J. Cryptol. (2013) 26: 714–743
DOI: 10.1007/s00145-012-9133-6

Journal of
CRYPTOLOGY

Round-Optimal Password-Based Authenticated Key Exchange

Jonathan Katz*
Dept. of Computer Science, University of Maryland, College Park, USA
jkatz@cs.umd.edu

Vinod Vaikuntanathan†
Dept. of Computer Science, University of Toronto, Toronto, Canada
vinodv@cs.toronto.edu

Communicated by Kiltz

Received 6 January 2012
Online publication 4 December 2012

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-based cryptosystems, the resulting protocol is also (reasonably) efficient. Somewhat surprisingly, our framework can be adapted to give protocols in the standard model that are *universally composable* while still using only one (simultaneous) round.

Key words. Password-based key exchange, Round complexity.

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 [19] (which protects only against an eavesdropping adversary and provides no authentication at all) are known; see, e.g., [4,7].

[*] Work done while the first author visiting IBM.

[†] Work done while the second author at IBM and Microsoft Research.

Password-based protocols allow users to bootstrap a 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) [11]. 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 in which an adversary enumerates passwords from the dictionary of potential passwords, and tries to match observed protocol transcripts to each one, are of no use.

Early work on password-based protocols [24,29] 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 [6] proposed the first protocols for password-based authenticated key exchange (PAKE) with heuristic arguments for their security. Several years later, provably secure PAKE protocols were constructed [3,12,36] in the random-oracle/ideal-cipher models, and many improvements and generalizations of these protocols are known. In contrast, only a handful of PAKE protocols are known in the standard model (i.e., without random oracles):

- *General assumptions:* Goldreich and Lindell [23] gave the first PAKE protocol in the standard model (and without requiring any additional setup). Their work does not handle concurrent executions of the protocol by the same party. Work of Barak et al. [2] shows a general feasibility result for computation over unauthenticated networks which implies a solution for PAKE as a special case, assuming a common reference string (CRS) is available to the parties.[2] All these protocols are impractical in terms of communication, computation, and round complexity. Nguyen and Vadhan [38] showed some efficiency improvements, but achieved a weaker notion of security. Their protocol is still far from practical.
- *Efficient protocols:* Katz, Ostrovsky, and Yung [34] demonstrated the first *efficient* PAKE protocol with a proof of security based on standard assumptions; extensions and improvements of their protocol were given in Refs. [16,21,22,33,35]. Different constructions of efficient PAKE protocols were given in Refs. [27,32]. These works all require a CRS.

Other relevant work, done subsequent to the present work, includes Refs. [15,25,26].

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 cannot depend on the other party's message. We stress, however,

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

[2] Reliance on a CRS is not a serious drawback in the context of PAKE where the CRS can be hard-coded into an implementation of the protocol. Note also that reliance on a CRS (or some other setup) is inherent for achieving *universally composable* PAKE [16].

that the adversary is assumed to be *rushing*; i.e., it may wait to receive an honest 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 [19], 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 [31,39]. One-round PAKE protocols are also known (e.g., [3]) in the random oracle model. All prior PAKE protocols based on standard assumptions, though, require three or more rounds. We remark that the protocols in Refs. [27,32] achieve *explicit* authentication in three rounds (whereas the protocols of Refs. [21,22,34,35] achieve only *implicit* authentication in three rounds, and require an additional round for explicit authentication), but the round complexity of these protocols cannot be further reduced even if only implicit authentication is desired.

1.1. *Our Results*

We show a new framework for constructing *one-round* PAKE 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; explicit authentication is impossible in one round without stronger setup assumptions (e.g., a global clock).

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

Somewhat surprisingly, we can extend our framework to give a universally composable PAKE protocol [14]—secure against static corruptions—*without increasing the round complexity at all* (and still without relying on random oracles). In contrast, the work of Canetti et al. [16] shows a method for obtaining universal composability (used also in Ref. [27]) that requires additional messages/rounds. Abdalla et al. [1] show a universally composable PAKE protocol, proven secure in the random oracle model, that requires three rounds. To the best of our knowledge, no prior universally composable PAKE 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 [17], as used for PAKE in Ref. [22] (and implicitly in Ref. [34]). Roughly speaking, and adapted to the present context, a smooth projective hash function is a keyed function H that can be computed in two ways: either using the hash key k or using a *projected key* s. If C is an encryption of some value pw using randomness r (with respect to some fixed public key), then $H_k(C, pw) = H_s(C, pw, r)$. On the other hand, if C is not an encryption of pw then the value of $H_k(C, pw)$ is independent of the projected key s. We refer the reader to Sect. 2.2 for formal definitions.

The basic structure of previous protocols [22,34], omitting various 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 $s' = \alpha(k', C, pw)$ corresponding to a hash key k'.
***Third round*:** The client sends a projected key $s = \alpha(k, C', pw)$ corresponding to a hash key k.

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 PAKE 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. The main technical difficulty is that an honestly generated client message (s, C) might be forwarded by an adversary to *multiple* server instances (and similarly for server messages), and the session keys computed in all these instances should 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 verification key sent in the first round [22,34] or a MAC derived from the shared session key [21]. Neither approach is viable if we want the entire protocol to take place in a single round.)

  To address the above difficulty, we rely on a technical lemma (which may be of independent interest) regarding the re-use of both the hash keys and the inputs to the smooth projective hash function.

Additional ideas are needed to obtain a *universally composable* protocol without increasing the number of rounds. We refer the reader to Sect. 5.1 for an overview of the techniques used there.

1.3. *Outline of the Paper*

In Sect. 2 we present a standard definition of security for PAKE due to Bellare et al. [3]. We also review there the notion of smooth projective hashing, and prove a technical lemma regarding its usage. In Sect. 3 we describe our basic framework for constructing one-round PAKE protocols, and prove security of this approach according to the definition of Bellare et al. [3]. We discuss in Sect. 4 two instantiations of our framework: one based on the decisional Diffie–Hellman assumption, and a second, more efficient instantiation, based on bilinear maps. In Sect. 5 we describe an extension of our framework that yields one-round, universally composable PAKE protocols.

