)$ *(but it is allowed to query* $(\mathsf{Dec}_{sk}(\mathsf{label}', C')$ *with* $C' = C$ *and* $\mathsf{label}' \neq \mathsf{label}$*).*

A.2 Simulation-Sound Non-Interactive Zero Knowledge (NIZK)

Simulation-sound NIZK was introduced in [42, 21]. Intuitively, a simulation-sound NIZK proof system is a NIZK proof system with the extra property that a polynomially bounded cheating prover is incapable of convincing the verifier of a false statement, even after seeing any number of simulated proofs of her choosing. We first recall the notion of (adaptive) NIZK:

Definition 4 ([23, 11, 9]) *A tuple of* PPT *algorithms* $\Pi = (\mathsf{CRSGen}, \mathcal{P}, \mathcal{V}, \mathcal{S}_1, \mathcal{S}_2)$ *is an* efficient NIZK proof system *for a language* $L \in \mathcal{NP}$ *with witness relation* R *if the following hold:*

- **Completeness**: *For all* n, *all* $x \in L \cap \{0,1\}^n$, *all* w *such that* $R(x, w) = 1$, *and all strings* $\sigma \leftarrow \mathsf{CRSGen}(1^n)$, *it holds that* $\mathcal{V}_\sigma(x, \mathcal{P}_\sigma(x, w)) = 1$.

- **Adaptive Soundness**: *For all adversaries* $\mathcal{A}$, *the following is negligible (in* n*):*

$$\Pr[\sigma \leftarrow \mathsf{CRSGen}(1^n); (x, \pi) \leftarrow \mathcal{A}(\sigma) : V_\sigma(x, \pi) = 1 \wedge \pi \notin L].$$

- **Adaptive Zero Knowledge**: *For all* PPT *adversaries* $\mathcal{A}$, *the following is negligible*

$$\left| \Pr[\mathsf{Expt}_{\mathcal{A}}(n) = 1] - \Pr[\mathsf{Expt}^{Sim}_{\mathcal{A}}(n) = 1] \right|,$$

where experiment $\mathsf{Expt}_{\mathcal{A}}(n)$ *is defined as:*

$$\begin{array}{l} \sigma \leftarrow \mathsf{CRSGen}(1^n) \\ \textit{Return } \mathcal{A}^{\mathcal{P}_\sigma(\cdot,\cdot)}(1^n, \sigma) \end{array}$$

and experiment $\mathsf{Expt}^{Sim}_{\mathcal{A}}(n)$ *is defined as:*

$$\begin{array}{l} (\sigma, \tau) \leftarrow \mathcal{S}_1(1^n) \\ \textit{Return } \mathcal{A}^{\mathcal{S}'_{\sigma,\tau}(\cdot,\cdot)}(1^n, \sigma), \end{array}$$

where $S'_{\sigma,\tau}(x, w) = \left\{ \begin{array}{ll} \mathcal{S}_2(x, \sigma, \tau) & R(x, w) = 1 \wedge x \in \{0,1\}^n \\ \perp & \textit{otherwise} \end{array} \right.$.

We next define the notion of simulation-sound NIZK. (Note that although one-time simulation soundness would suffice for our applications in Section 3.2, we only define the stronger notion of unbounded simulation-sound NIZK. See [42] for a definition of the former.)

Definition 5 *Let* $\Pi = (\mathsf{CRSGen}, \mathcal{P}, \mathcal{V}, \mathcal{S}_1, \mathcal{S}_2)$ *be an efficient NIZK proof system for a language* $L \in \mathcal{NP}$. *We say* Π *is* simulation sound *if for all* PPT *adversaries* $\mathcal{A}$ *it holds that* $\Pr[\mathsf{Expt}_{\mathcal{A},\Pi}(n) = 1]$ *is negligible, where* $\mathsf{Expt}_{\mathcal{A},\Pi}(n)$ *denotes the following experiment:*

$$\begin{array}{l} (\sigma, \tau) \leftarrow \mathcal{S}_1(1^n) \\ (x, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_2(\cdot,\sigma,\tau)}(1^n, \sigma) \\ \textit{Let } Q \textit{ be the list of proofs returned by } \mathcal{S}_2\textit{, above} \\ \textit{Return } 1 \textit{ iff } (\pi \notin Q \textit{ and } x \notin L \textit{ and } \mathcal{V}_\sigma(x, \pi) = 1)\,. \end{array}$$

Assuming the existence of enhanced trapdoor permutations, every language in $\mathcal{NP}$ has a simulation-sound NIZK proof system [21].

A.3 The Universal Composability Framework

We provide a brief review of the universally composable security framework [16]. The framework allows for defining the security properties of cryptographic tasks so that security is maintained under general composition with an unbounded number of instances of arbitrary protocols running concurrently. In the UC framework, the security requirements of a given task are captured by specifying an ideal functionality run by a "trusted party" that obtains the inputs of the participants and provides them with the desired outputs. Informally, then, a protocol securely carries out a given task if running the protocol in the presence of a real-world adversary amounts to "emulating" the desired ideal functionality.