2. Definitions and Background

Throughout, we denote the security parameter by n.

2.1. *Password-Based Authenticated Key Exchange*

We present a definition of security for PAKE due to Bellare, Pointcheval, and Rogaway [3], based on prior work of Bellare and Rogaway [4,5]. The text here is taken almost verbatim from Katz et al. [34].

Participants, passwords, and initialization Prior to any execution of the protocol there is an initialization phase during which public parameters and a CRS are established. We assume a fixed set User of protocol participants (also called principals or users). For every distinct $U, U' \in \mathsf{User}$, users U and U' share a password $pw_{U,U'}$. We assume that each $pw_{U,U'}$ is chosen independently and uniformly 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 implicitly consider arbitrary password distributions in 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 is associated with the following variables:

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

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

- $\mathsf{Send}(U, i, \mathsf{msg})$—This sends message msg to instance Π_U^i. This instance runs according to the protocol specification, updating state as appropriate. The message output by Π_U^i is given to the adversary.

The adversary can prompt the unused instance Π_U^i to initiate the protocol with partner U' by querying $\mathsf{Send}(U, i, U')$. In response, 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 $\mathcal{A}$ 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 $\mathsf{Adv}_{\mathcal{A},\Pi}(n) \le Q(n)/D + \mathsf{negl}(n)$.

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 [17], and our discussion here is based on that of Gennaro and Lindell [22]. Rather than aiming for utmost generality, we tailor the definitions to our application.

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

1. $X \overset{\text{def}}{=} \{(\mathsf{label}, C, pw)\}$, where $(\mathsf{label}, C) \in C_{pk}$ and $pw \in \{1, \ldots, D\}$.
2. $L_{pw} \overset{\text{def}}{=} \{(\mathsf{label}, \mathsf{Enc}_{pk}(\mathsf{label}, pw), pw)\}$, where $\mathsf{label} \in \{0, 1\}^*$.

That is, X consists of all tuples of valid labels, valid ciphertexts, and passwords, while L_{pw} consists of all tuples where the third component is pw, the first component is label, and the second component is an encryption of pw using label. Let $L = \bigcup_{pw=1}^{D} L_{pw}$, and note that $L \subset X$. It follows from CCA security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ that the following is negligible for any probabilistic polynomial-time $\mathcal{A}$:

$$\left| \Pr\left[\begin{array}{c} (pk, sk) \leftarrow \mathsf{Gen}(1^n); \\ (\mathsf{label}, pw) \leftarrow \mathcal{A}^{\mathsf{Dec}_{sk}(\cdot,\cdot)}(pk); \\ C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw) \end{array} : \mathcal{A}^{\mathsf{Dec}_{sk}(\cdot,\cdot)}(C) = 1 \right] \right.$$
$$\left. - \Pr\left[\begin{array}{c} (pk, sk) \leftarrow \mathsf{Gen}(1^n); \\ (\mathsf{label}, pw) \leftarrow \mathcal{A}^{\mathsf{Dec}_{sk}(\cdot,\cdot)}(pk); \\ C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, 0) \end{array} : \mathcal{A}^{\mathsf{Dec}_{sk}(\cdot,\cdot)}(C) = 1 \right] \right|,$$

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

Smooth projective hash functions Fix pk and sets X, L 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 $\mathbb{G}$, 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 $(K, \mathbb{G}, \mathcal{H} = \{H_k : X \to \mathbb{G}\}_{k\in K}, S, \alpha : K \to S)$ such that:

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

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:

$$\bigl\{k \leftarrow K; s := \alpha(k) : \bigl(s, H_k\bigl(f(s)\bigr)\bigr)\bigr\} \quad \text{and} \quad \bigl\{k \leftarrow K; s := \alpha(k); g \leftarrow \mathbb{G} : (s, g)\bigr\}. \tag{1}$$

Above we have modified the definition from Gennaro and Lindell [22] in two ways: first, α is *non-adaptive* and depends on k only (rather than both k and x); second, we require Eq. (1) to hold even for *adaptive* choice of $f(s) \notin L$. The first modification allows us to achieve the functionality of password-based authenticated key exchange in one round, whereas the second is used for proving security of the resulting protocol.

A technical lemma We now prove a technical lemma regarding smooth projective hash functions. Somewhat informally, Gennaro and Lindell [22] showed that, for randomly generated pk and any (label, pw), the distribution

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

is computationally indistinguishable from the distribution

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

(Note that 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

$$\left\{\begin{matrix} k_1, \dots, k_\ell \leftarrow K; \forall i : s_i := \alpha(k_i); \\ C_1, \dots, C_\ell \leftarrow \mathsf{Enc}_{pk}(pw) \end{matrix} : \bigl(\{s_i\}, \{C_i\}, \bigl\{H_{k_i}(C_j, pw)\bigr\}_{i,j=1}^{\ell}\bigr)\right\}$$

is computationally indistinguishable from the distribution

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

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 $(K, \mathbb{G}, \mathcal{H} = \{H_k : X \to \mathbb{G}\}_{k \in K}, S, \alpha : K \to S)$ be a smooth projective hash function for pk. Give pk to $\mathcal{A}$.
2. Sample $k_1, \dots, k_\ell \leftarrow K$, and let $s_i := \alpha(k_i)$ for all i. Give $s_1, \dots, s_\ell$ to $\mathcal{A}$.

3. $\mathcal{A}$ may adaptively query a (modified) encryption oracle that takes as input (label, pw) for $pw \in [D]$, 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 $g_1, \ldots, g_\ell \leftarrow \mathbb{G}$.
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 $b' = b$.

Lemma 1. *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a CCA-secure labeled encryption scheme, and* $(K, \mathbb{G}, \mathcal{H} = \{H_k : X \to \mathbb{G}\}_{k \in K}, S, \alpha : K \to S)$ *a smooth projective hash function. For any polynomial* ℓ *and probabilistic polynomial-time* $\mathcal{A}$*, we have* $\Pr[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(n)$.

Proof. Let ℓ' be a (polynomial) bound on the number of encryption queries asked by $\mathcal{A}$, and let $C_i \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}_i, pw_i)$ be the ciphertext returned in response to the ith query of $\mathcal{A}$. 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 $\mathbb{G}$.

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

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

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

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

where the $g_{a,b}$ denote uniform and independent elements of $\mathbb{G}$. (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. Note that only the distribution of the (i, j)th element has changed.) 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., $C_j \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}_j, 0)$). It follows 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 $h_{i,j} \in \mathbb{G}$:

1. Choose random $k_i \leftarrow K$ and give $s_i = \alpha(k_i)$ to B.
2. B computes $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and internally runs $\mathcal{A}$ on initial input pk. 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), then B does:
   (a) For the kth such query where $k < j$, 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 $g_{1,k}, \ldots, g_{\ell,k} \leftarrow \mathbb{G}$.
   (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 $(\mathsf{label}_j, C_j, pw_j)$, and receives $h_{i,j}$. It then chooses $g_{1,j}, \ldots, g_{i-1,j} \leftarrow \mathbb{G}$, 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 where $k > j$, 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.

   B answers any decryption queries of $\mathcal{A}$ using sk.

Note that $pw_j \neq 0$ (since $0 \notin [D]$), and so $(\mathsf{label}_j, C_j, pw_j) \in X \setminus L$. The view of $\mathcal{A}$ is distributed according to $\mathsf{real}'_{i,j}$ if $h_{i,j} = H_{k_i}(\mathsf{label}_j, C_j, pw_j)$, and according to $\mathsf{rand}'_{i,j}$ if $h_{i,j}$ is chosen uniformly from $\mathbb{G}$. It follows from Equation (1) that $\mathsf{real}'_{i,j}$ and $\mathsf{rand}'_{i,j}$ are statistically close. □

3. A Framework for One-Round PAKE Protocols

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 Sect. 2.2.

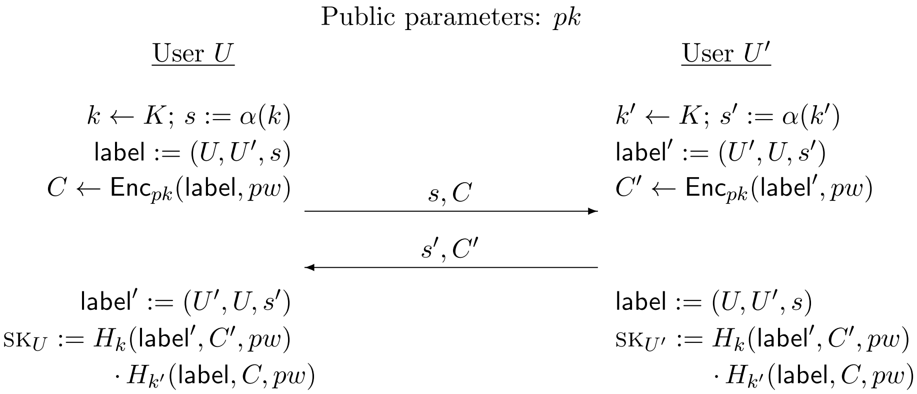
Public parameters: pk
User U
User U'
$k \leftarrow K$; $s := \alpha(k)$
$\mathsf{label} := (U, U', s)$
$C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, pw)$
s, C
$k' \leftarrow K$; $s' := \alpha(k')$
$\mathsf{label}' := (U', U, s')$
$C' \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}', pw)$
s', C'
$\mathsf{label}' := (U', U, s')$
$\mathrm{SK}_U := H_k(\mathsf{label}', C', pw) \cdot H_{k'}(\mathsf{label}, C, pw)$
$\mathsf{label} := (U, U', s)$
$\mathrm{SK}_{U'} := H_k(\mathsf{label}', C', pw) \cdot H_{k'}(\mathsf{label}, C, pw)$

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

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 Sect. 4, a public key can be derived from a common random string.) Let $(K, \mathbb{G}, \mathcal{H} = \{H_k : X \rightarrow \mathbb{G}\}_{k \in K}, S, \alpha : K \rightarrow S)$ 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 Fig. 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

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

U computes $H_k(\mathsf{label}', C', pw)$ using 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.

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

Proof. Let Π denote the protocol in Fig. 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 $\{1, \dots, D\}$, and so 0 is never a valid password.) The (common) session key is computed as

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

where both values are computed using the (known) keys k, k'. A proof of the following is immediate from semantic security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

Claim 1. $|\mathsf{Adv}_{\mathcal{A},0}(n) - \mathsf{Adv}_{\mathcal{A},1}(n)|$ *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 $\mathbb{G}$.

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

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

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. Eq. (1)) that $(s, H_k(\mathsf{label}', C', pw))$ is statistically close to (s, g), where g is uniform in $\mathbb{G}$. This means that, in Expt_1, $\mathsf{sk}_U = \mathsf{sk}_{U'}$ is statistically close to uniform in $\mathbb{G}$, 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, $\mathsf{Send}_1(U, i, \mathsf{msg})$, represents $\mathcal{A}$ sending the message msg to instance Π_U^i. In response, a session key sk_U^i is computed. (Nothing is output in response to this query, but the value of the computed session key affects a subsequent Reveal or Test query for instance Π_U^i.) For a query $\mathsf{Send}_1(U, i, \mathsf{msg})$ with $\mathsf{pid}_U^i = U'$, we say a valid 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 a valid msg is *adversarially generated.* (An invalid message is always ignored by the instance that receives it, and so we assume from now on that $\mathcal{A}$ does not send such messages.)

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 $\mathsf{pid}_U^i = U'$, let $\mathsf{label}' = (U', U, s')$, and let $pw = pw_{U,U'}$):

1. If msg is adversarially generated, then compute $pw' := \mathsf{Dec}_{sk}(\mathsf{label}', C')$. Then:
   (a) If $pw' = pw$, the simulator declares that $\mathcal{A}$ succeeds and terminates the experiment.
   (b) If $pw' \neq pw$, the simulator chooses sk_U^i uniformly from $\mathbb{G}$.
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 1(a) can only increase the advantage of $\mathcal{A}$. The change in Case 1(b) introduces a negligible statistical difference; the analysis is as in Claim 2, except that we now specifically use the fact that Equation (1) holds even under adaptive choice of $(\mathsf{label}', C', pw) \notin L$. The change in Case 2 does not affect the computed value sk_U^i since $(\mathsf{label}, C, pw) \in L$. □

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 $\mathbb{G}$.