The notion of emulation in the UC framework is considerably stronger than that considered in previous models. As usual, the real-world model includes the parties running the protocol and an adversary $\mathcal{A}$ who controls their communication and potentially corrupts parties, while the ideal-world includes a simulator $\mathcal{S}$ who interacts with an ideal functionality $\mathcal{F}$ and also with dummy players who simply send input to/receive output from $\mathcal{F}$; "emulating an ideal process" requires that for any adversary $\mathcal{A}$ there should exist a simulator $\mathcal{S}$ that causes the outputs of the parties in the ideal process to have a "similar" (i.e., computationally-indistinguishable) distribution to the outputs of the parties in a real-world execution of the protocol. In the UC framework, the requirement on $\mathcal{S}$ is more stringent than this. Specifically, there is also an additional entity called the *environment* $\mathcal{Z}$. This environment generates the inputs to all parties, observes all their outputs, and interacts with the adversary in an arbitrary way throughout the computation. A protocol π is said to *securely realize* an ideal functionality $\mathcal{F}$ if for any real-world adversary $\mathcal{A}$ that interacts with $\mathcal{Z}$ and real players running π, there exists an ideal-world simulator $\mathcal{S}$ that interacts with $\mathcal{Z}$, the ideal functionality $\mathcal{F}$, and the "dummy" players communicating with $\mathcal{F}$, such that *no* poly-time environment $\mathcal{Z}$ can distinguish whether it is interacting with $\mathcal{A}$ (in the real world) or $\mathcal{S}$ (in the ideal world). $\mathcal{Z}$ thus serves as an "interactive distinguisher" between a real-world execution of the protocol π and an ideal execution of functionality $\mathcal{F}$. A key point is that $\mathcal{Z}$ cannot be re-wound by $\mathcal{S}$; in other words, $\mathcal{S}$ must provide a so-called "straight-line" simulation.

The following *universal composition theorem* is proven in [16]. Consider a protocol π that operates in the $\mathcal{F}$-$\mathsf{hybrid\ model}$, where parties can communicate as usual and in addition have ideal access to an unbounded number of *copies* of the functionality $\mathcal{F}$. Let ρ be a protocol that securely realizes $\mathcal{F}$ as sketched above, and let π^ρ be identical to π with the exception that the interaction with *each copy* of $\mathcal{F}$ is replaced with an interaction with a *separate instance* of ρ. Then, π and π^ρ have essentially the same input/output behavior. In particular, if π securely realizes some functionality $\mathcal{G}$ in the $\mathcal{F}$-hybrid model then π^ρ securely realizes $\mathcal{G}$ in the standard model (i.e., without access to any functionality).

A.3.1 Universally-Composable Password-Based Key Exchange

In Figure 3 we present the functionality $\mathcal{F}_{\mathsf{pwKE}}$ for password-based key exchange, taken directly from [17]. We refer the reader to their work for extensive discussion regarding the particular choices made regarding this formulation of this functionality.

Functionality $\mathcal{F}_{\mathsf{pwKE}}$

The functionality $\mathcal{F}_{\mathsf{pwKE}}$ is parameterized by a security parameter n. It interacts with an adversary $\mathcal{S}$ and a set of parties via the following queries:

Upon receiving a query $(\mathsf{NewSession}, \mathsf{sid}, P_i, P_j, pw)$ **from party** P_i**:**

Send $(\mathsf{NewSession}, \mathsf{sid}, P_i, P_j)$ to $\mathcal{S}$. In addition, if this is the first NewSession query, or if this is the second NewSession query and there is a record (P_j, P_i, pw'), then record (P_i, P_j, pw) and mark this record fresh.

Upon receiving a query $(\mathsf{TestPwd}, \mathsf{sid}, P_i, pw')$ **from the adversary** $\mathcal{S}$**:**

If there is a record of the form (P_i, P_j, pw) which is fresh, then do: If $pw' = pw$, mark the record compromised and reply to $\mathcal{S}$ with "correct guess". If $pw \neq pw'$, mark the record interrupted and reply with "wrong guess".

Upon receiving a query $(\mathsf{NewKey}, \mathsf{sid}, P_i, \mathrm{SK})$ **from** $\mathcal{S}$**, where** $|\mathrm{SK}| = n$**:**

If there is a record of the form (P_i, P_j, pw), and this is the first NewKey query for P_i, then:

- If this record is compromised, or either P_i or P_j is corrupted, then output $(\mathsf{sid}, \mathrm{SK})$ to player P_i.
- If this record is fresh, and there is a record (P_j, P_i, pw') with $pw' = pw$, and a key SK' was sent to P_j, and (P_j, P_i, pw) was fresh at the time, then output $(\mathsf{sid}, \mathrm{SK}')$ to P_i.
- In any other case, pick a new random key $\mathrm{SK}' \leftarrow \{0,1\}^n$ and send $(\mathsf{sid}, \mathrm{SK}')$ to P_i.

Either way, mark the record (P_i, P_j, pw) as completed.

Figure 3: The password-based key-exchange functionality $\mathcal{F}_{\mathsf{pwKE}}$.