Claim 4. $|\mathsf{Adv}_{\mathcal{A},3}(n) - \mathsf{Adv}_{\mathcal{A},4}(n)|$ *is negligible.*

Proof. The proof relies on Lemma 1 (cf. Sect. 2.2) and CCA-security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. 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. $\mathcal{S}$ 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 $h_{1,i},\dots,h_{\ell,i}$. 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 $\mathsf{pid}^j_U = U'$ and $\mathsf{label}' = (U', U, s')$, 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 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 $\mathsf{sk}^j_U := h_{i,r} \cdot h_{r,i}$.
   (b) 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 $\mathbb{G}$. 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.

Let b be as in the experiment defined in Sect. 2.2. 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 $b = 1$ 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 $\mathbb{G}$. 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 time the value $h_{i,r}$ could be used in the experiment is if $\mathcal{A}$ queries $\mathsf{Send}_1(U', \star, (s_i, C_i))$ to the instance $\Pi^\star_{U'}$ which sent (s_r, C_r). But then $\Pi^\star_{U'}$ and Π^j_U are partnered, and so the session key $\mathsf{sk}^\star_{U'}$ will be set equal to sk^j_U (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 independently uniform in $\mathbb{G}$.

The claim 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. $|\mathsf{Adv}_{\mathcal{A},4}(n) - \mathsf{Adv}_{\mathcal{A},5}(n)|$ *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). 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. □

4. Instantiating the Building Blocks

We now discuss two possible instantiations of the building blocks required by the protocol of Sect. 3.

- Our first instantiation is based on the decisional Diffie–Hellman (DDH) assumption and (generic) simulation-sound NIZK proofs [40]; see Sect. A.2 for a definition. (This instantiation could also be based on the quadratic residuosity assumption or the Paillier assumption, as in Ref. [22]. We omit further details.)
- Our second, more efficient instantiation is based on the decisional linear assumption [10] in groups with a bilinear map, and a specific simulation-sound NIZK proof system.

4.1. *A Construction Based on the DDH Assumption*

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

A CCA-secure encryption scheme We construct a CCA-secure encryption scheme by applying the Naor–Yung/Sahai paradigm [37,40] 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 [40].

Fixing $\mathbb{G}$, let $\mathsf{ElGamal}_{g,h}(m)$ denote an El Gamal encryption of $m \in \mathbb{G}$ with respect to (g, h); namely, $\mathsf{ElGamal}_{g,h}(m)$ outputs $(g^r, h^r \cdot m)$, where $r \in \mathbb{Z}_p$ is chosen uniformly at random. (We assume passwords can be represented as elements of $\mathbb{G}$.) To encrypt a message $m \in \mathbb{G}$ in our CCA-secure scheme, the sender outputs the ciphertext

$$\big(\mathsf{ElGamal}_{g_1,h_1}(m), \mathsf{ElGamal}_{g_2,h_2}(m), \pi\big),$$

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 Sahai [40] 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 a group $\mathbb{G}$ and a public key $pk = (g_1, h_1, g_2, h_2, \mathsf{crs})$ as above, and define sets X and $\{L_{pw}\}$ as in Sect. 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

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

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

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

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 $(X, \{L_{pw}\})$.*

Proof. 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

$$C = (c_1, d_1, c_2, d_2, \pi) = \left(g_1^{r_1},\ h_1^{r_1} \cdot pw,\ g_2^{r_2},\ h_2^{r_2} \cdot pw,\ \pi\right)$$

for some $r_1, r_2 \in \mathbb{Z}_p$. For $k = (x_1, y_1, x_2, y_2)$ we have

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

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

$$x_1 + (\log_{g_1} h_1) \cdot y_1 = \log_{g_1} s_1, \tag{2}$$

$$x_2 + (\log_{g_2} h_2) \cdot y_2 = \log_{g_2} s_2, \tag{3}$$

where the above are modulo the group order p. For any $(\mathsf{label}, C, pw) \in X \setminus L$, we can write $C = (g_1^{r_1}, h_1^{r_1} \cdot pw', g_2^{r_2}, h_2^{r_2} \cdot pw', \pi)$ 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:

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

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

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

Since this equation in the unknowns x_1, y_1, x_2, y_2 is linearly independent of Equations (2) and (3), 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 $\mathbb{G}$. □

4.2. *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, constructed using techniques adapted from Refs. [13,28]. Our construction here relies on the *decisional linear assumption* as introduced by Boneh et al. [10]; 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. [10], 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 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 $m \in \mathbb{G}$ is encrypted by choosing random $r, s \in \mathbb{Z}_p$ and computing the ciphertext $(f^r, g^s, h^{r+s} \cdot m)$. Given a ciphertext (c_1, c_2, c_3), we can recover 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 Refs. [13,28]. 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

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

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 Refs. [37,40] that this yields a CCA-secure scheme under the decisional linear assumption. Ciphertexts consist of 66 group elements altogether (see Appendix B).

A smooth projective hash function (Similar constructions were given in Refs. [30,41].) 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 $\{L_{pw}\}$ as in Sect. 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

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

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

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

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 $(X, \{L_{pw}\})$.*

Proof. 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

$$C = (c_1, d_1, e_1, c_2, d_2, e_2, \pi) = \left(f_1^{r_1},\ g_1^{s_1},\ h^{r_1+s_1} \cdot pw,\ f_2^{r_2},\ g_2^{s_2},\ h^{r_2+s_2} \cdot pw,\ \pi\right)$$

for some $r_1, s_1, r_2, s_2 \in \mathbb{Z}_p$. For a hash key $k = (x_1, y_1, z_1, x_2, y_2, z_2)$ we have

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

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

$$\begin{pmatrix} \alpha_1 & 0 & 1 & 0 & 0 & 0 \\ 0 & \beta_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_2 & 0 & 1 \\ 0 & 0 & 0 & 0 & \beta_2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix}, \tag{4}$$

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

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

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. The proof holds even when this is not the case.) We then have

$$\begin{aligned} & H_k(\mathsf{label}, C, pw) \\ &= f_1^{r_1 x_1} \cdot g_1^{s_1 y_1} \cdot h^{(r_1+s_1) z_1} \cdot \left(pw'/pw\right)^{z_1} \cdot f_2^{r_2 x_2} \cdot g_2^{s_2 y_2} \cdot h^{(r_2+s_2) z_2} \cdot \left(pw'/pw\right)^{z_2} \\ &= S_1^{r_1} S_2^{s_1} S_3^{r_2} S_4^{s_2} \cdot \left(pw'/pw\right)^{z_1+z_2}, \end{aligned} \tag{5}$$

where $pw'/pw \neq 1$. For any $g \in \mathbb{G}$, we have $(pw'/pw)^{z_1+z_2} = g$ iff

$$\log_h\bigl(pw'/pw\bigr) \cdot z_1 + \log_h\bigl(pw'/pw\bigr) \cdot z_2 = \log_h g. \tag{6}$$

Since Eq. (6) is linearly independent of the system of equations given by (4), the probability that $(pw'/pw)^{z_1+z_2} = g$ is exactly $1/|\mathbb{G}|$ even conditioned on the value $\alpha(k)$. Looking at Eq. (5), 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 $\mathbb{G}$. □

5. A One-Round, Universally Composable PAKE Protocol

Canetti et al. [16] gave a definition of security for password-based authenticated key exchange in the universal composability (UC) framework [14]. Their definition guarantees a strong, simulation-based notion of security that, in particular, guarantees that security is maintained even when the protocol is run concurrently with arbitrary other protocols. 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 Canetti et al. [16] for a more complete discussion.

A brief review of the UC framework and the password-based key-exchange functionality $\mathcal{F}_{\mathsf{pwKE}}$ are given in Appendix C. We let $\hat{\mathcal{F}}_{\mathsf{pwKE}}$ denote the multi-session extension of $\mathcal{F}_{\mathsf{pwKE}}$.

5.1. *Overview of the Construction*

We do not know how to prove that the protocol from Sect. 3 is universally composable. The main difficulty is that the definition of PAKE in the UC framework requires simulation *even if the adversary guesses the correct password*. (In contrast, in the proof of security in Sect. 3 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 prior work [16], the issue above was resolved by having one party send a "precommitment" to the password, and then run a regular PAKE protocol and give 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 Fig. 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

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

$$\begin{array}{rcl}
\underline{\text{User } U} & & \underline{\text{User } U'} \\
k \leftarrow K;\ s := \alpha(k) & & k' \leftarrow K;\ s' := \alpha(k') \\
C_1 \leftarrow \mathsf{Enc}_{pk_1}(k) & & C_1' \leftarrow \mathsf{Enc}_{pk_1}(k') \\
\pi := \mathcal{P}_{\mathsf{crs}}((s, C_1) \in L^*) & & \pi' := \mathcal{P}_{\mathsf{crs}}((s', C_1') \in L^*) \\
\mathsf{label} := (\mathsf{sid}, U, U', s, C_1, \pi) & & \mathsf{label}' := (\mathsf{sid}, U', U, s', C_1', \pi') \\
C_2 \leftarrow \mathsf{Enc}_{pk_2}(\mathsf{label}, pw) & \xrightarrow{\quad s, C_1, \pi, C_2 \quad} & C_2' \leftarrow \mathsf{Enc}_{pk_2}(\mathsf{label}', pw) \\
 & \xleftarrow{\quad s', C_1', \pi', C_2' \quad} & \\
\mathsf{label}' := (\mathsf{sid}, U', U, s', C_1', \pi') & & \mathsf{label} := (\mathsf{sid}, U, U', s, C_1, \pi) \\
\mathsf{sk}_U := H_k(\mathsf{label}', C_2', pw) & & \mathsf{sk}_{U'} := H_k(\mathsf{label}', C_2', pw) \\
\cdot H_{k'}(\mathsf{label}, C_2, pw) & & \cdot H_{k'}(\mathsf{label}, C_2, pw)
\end{array}$$

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

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.

5.2. *Description of the Protocol*

In addition to the building blocks used in Sect. 3, here we also rely on an unbounded simulation-sound [18] NIZK proof system (CRSGen, $\mathcal{P}$, $\mathcal{V}$) for a language L^* defined below; refer to Sect. A.2 for definitions.

Public parameters The public parameters consist of two public keys pk_1, pk_2 generated by $\mathsf{Gen}(1^n)$ and a common random string crs for the simulation-sound NIZK proof system. Let $(K, \mathbb{G}, \mathcal{H} = \{H_k\}_{k \in K}, S, \alpha : K \to S)$ 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 sid. (The sid is an artifact of the UC framework, and it is guaranteed that (1) parties communicating with each other begin holding matching sids, and (2) each sid is used only once. Existence of these sids 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 Fig. 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:

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

U computes an NIZK proof π that $(C_1, s) \in L^*$, using crs. It then sets $\mathsf{label} := (\mathsf{sid}, U, U', s, C_1, \pi)$ 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 $\mathsf{label}' := (\mathsf{sid}, U', U, s', C_1', \pi')$ and computes $\mathsf{sk}_U := H_k(\mathsf{label}', C_2', pw) \cdot H_{k'}(\mathsf{label}, C_2, pw)$. Note that 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.

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

Proof. Let $\mathcal{A}$ be an adversary that interacts with the parties running the protocol. We construct an ideal-world adversary (i.e., 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.

$\mathcal{S}$ 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 (uncorrupted) 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 $\hat{\mathcal{F}}_{\mathsf{pwKE}}$. (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, or if the passwords used by these instances of P_i and P_j differ, and otherwise setting the session key for this instance of P_i equal to the session key already computed for the partnered instance of P_j; cf. Sect. C.1.1.)
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 $H_{k'}(\mathsf{label}, C_2, pw')$ 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 $\hat{\mathcal{F}}_{\mathsf{pwKE}}$.