B A Simulation-Sound NIZK Proof of Plaintext Equality

Fix groups $\mathbb{G}, \mathbb{G}_T$ of prime order p, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ as in Section 3.2.2. Fix also two public keys $pk_1 = (f_1, g_1, h)$ and $pk_2 = (f_2, g_2, h)$. We encrypt a message m with respect to pk_1 by choosing random r, s and computing the ciphertext $(f_1^r, g_1^s, h^{r+s} f_1^m)$. We encrypt a message m with respect to pk_2 by choosing random $r, s \in \mathbb{Z}_p$ and computing the ciphertext $(f_2^r, g_2^s, h^{r+s} f_1^m)$. We stress that the public keys use the same value h, and that the group element f_1^m is used when encrypting with respect to both public keys.

We first describe a (potentially malleable) NIZK proof of plaintext equality. That is, given two ciphertexts (F_1, G_1, H_1) and (F_2, G_2, H_2) encrypted with respect to pk_1, pk_2, respectively, we describe a proof that these ciphertexts encrypt the same message. The observation is that plaintext equality is equivalent to the existence of $r_1, s_1, r_2, s_2 \in \mathbb{Z}_p$ such that:

$$\begin{aligned} F_1 &= f_1^{r_1} && (3)\\ G_1 &= g_1^{s_1} && (4)\\ F_2 &= f_2^{r_2} && (5)\\ G_2 &= g_2^{s_2} && (6)\\ H_1/H_2 &= h^{r_1+s_1-r_2-s_2}. && (7) \end{aligned}$$

As shown in [29] (see also [15, Section 4.4] for a self-contained description), NIZK proofs of satisfiability (with a CRS) can be constructed for a system of equations as above; since, in our case, we have 5 (linear) equations in 4 variables, proofs would contain 22 group elements.[2]

[2] Our calculations here are based on the decisional linear assumption (the 2-linear assumption in the terminology of [15]). If we are willing to use the 1-linear assumption, the efficiency of our proofs can be improved.

Camenisch et al. [15] show a construction of an *unbounded* simulation-sound NIZK. For our purposes, a simpler construction that is *one-time* simulation sound [42] will suffice. Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a one-time signature scheme, where for simplicity we assume verification keys are elements of $\mathbb{G}$ (this can always be achieved using an extra step of hashing). To make the above (one-time) simulation-sound, we add to the CRS group elements (f, g, h, F, G, H). Roughly, proofs of plaintext equality now contain:

1. A fresh signature verification key vk.
2. A proof that *either* there exists a satisfying assignment to Equations (3)–(7), *or* that the given tuple (f, g, h, F, G, H) is an encryption of vk. I.e., there exist r, s such that:
$$F = f^r, \qquad G = g^s, \qquad H/vk = h^{r+s}. \tag{8}$$
3. A signature σ (with respect to vk) on the proof from the previous step.

Noting that Equation (8) describes a system of 3 (linear) equations in 2 variables, and using the techniques from [15, Appendix A.2], an NIZK proof as required in step 2 can be done using 58 group elements, for a total of 60 group elements for the entire simulation-sound NIZK proof (assuming signatures are one group element for simplicity). See also footnote 2.

C Deferred Proofs

C.1 Proof of Lemma 1

Let ℓ' be a (polynomial) bound on the number of encryption queries asked by $\mathcal{A}$. Then when $b = 0$, the values given to $\mathcal{A}$ include pk, $s_1, \ldots, s_\ell$, the ciphertexts $C_1, \ldots, C_{\ell'}$, and the values

$$\begin{pmatrix} H_{k_1}(\mathsf{label}_1, C_1, pw_1) & \cdots & \cdots & H_{k_1}(\mathsf{label}_{\ell'}, C_{\ell'}, pw_{\ell'}) \\ \vdots & \ddots & \ddots & \vdots \\ H_{k_\ell}(\mathsf{label}_1, C_1, pw_1) & \cdots & \cdots & H_{k_\ell}(\mathsf{label}_{\ell'}, C_{\ell'}, pw_{\ell'}) \end{pmatrix}.$$

We show that this is computationally indistinguishable from the experiment where $\mathcal{A}$ gets pk, $s_1, \ldots, s_\ell$, ciphertexts $C_1, \ldots, C_{\ell'}$, and a matrix of $\ell \cdot \ell'$ uniform and independent elements of G.

To prove this we show that, for arbitrary i, j, the experiment in which $\mathcal{A}$ is given

$$\begin{pmatrix} g_{1,1} & & \vdots & & \vdots \\ \vdots & & g_{i-1,j} & & \vdots \\ \vdots & \cdots & H_{k_i}(\mathsf{label}_j, C_j, pw_j) & \cdots & \vdots \\ \vdots & & H_{k_{i+1}}(\mathsf{label}_j, C_j, pw_j) & & \vdots \\ g_{\ell,1} & & \vdots & & H_{k_\ell}(\mathsf{label}_{\ell'}, C_{\ell'}, pw_{\ell'}) \end{pmatrix}$$

is computationally indistinguishable from the experiment in which $\mathcal{A}$ is given

$$\begin{pmatrix} g_{1,1} & & \vdots & & \vdots \\ \vdots & & g_{i-1,j} & & \vdots \\ \vdots & \cdots & g_{i,j} & \cdots & \vdots \\ \vdots & & H_{k_{i+1}}(\mathsf{label}_j, C_j, pw_j) & & \vdots \\ g_{\ell,1} & & \vdots & & H_{k_\ell}(\mathsf{label}_{\ell'}, C_{\ell'}, pw_{\ell'}) \end{pmatrix},$$

where the $g_{a,b}$ denote uniform and independent elements of G. (In both cases, $\mathcal{A}$ is also given pk, $s_1, \ldots, s_\ell$, and $C_1, \ldots, C_{\ell'}$, and may access the decryption oracle as in the original experiment.) Once we show this, the lemma follows by a standard hybrid argument.

We denote the first experiment, in which the (i,j)th entry of the matrix is computed as $H_{k_i}(\mathsf{label}_j, C_j, pw_j)$, by $\mathsf{real}_{i,j}$; we denote the second experiment, in which the (i,j)th entry of the matrix is a random $g_{i,j}$, by $\mathsf{rand}_{i,j}$. To prove that these two experiments are indistinguishable, we introduce two additional experiments. Experiment $\mathsf{real}'_{i,j}$ (resp., $\mathsf{rand}'_{i,j}$) is identical to $\mathsf{real}_{i,j}$ (resp., $\mathsf{rand}_{i,j}$) except that now the jth ciphertext C_j returned by the encryption oracle is computed as an encryption of 0 (i.e., $C_j \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, 0)$). It follows immediately from the CCA-security of the encryption scheme that $\mathsf{real}_{i,j}$ and $\mathsf{real}'_{i,j}$ (resp., $\mathsf{rand}_{i,j}$ and $\mathsf{rand}'_{i,j}$) are computationally indistinguishable.

To complete the proof that $\mathsf{real}_{i,j}$ and $\mathsf{rand}_{i,j}$ are computationally indistinguishable, we show that $\mathsf{real}'_{i,j}$ and $\mathsf{rand}'_{i,j}$ are statistically close. To see this, consider the following experiment involving an algorithm B who is given $s_i = \alpha(k_i)$ for unknown (random) k_i, outputs $(\mathsf{label}, C, pw) \in X \setminus L$, and is given in response an element $h_{i,j} \in G$:

1. Compute $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and choose $k_i \leftarrow K$. Give B the values (pk, sk) and $s_i = \alpha(k_i)$.
2. B internally runs $\mathcal{A}$ on initial input pk. (Note that B can easily answer decryption queries of $\mathcal{A}$ using sk.) Then B samples $k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_\ell \leftarrow K$, sets $s_m = \alpha(k_m)$ for all $m \neq i$, and gives $s_1, \ldots, s_\ell$ to $\mathcal{A}$.
3. When $\mathcal{A}$ queries its encryption oracle with (label, pw), we have B act as follows:
   (a) For the kth such query, with $k < j$, we have B compute $C_k \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw)$ and give to $\mathcal{A}$ the ciphertext C_k along with random $g_{1,k}, \ldots, g_{\ell,k} \leftarrow G$.
   (b) For the jth such query (so $(\mathsf{label}_j, pw_j) = (\mathsf{label}, pw)$), B computes $C_j \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}_j, 0)$, outputs $(\mathsf{label}_j, C_j, pw)$, and receives $h_{i,j}$. It then chooses $g_{1,j}, \ldots, g_{i-1,j} \leftarrow G$, and gives to $\mathcal{A}$ the values $g_{1,j}, \ldots, g_{i-1,j}, h_{i,j}, H_{k_{i+1}}(\mathsf{label}_j, C_j, pw_j), \ldots, H_{k_\ell}(\mathsf{label}_j, C_j, pw_j)$. Note that B can compute these latter values since it knows $k_{i+1}, \ldots, k_\ell$.
   (c) For the kth such query, with $k > j$, we have B compute $C_k \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw)$ and give to $\mathcal{A}$ the ciphertext C_k along with $H_{k_1}(\mathsf{label}, C_k, pw), \ldots, H_{k_\ell}(\mathsf{label}, C_k, pw)$. Note that B can compute $H_{k_i}(\mathsf{label}, C_k, pw)$, even though it does not know k_i, because of the fact that $(\mathsf{label}, C_k, pw) \in L_{pw}$ and B knows the randomness used to compute C_k.

Observe that the view of $\mathcal{A}$ is distributed according to $\mathsf{real}'_{i,j}$ if $h_{i,j} = H_{k_i}(\mathsf{label}_j, C_j, pw)$, and is distributed according to $\mathsf{rand}'_{i,j}$ if $h_{i,j}$ is chosen uniformly from G. It follows from Equation (2) that $\mathsf{real}'_{i,j}$ and $\mathsf{rand}'_{i,j}$ are statistically close.

C.2 Proof of Lemma 2

Sampling a uniform $k \in K$, computing $H_k(x)$ given $k \in K$ and $x \in X$, and computing $\alpha(k)$ for $k \in K$ are all easy.