Let $\textsc{ideal}_{\hat{\mathcal{F}}_{\mathsf{pwKE}},\mathcal{S},\mathcal{Z}}$ denote the view of the environment $\mathcal{Z}$ in the ideal world when interacting with $\mathcal{S}$, and let $\textsc{exec}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the view of $\mathcal{Z}$ in the real world when the protocol from Fig. 2 is being run. Our aim is to show that these distributions are computationally indistinguishable. We do this by considering a sequence of experiments $\mathsf{Expt}_1, \ldots, \mathsf{Expt}_5$ (where Expt_0 corresponds to the real-world execution). Let $\textsc{exec}_{i,\mathcal{A},\mathcal{Z}}$ denote the view of $\mathcal{Z}$ in Expt_i. We show that $\textsc{exec}_{i,\mathcal{A},\mathcal{Z}} \approx_c \textsc{exec}_{i+1,\mathcal{A},\mathcal{Z}}$ for all i (where this denotes computational indistinguishability), and then argue that $\textsc{exec}_{5,\mathcal{A},\mathcal{Z}}$ is identical to $\textsc{ideal}_{\hat{\mathcal{F}}_{\mathsf{pwKE}},\mathcal{S},\mathcal{Z}}$. This completes the proof.

Experiment Expt_0. Recall, this experiment involves the environment $\mathcal{Z}$ interacting with the adversary $\mathcal{A}$, who in turn interacts with parties running the real protocol as specified in Fig. 2. The view of $\mathcal{Z}$ consists of the 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 Expt_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 $\textsc{exec}_{1,\mathcal{A},\mathcal{Z}} \approx_c \textsc{exec}_{0,\mathcal{A},\mathcal{Z}}$.

Experiment Expt_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). It follows immediately from semantic security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ that $\textsc{exec}_{2,\mathcal{A},\mathcal{Z}} \approx_c \textsc{exec}_{1,\mathcal{A},\mathcal{Z}}$.

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 valid $\mathsf{msg}' = (s', C_1', \pi', C_2')$ sent to an instance of (uncorrupted user) P_i who is using session ID sid and is partnered with P_j. We say msg' is *previously used* if it was sent by an instance of the (uncorrupted user) P_j that is using the same session ID sid, and is partnered with P_i. We say a previously used msg' is *associated with* pw if pw is the password that was used by the instance of P_j that sent msg'. If msg' is not previously used, we say it is *adversarially generated*.

Experiment Expt_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 msg' 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 let pw be the password being used by the current instance of P_i. There are several sub-cases:

1. If msg' is adversarially generated, compute $pw' := \mathsf{Dec}_{sk_2}(\mathsf{label}', C_2')$. Then:
   (a) If $pw' = pw$, compute $k' := \mathsf{Dec}_{sk_1}(C_1)$. Then compute $\mathsf{sk} := H_k(\mathsf{label}', C_2', pw) \cdot H_{k'}(\mathsf{label}, C_2, pw)$, using k' to compute the second hash value.
   (a) If $pw' \neq pw$, choose sk uniformly from $\mathbb{G}$.
2. If msg' is previously used, then the simulator knows the password pw' associated with msg', and also knows a value k' such that $s' = \alpha(k')$. Then:
   (a) If $pw' = pw$, compute $\mathsf{sk} := H_k(\mathsf{label}', C_2', pw) \cdot H_{k'}(\mathsf{label}, C_2, pw)$, using k' to compute the second hash value (rather than using the randomness used to generate C_2, as in Expt_2).
   (b) If $pw' \neq pw$, choose sk uniformly from $\mathbb{G}$.

Claim 6. $\textsc{exec}_{3,\mathcal{A},\mathcal{Z}} \approx_c \textsc{exec}_{2,\mathcal{A},\mathcal{Z}}$.

Proof. This follows from simulation soundness of $(\mathsf{CRSGen}, \mathcal{P}, \mathcal{V})$. The proof is similar to that of Claim 3, though there are some differences due to the particularities of the UC setting. Imagine first that only the changes in Case 1(b) and 2(b) are made. These changes introduce a negligible statistical difference between Expt_3 and Expt_2, exactly as in Claim 3 (though note that Case 2(b) cannot occur in the context of the previous claim since parties are assumed to always use matching passwords there).

Consider next the change in Case 1(a). 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 Expt_3 is identical to the session key that would have been computed in Expt_2. The same holds (always) in Case 2(a). □

Experiment Expt_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 $\textsc{exec}_{4,\mathcal{A},\mathcal{Z}} \approx_c \textsc{exec}_{3,\mathcal{A},\mathcal{Z}}$.

Experiment Expt_5**.** We once again change the way session keys are computed. Consider an instance of user P_i, with session ID sid, partner P_j, and using password pw, who receives an incoming message $\mathsf{msg}' = (s', C_1', \pi', C_2')$ that is previously used and is associated with pw. Then:

- If the corresponding partnered instance of P_j has already computed a session key sk, then set the session key of the current instance of P_i equal to sk.
- Otherwise, choose the session key for the current instance of P_i uniformly from $\mathbb{G}$.

Claim 7. $\text{EXEC}_{5,\mathcal{A},\mathcal{Z}} \approx_c \text{EXEC}_{4,\mathcal{A},\mathcal{Z}}$.

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 (1)) 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 $\mathbb{G}$. This means that the secret key is statistically close to a uniform element in $\mathbb{G}$, even conditioned on the given transcript. The claim follows. □

The distribution $\text{EXEC}_{5,\mathcal{A},\mathcal{Z}}$ is identical to the distribution produced by the simulator, namely $\text{IDEAL}_{\hat{\mathcal{F}}_{\mathsf{pwKE}},\mathcal{S},\mathcal{Z}}$. This follows by inspection. Indeed, the only difference is a syntactic one: in experiment Expt_5 the session keys are computed and stored locally by the simulator, whereas in the ideal world the functionality computes the session keys and sends them to the (dummy) parties.