We now show that if $(\mathsf{label}, C, pw) \in L$, then $H_k(\mathsf{label}, C, pw)$ can be computed efficiently given $\alpha(k)$ and the randomness used to generate C. Since $(\mathsf{label}, C, pw) \in L$, we have that $C = (g_1^{r_1},\ h_1^{r_1} g_1^{pw},\ g_2^{r_2},\ h_2^{r_2} g_2^{pw},\ \pi)$ for some $r_1, r_2 \in \mathbb{Z}_p$. For $k = (x_1, y_1, x_2, y_2)$ we have

$$\begin{aligned} H_k(\mathsf{label}, C, pw) &= c_1^{x_1} \cdot (d_1/g_1^{pw})^{y_1} \cdot c_2^{x_2} \cdot (d_2/g_2^{pw})^{y_2} \\ &= g_1^{r_1 x_1} \cdot (h_1)^{r_1 y_1} \cdot g_2^{r_2 x_2} \cdot (h_2)^{r_2 y_2} \\ &= (g_1^{x_1} h_1^{y_1})^{r_1} \cdot (g_2^{x_2} h_2^{y_2})^{r_2}\,. \end{aligned}$$

This can be computed easily given r_1, r_2 and $\alpha(k) = (g_1^{x_1} h_1^{y_1},\ g_2^{x_2} h_2^{y_2})$.

Next, we show that if $(\mathsf{label}, C, pw) \in X \setminus L$, then the value of $H_k(\mathsf{label}, C, pw)$ is uniform conditioned on $\alpha(k)$. (This holds even if (label, C, pw) are chosen adaptively depending on $\alpha(k)$.) Fix any $\alpha(k) = (s_1, s_2)$. This constrains $k = (x_1, y_1, x_2, y_2)$ to satisfy

$$x_1 + \log_{g_1} h_1 \cdot y_1 = \log_{g_1} s_1 \tag{9}$$

$$x_2 + \log_{g_2} h_2 \cdot y_2 = \log_{g_2} s_2, \tag{10}$$

where the above are modulo the group order p. For any $(\mathsf{label}, C, pw) \in X \setminus L$, we can write $C = (g_1^{r_1}, h_1^{r_1} g_1^{pw'}, g_2^{r_2}, h_2^{r_2} g_2^{pw'}, \pi)$ for some $pw' \neq pw$. (We assume for simplicity that the same pw' is encrypted twice; since π is valid, this is the case with all but negligible probability.) Then:

$$\begin{aligned} H_k(\mathsf{label}, C, pw) &= g_1^{r_1 x_1} \cdot \left(h_1 \cdot g_1^{pw - pw'}\right)^{r_1 y_1} \cdot g_2^{r_2 x_2} \cdot \left(h_2 \cdot g_2^{pw' - pw}\right)^{r_2 y_2} \\ &= g_1^{r_1 x_1} \cdot h_1^{r_1' y_1} \cdot g_2^{r_2 x_2} \cdot h_2^{r_2' y_2}, \end{aligned}$$

for some $r_1' \neq r_1$ and $r_2' \neq r_2$. So, for any $g \in \mathbb{G}$, we have $H_k(\mathsf{label}, C, pw) = g$ iff

$$r_1 \cdot x_1 + (r_1' \cdot \log_{g_1} h_1) \cdot y_1 + (r_2 \cdot \log_{g_1} g_2) \cdot x_2 + (r_2' \cdot \log_{g_1} h_2) \cdot y_2 = \log_{g_1} g.$$

Since this equation in the unknowns x_1, y_1, x_2, y_2 is linearly independent of Equations (9) and (10), we see that the probability that $H_k(\mathsf{label}, C, pw) = g$ is exactly $1/|\mathbb{G}|$, and so the distribution of $H_k(\mathsf{label}, C, pw)$ is uniform in $\mathbb{G}$.

C.3 Proof of Lemma 3

Sampling a uniform $k \in K$, computing $H_k(x)$ given $k \in K$ and $x \in X$ and computing $\alpha(k)$ for $k \in K$ are all easy.

We show that if $(\mathsf{label}, C, pw) \in L$, then $H_k(\mathsf{label}, C, pw)$ can be computed efficiently given $\alpha(k)$ and the randomness used to generate C. Since $(\mathsf{label}, C, pw) \in L$, we have that $C = (f_1^{r_1}, g_1^{s_1}, h^{r_1+s_1} f_1^{pw}, f_2^{r_2}, g_2^{s_2}, h^{r_2+s_2} f_2^{pw})$ for some $r_1, s_1, r_2, s_2 \in \mathbb{Z}_p$. For $k = (x_1, y_1, z_1, x_2, y_2, z_2)$ we have

$$\begin{aligned} H_k(\mathsf{label}, C, pw) &= c_1^{x_1} d_1^{y_1} \cdot (e_1/f_1^{pw})^{z_1} \cdot c_2^{x_2} \cdot d_2^{y_2} \cdot (e_2/f_2^{pw})^{z_2} \\ &= (f_1^{x_1} h^{z_1})^{r_1} \cdot (g_1^{y_1} h^{z_1})^{s_1} \cdot (f_2^{x_2} h^{z_2})^{r_2} \cdot (g_2^{y_2} h^{z_2})^{s_2}\,. \end{aligned}$$

This can be computed easily given r_1, s_1, r_2, s_2 and $\alpha(k) = (f_1^{x_1} h^{z_1},\ g_1^{y_1} h^{z_1},\ f_2^{x_2} h^{z_2},\ g_2^{y_2} h^{z_2})$.

Next, we show that if $(\mathsf{label}, C, pw) \in X \setminus L$, then the value of $H_k(\mathsf{label}, C, pw)$ is uniform conditioned on $\alpha(k)$. (This holds even if (label, C, pw) are chosen adaptively depending on $\alpha(k)$.) Fix any $\alpha(k) = (S_1, S_2, S_3, S_4)$. Letting $\alpha_i = \log_h f_i$ and $\beta_i = \log_h g_i$, this value of $\alpha(k)$ constrains $k = (x_1, y_1, z_1, x_2, y_2, z_2)$ to satisfy

$$\begin{pmatrix} \alpha_1 & 0 & 1 & 0 & 0 & 0 \\ 0 & \beta_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_2 & 0 & 1 \\ 0 & 0 & 0 & 0 & \beta_2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix}, \tag{11}$$

where $\gamma_i = \log_h S_i$. For any $(\mathsf{label}, C, pw) \in X \setminus L$, we can write

$$C = (f_1^{r_1},\ g_1^{s_1},\ h^{r_1+s_1} f_1^{pw'},\ f_2^{r_2},\ g_2^{s_2},\ h^{r_2+s_2} f_2^{pw'},\ \pi)$$

for some $pw' \neq pw$. (We assume for simplicity that the same pw' is encrypted twice; since π is valid, this is the case with all but negligible probability.) We then have

$$\begin{aligned} H_k(\mathsf{label}, C, pw) &= f_1^{r_1 x_1} \cdot g_1^{s_1 y_1} \cdot h^{(r_1+s_1) z_1} \cdot \left(f_1^{\Delta}\right)^{z_1} \cdot f_2^{r_2 x_2} \cdot g_2^{s_2 y_2} \cdot h^{(r_2+s_2) z_2} \cdot \left(f_2^{\Delta}\right)^{z_2} \\ &= S_1^{r_1} S_2^{s_1} S_3^{r_2} S_4^{s_2} \cdot \left(f_1^{z_1} f_2^{z_2}\right)^{\Delta}, \end{aligned} \tag{12}$$

where $\Delta = pw' - pw \neq 0$. For any $g \in \mathbb{G}$, we have $f_1^{z_1} f_2^{z_2} = g$ iff

$$\alpha_1 \cdot z_1 + \alpha_2 \cdot z_2 = \log_h g. \tag{13}$$

Since the system of equations given by (11) and (13) is under-defined, the probability that $f_1^{z_1} f_2^{z_2} = g$ is exactly $1/|\mathbb{G}|$ even conditioned on the value $\alpha(k)$. Looking at Equation (12), and noting that $S_1^{r_1} S_2^{s_1} S_3^{r_2} S_4^{s_2}$ is entirely determined by $\alpha(k)$ and C, we conclude that the distribution of $H_k(\mathsf{label}, C, pw)$ is uniform in $\mathbb{G}$.

C.4 Proof of Theorem 2

Let $\mathcal{A}$ be an adversary that interacts with the parties running the protocol. We construct an ideal-world adversary (simulator) $\mathcal{S}$ interacting with the ideal functionality $\hat{\mathcal{F}}_{\mathsf{pwKE}}$, such that no PPT environment $\mathcal{Z}$ can distinguish an interaction with $\mathcal{A}$ in the real world from an interaction with $\mathcal{S}$ in the ideal world.

C.4.1 Description of the Simulator

$\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with $\mathcal{Z}$ and the parties in the network. $\mathcal{S}$ forwards all messages to/from $\mathcal{A}$ and $\mathcal{Z}$ in the usual way.

Generating the public parameters. $\mathcal{S}$ generates pk_1 and pk_2 along with their corresponding secret keys sk_1 and sk_2. It also runs $(\mathsf{crs}, \tau) \leftarrow \mathcal{S}_1(1^k)$, where $\mathcal{S}_1$ is the initial simulator for the simulation-sound NIZK proof system. The public parameters $(pk_1, pk_2, \mathsf{crs})$ are given to $\mathcal{A}$, and then $\mathcal{S}$ responds to the messages of $\mathcal{A}$ as described below.

Receiving a $(\mathsf{NewSession}, \mathsf{sid}, P_i, P_j)$ **message from** $\hat{\mathcal{F}}_{\mathsf{pwKE}}$**.** Upon receiving such a message (indicating that P_i should initiate the protocol with P_j), $\mathcal{S}$ proceeds as follows. Choose a random hash key $k \leftarrow K$ and compute $s := \alpha(k)$. Compute the ciphertext $C_1 \leftarrow \mathsf{Enc}_{pk_1}(0)$ and a simulated NIZK proof π for the statement $(C_1, s) \in L^*$. Set $\mathsf{label} := (\mathsf{sid}, P_i, P_j, s, C_1, \pi)$ and compute $C_2 \leftarrow \mathsf{Enc}_{pk_2}(\mathsf{label}, 0)$. Give the message (s, C_1, π, C_2) to $\mathcal{A}$.