The above shows that $\text{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \equiv \text{EXEC}_{0,\mathcal{A},\mathcal{Z}} \approx_c \text{EXEC}_{5,\mathcal{A},\mathcal{Z}} \equiv \text{IDEAL}_{\hat{\mathcal{F}}_{\mathsf{pwKE}},\mathcal{S},\mathcal{Z}}$, completing the proof of the theorem. □

Appendix 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 $C \leftarrow \mathsf{Enc}_{pk}(\mathsf{label}, m)$.
- 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. Define a left-or-right encryption oracle $\mathsf{Enc}_{pk,b}(\cdot,\cdot,\cdot)$ (where $b \in \{0,1\}$) as follows:

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

Definition 3. A public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is secure against adaptive chosen-ciphertext attacks (CCA-secure) if the following is negligible for all PPT algorithms $\mathcal{A}$:

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

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

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

Simulation-sound NIZK was introduced in Refs. [18,40]. 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 [8,9,20]. 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):

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

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

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

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

$$\sigma \leftarrow \mathsf{CRSGen}\left(1^n\right)$$

$$\text{Return } \mathcal{A}^{\mathcal{P}_\sigma(\cdot,\cdot)}\left(1^n, \sigma\right)$$

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

$$(\sigma, \tau) \leftarrow \mathcal{S}_1\left(1^n\right)$$

$$\text{Return } \mathcal{A}^{\mathcal{S}'_{\sigma,\tau}(\cdot,\cdot)}\left(1^n, \sigma\right),$$

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

We next define the notion of simulation-sound NIZK. (Note that although one-time simulation soundness would suffice for our applications in Sect. 4, we only define the stronger notion of unbounded simulation-sound NIZK. See [40] 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:

$$(\sigma, \tau) \leftarrow \mathcal{S}_1(1^n)$$

$$(x, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_2(\cdot,\sigma,\tau)}(1^n, \sigma)$$

Let Q be the list of proofs returned by $\mathcal{S}_2$, above

Return 1 iff $(\pi \notin Q$ and $x \notin L$ and $\mathcal{V}_\sigma(x, \pi) = 1)$.

Assuming the existence of doubly enhanced trapdoor permutations, every language in $\mathcal{NP}$ has a simulation-sound NIZK proof system [18].

Appendix 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 Sect. 4.2. 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 $(f_1^r, g_1^s, h^{r+s} \cdot m)$. We encrypt a message m with respect to pk_2 by choosing random $r, s \in \mathbb{Z}_p$ and computing the ciphertext $(f_2^r, g_2^s, h^{r+s} \cdot m)$. We stress that the public keys use the same value h.

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:

$$F_1 = f_1^{r_1}, \tag{B.1}$$

$$G_1 = g_1^{s_1}, \tag{B.2}$$

$$F_2 = f_2^{r_2}, \tag{B.3}$$

$$G_2 = g_2^{s_2}, \tag{B.4}$$

$$H_1/H_2 = h^{r_1+s_1-r_2-s_2}. \tag{B.5}$$

As shown in Ref. [28] (see also [13, Sect. 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.[3]

Camenisch et al. [13] show a construction of an *unbounded* simulation-sound NIZK proof system. For our purposes in Sect. 3, a simpler construction that is *one-time* simulation sound [40] suffices. Let (Gen, Sign, 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 Eqs. (B.1)–(B.5), *or* that the given tuple (f, g, h, F, G, H) is an encryption of vk, i.e., that there exist r, s such that:

$$F = f^r, \quad G = g^s, \quad H/vk = h^{r+s}. \tag{B.6}$$

3. A signature σ (with respect to vk) on the proof from the previous step.

Equation (B.6) describes a system of 3 linear equations in 2 variables. Using the techniques from Ref. [13, Appendix A.2], an NIZK proof as required in step 2 can be obtained using 58 group elements. (In detail: Applying the OR-transformation from Ref. [13, Appendix A.2.2] to the two systems of equations given by Eqs. (B.1)–(B.5) and Eq. (B.6), respectively, we obtain one system of 8 linear equations and one quadratic equation in 7 variables. Applying the transformation from Ref. [13, Appendix A.2.1], we get a system of 11 linear equations in 11 variables, plus 3 extra group elements that need to be sent. From Ref. [13, Sect. 4.4] we see that an NIZK proof for the former can be done using 55 group elements.) This gives a total of 60 group elements for the entire simulation-sound NIZK proof (assuming signatures are one group element for simplicity). See also footnote 3.

Appendix C. Universally Composable Password-Based Key Exchange

C.1. *The Universal Composability Framework*

We provide a brief review of the universally composable security framework [14]. 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.

[3] Our calculations here are based on the decisional linear assumption (the 2-linear assumption in the terminology of Ref. [13]). If we are willing to use the 1-linear assumption, the efficiency of our proofs can be improved.

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 $\mathsf{NewSession}$ query, or if this is the second $\mathsf{NewSession}$ 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 $\mathsf{compromised}$ and reply to $\mathcal{S}$ with "correct guess". If $pw \neq pw'$, mark the record $\mathsf{interrupted}$ and reply with "wrong guess".

Upon receiving a query $(\mathsf{NewKey}, \mathsf{sid}, P_i, \mathsf{sk})$ **from** $\mathcal{S}$**, where** $|\mathsf{sk}| = n$**:**
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 $\mathsf{compromised}$, or either P_i or P_j is corrupted, then output $(\mathsf{sid}, \mathsf{sk})$ 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 $(\mathsf{sid}, \mathsf{sk}')$ to P_i.
- In any other case, pick a new random key $\mathsf{sk}' \leftarrow \{0,1\}^n$ and send $(\mathsf{sid}, \mathsf{sk}')$ to P_i.

In all cases, mark the record (P_i, P_j, pw) as $\mathsf{completed}$.

Fig. C.1. The password-based key-exchange functionality $\mathcal{F}_{\mathsf{pwKE}}$.

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 Ref. [14]. 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).

C.1.1. *Universally-Composable Password-Based Key Exchange*

In Fig. C.1 we present the functionality $\mathcal{F}_{\mathsf{pwKE}}$ for password-based key exchange, taken directly from Canetti et al. [16]. We refer the reader to their work for extensive discussion regarding the particular choices made regarding this formulation of this functionality.