Receiving a message $\mathsf{msg}' = (s', C_1', \pi', C_2')$ **from** $\mathcal{A}$**.** Let P_i denote the user to whom $\mathcal{A}$ sends this message, and let sid denote the session ID with which this message is associated. Let P_j denote the partner of P_i for this session.

If msg' is invalid then $\mathcal{S}$ does nothing. Otherwise, we say msg' is *previously used* if it was sent by $\mathcal{S}$ (on behalf of P_j) upon receiving the message $(\mathsf{NewSession}, \mathsf{sid}, P_j, P_i)$ from $\hat{\mathcal{F}}_{\mathsf{pwKE}}$. In any other case we say msg' is *adversarially generated*. To respond to this message, $\mathcal{S}$ does:

1. If msg' is previously used, then $\mathcal{S}$ sends $(\mathsf{NewKey}, \mathsf{sid}, P_i, \perp)$ to the functionality $\hat{\mathcal{F}}_{\mathsf{pwKE}}$. (Note this has the effect of choosing a random session key for this instance of P_i if it terminates before the partnered session at P_j, and otherwise sends to P_i the same session key already computed for the partnered session at P_j; cf. Figure 3 in Appendix A.3.1.)

2. If msg' is adversarially generated, then $\mathcal{S}$ decrypts the ciphertext C_2' using the secret key sk_2 to obtain a password pw'. Then $\mathcal{S}$ queries the functionality $\hat{\mathcal{F}}_{\mathsf{pwKE}}$ on input $(\mathsf{TestPwd}, \mathsf{sid}, P_i, pw')$, which replies with either "correct guess" or "wrong guess".

   (a) If the reply is "correct guess", then $\mathcal{S}$ decrypts C_1' using sk_1 to obtain k'. It then computes $\mathsf{sk} := H_k(\mathsf{label}', C_2', pw') \cdot H_{k'}(\mathsf{label}, C_2, pw')$, where $H_{k'}(C_2)$ is computed using k'. Finally, $\mathcal{S}$ sends $(\mathsf{NewKey}, \mathsf{sid}, P_i, \mathsf{sk})$ to $\hat{\mathcal{F}}_{\mathsf{pwKE}}$.

   (b) If the reply is "wrong guess", then $\mathcal{S}$ sends $(\mathsf{NewKey}, \mathsf{sid}, P_i, \perp)$ to $\hat{\mathcal{F}}_{\mathsf{pwKE}}$.

C.4.2 Proof of Indistinguishability

Let $\text{IDEAL}_{\hat{\mathcal{F}}_{\mathsf{pwKE}}, \mathcal{S}, \mathcal{Z}}$ denote the view of the environment $\mathcal{Z}$ in the ideal world when interacting with $\mathcal{S}$, and let $\text{EXEC}_{P, \mathcal{A}, \mathcal{Z}}$ denote the view of $\mathcal{Z}$ in the real world when the protocol from Figure 2 is being run. Our aim is to show that these distributions are computationally indistinguishable. We do this by considering a sequence of experiments $P_1, \ldots, P_5$ (where P_0 corresponds to the real-world execution). Let $\text{EXEC}_{P_i, \mathcal{A}, \mathcal{Z}}$ denote the view of $\mathcal{Z}$ in P_i. We show that $\text{EXEC}_{P_i, \mathcal{A}, \mathcal{Z}}$ is computationally indistinguishable from $\text{EXEC}_{P_{i+1}, \mathcal{A}, \mathcal{Z}}$ for all i, and then argue that $\text{EXEC}_{P_5, \mathcal{A}, \mathcal{Z}}$ is identical to $\text{IDEAL}_{\hat{\mathcal{F}}_{\mathsf{pwKE}}, \mathcal{S}, \mathcal{Z}}$. This completes the proof.

Experiment P_0: Recall, experiment P_0 involves the environment $\mathcal{Z}$ interacting with the adversary $\mathcal{A}$, who in turn interacts with parties running the real protocol as specified in Figure 2. The view of $\mathcal{Z}$ consists of public parameters and all the protocol messages (forwarded to it by $\mathcal{A}$) as well as all the session keys produced by parties during the course of the experiment.

Experiment P_1: We change the distribution of the public parameters and the messages generated by the parties in the protocol. Specifically, the common random string crs is replaced with a simulated one, and the proof π in every outgoing message is replaced with a simulated proof. It follows from the zero-knowledge properties of the proof system that $\text{EXEC}_{P_1, \mathcal{A}, \mathcal{Z}} \approx_c \text{EXEC}_{P_0, \mathcal{A}, \mathcal{Z}}$.

Experiment P_2: Here, we again change the distribution of the outgoing messages by always computing the ciphertext C_1 as an encryption of 0 (rather than an encryption of the hash key k). The proof of the following claim is immediate:

Claim 6 *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a CPA-secure encryption scheme, then* $\text{EXEC}_{P_2,\mathcal{A},\mathcal{Z}} \approx_c \text{EXEC}_{P_1,\mathcal{A},\mathcal{Z}}$.

Before continuing, we define the notion of a *previously used* message. Note that the definition here is slightly different from the definition used in the proof of Theorem 1.

Consider a message $\mathsf{msg}' = (s', C_1', \pi', C_2')$ sent to a user P_i and associated with session ID sid. Let P_j denote the partner of P_i in this session. We say msg' is previously used if it was sent by P_j for the same session ID sid, where P_j is partnered with P_i in that session. In any other case, we say msg' is *adversarially generated.*

Experiment P_3: We change the way session keys are computed. Specifically, consider an instance of user P_i, with session ID sid and partner P_j, who receives an incoming message $\mathsf{msg}' = (s', C_1', \pi', C_2')$. If this message is invalid, then it is handled as before (and no session key is computed). Otherwise, set $\mathsf{label}' := (\mathsf{sid}, P_j, P_i, s', C_1', \pi')$ and then:

- If msg' is adversarially generated, compute $pw' := \mathsf{Dec}_{sk_2}(\mathsf{label}', C_2')$. Let pw be the value of the password being used by the current instance.
  1. If $pw' = pw$, compute $k' := \mathsf{Dec}_{sk_1}(C_1)$. Then compute $\mathsf{sk} := H_k(\mathsf{label}', C_2', pw) \cdot H_{k'}(\mathsf{label}, C_2, pw)$, where the hash values are computed using the known keys k and k'.
  2. If $pw' \neq pw$, choose sk uniformly from G.
- If msg is previously used, then in particular the simulator knows a value k' such that $s' = \alpha(k')$. The simulator computes $\mathsf{sk} := H_k(\mathsf{label}', C_2', pw) \cdot H_{k'}(\mathsf{label}, C_2, pw)$, but using k' to compute the second hash value (rather than using the randomness used to generate C_2, as done in P_2).

Claim 7 *If* $(\mathsf{CRSGen}, \mathcal{P}, \mathcal{V})$ *is simulation-sound,* $\text{EXEC}_{P_3,\mathcal{A},\mathcal{Z}} \approx_s \text{EXEC}_{P_2,\mathcal{A},\mathcal{Z}}$.

Proof The only difference between the proof here and the proof of Claim 3 is with regard to what happens when msg' is adversarially generated and $pw' = pw$, so we focus on that case. Since the proof system is simulation-sound, with all but negligible probability $\mathcal{S}$ extracts a value k' such that $\alpha(k') = s'$. Assuming this occurs, the session key computed in P_3 is identical to the session key that would be computed in P_2. ■

Experiment P_4: Here, in every outgoing message the ciphertext C_2 is generated as an encryption of 0, rather than as an encryption of pw. It follows readily from the CCA-security of the encryption scheme used that $\text{EXEC}_{P_4,\mathcal{A},\mathcal{Z}} \approx_c \text{EXEC}_{P_3,\mathcal{A},\mathcal{Z}}$.

Experiment P_5: We once again change the secret key computed on input a message $\mathsf{msg}' = (s', C_1', \pi', C_2')$ that is previously used. In this case, let $\mathsf{pid}_U^i = U'$ and proceed as follows:

- If there exists an instance $\Pi_{U'}^j$ partnered with Π_U^i, then set $\mathsf{sk}_U^i := \mathsf{sk}_{U'}^j$.
- Otherwise, choose sk_U^i uniformly from G.

Claim 8 $\text{EXEC}_{P_5,\mathcal{A},\mathcal{Z}} \approx_s \text{EXEC}_{P_4,\mathcal{A},\mathcal{Z}}$.

Proof Since msg' is previously used, the ciphertext C_2' is an encryption of 0. Thus $(\mathsf{label}', C_2', pw)$ is *not* in L, and it follows (cf. Equation (2)) that $(s, H_k(\mathsf{label}', C_2', pw))$ is statistically close to (s, g), where $s = \alpha(k)$ and g is a uniform element in G. This means that the secret key is statistically close to a uniform element in G, even conditioned on the given transcript. The claim follows. ■

Finally, we claim that the distribution $\textsc{exec}_{P_5,\mathcal{A},\mathcal{Z}}$ is identical to the distribution produced by the simulator, namely $\textsc{ideal}_{\hat{\mathcal{F}}_{\mathsf{pwKE}},\mathcal{S},\mathcal{Z}}$. This follows by inspection. The only change is a syntactic one: in experiment P_5 the secret keys are computed and stored locally, whereas in the ideal world the simulator computes the secret keys and sends them to the functionality, which in turn forwards them to the (dummy) parties.

The above shows that $\textsc{exec}_{P,\mathcal{A},\mathcal{Z}} \equiv \textsc{exec}_{P_0,\mathcal{A},\mathcal{Z}} \approx_c \textsc{exec}_{P_5,\mathcal{A},\mathcal{Z}} \equiv \textsc{ideal}_{\hat{\mathcal{F}}_{\mathsf{pwKE}},\mathcal{S},\mathcal{Z}}$, completing the proof of the theorem.