References

[1] M. Abdalla, D. Catalano, C. Chevalier, D. Pointcheval, Efficient two-party password-based key exchange protocols in the UC framework, in *Cryptographers' Track—RSA 2008*. LNCS, vol. 4964 (Springer, Berlin, 2008), pp. 335–351

[2] B. Barak, R. Canetti, Y. Lindell, R. Pass, T. Rabin, Secure computation without authentication. *J. Cryptol.* **24**(4), 720–760 (2011)

[3] M. Bellare, D. Pointcheval, P. Rogaway, Authenticated key exchange secure against dictionary attacks, in *Advances in Cryptology—Eurocrypt 2000*. LNCS, vol. 1807 (Springer, Berlin, 2000), pp. 139–155

[4] M. Bellare, P. Rogaway, Entity authentication and key distribution, in *Advances in Cryptology—Crypto'93*. LNCS, vol. 773 (Springer, Berlin, 1994), pp. 232–249

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

[6] S.M. Bellovin, M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, in *IEEE Symposium on Security & Privacy* (IEEE Press, New York, 1992), pp. 72–84

[7] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, M. Yung, Systematic design of two-party authentication protocols. *IEEE J. Sel. Areas Commun.* **11**(5), 679–693 (1993)

[8] M. Blum, A. De Santis, S. Micali, G. Persiano, Noninteractive zero-knowledge. *SIAM J. Comput.* **20**(6), 1084–1118 (1991)

[9] M. Blum, P. Feldman, S. Micali, Proving security against chosen cyphertext attacks, in *Advances in Cryptology—Crypto'88*. LNCS, vol. 403 (Springer, Berlin, 1990), pp. 256–268

[10] D. Boneh, X. Boyen, H. Shacham, Short group signatures, in *Advances in Cryptology—Crypto 2004*. LNCS, vol. 3152 (Springer, Berlin, 2004), pp. 41–55

[11] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, A. Smith, Secure remote authentication using biometric data, in *Advances in Cryptology—Eurocrypt 2005*. LNCS, vol. 3494 (Springer, Berlin, 2005), pp. 147–163

[12] V. Boyko, P.D. MacKenzie, S. Patel, Provably secure password-authenticated key exchange using Diffie–Hellman, in *Advances in Cryptology—Eurocrypt 2000*. LNCS, vol. 1807 (Springer, Berlin, 2000), pp. 156–171

[13] J. Camenisch, N. Chandran, V. Shoup, A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks, in *Advances in Cryptology—Eurocrypt 2009*, ed. by A. Joux. LNCS, vol. 5479 (Springer, Berlin, 2009), pp. 351–368

[14] R. Canetti, Universally composable security: A new paradigm for cryptographic protocols, in *42nd Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Press, New York, 2001), pp. 136–145. Full version at http://eprint.iacr.org/2000/067/

[15] R. Canetti, D. Dachman-Soled, V. Vaikuntanathan, H. Wee, Efficient password authenticated key exchange via oblivious transfer, in *Public-Key Cryptography—PKC 2012*. LNCS, vol. 7293 (Springer, Berlin, 2012), pp. 449–466

[16] R. Canetti, S. Halevi, J. Katz, Y. Lindell, P.D. MacKenzie, Universally composable password-based key exchange, in *Advances in Cryptology—Eurocrypt 2005*. LNCS, vol. 3494 (Springer, Berlin, 2005), pp. 404–421

[17] R. Cramer, V. Shoup, Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption, in *Advances in Cryptology—Eurocrypt 2002*. LNCS, vol. 2332 (Springer, Berlin, 2002), pp. 45–64

[18] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, A. Sahai, Robust non-interactive zero knowledge, in *Advances in Cryptology—Crypto 2001*. LNCS, vol. 2139 (Springer, Berlin, 2001), pp. 566–598

[19] W. Diffie, M.E. Hellman, New directions in cryptography. *IEEE Trans. Inf. Theory* **22**(6), 644–654 (1976)

[20] U. Feige, D. Lapidot, A. Shamir, Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM J. Comput.* **29**(1), 1–28 (1999)

[21] R. Gennaro, Faster and shorter password-authenticated key exchange, in *5th Theory of Cryptography Conference—TCC 2008*. LNCS, vol. 4948 (Springer, Berlin, 2008), pp. 589–606

[22] R. Gennaro, Y. Lindell, A framework for password-based authenticated key exchange. *ACM Trans. Inf. Syst. Secur.* **9**(2), 181–234 (2006)

[23] O. Goldreich, Y. Lindell, Session-key generation using human passwords only. *J. Cryptol.* **19**(3), 241–340 (2006)

[24] L. Gong, T.M.A. Lomas, R.M. Needham, J.H. Saltzer, Protecting poorly chosen secrets from guessing attacks. *IEEE J. Sel. Areas Commun.* **11**(5), 648–656 (1993)

[25] V. Goyal, Positive results for concurrently secure computation in the plain model, in *53rd Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Press, New York, 2012)

[26] V. Goyal, A. Jain, R. Ostrovsky, Password-authenticated session-key generation on the Internet in the plain model, in *Advances in Cryptology—Crypto 2010*. LNCS, vol. 6223 (Springer, Berlin, 2010), pp. 277–294

[27] A. Groce, J. Katz, A new framework for efficient password-based authenticated key exchange, in *17th ACM Conf. on Computer and Communications Security (CCS)* (ACM, New York, 2010), pp. 516–525

[28] J. Groth, A. Sahai, Efficient non-interactive proof systems for bilinear groups, in *Advances in Cryptology—Eurocrypt 2008*. LNCS, vol. 4965 (Springer, Berlin, 2008), pp. 415–432

[29] S. Halevi, H. Krawczyk, Public-key cryptography and password protocols. *ACM Trans. Inf. Syst. Secur.* **2**(3), 230–268 (1999)

[30] D. Hofheinz, E. Kiltz, Secure hybrid encryption from weakened key encapsulation, in *Advances in Cryptology—Crypto 2007*. LNCS, vol. 4622 (Springer, Berlin, 2007), pp. 553–571

[31] I.R. Jeong, J. Katz, D.H. Lee, One-round protocols for two-party authenticated key exchange, in *2nd Intl. Conference on Applied Cryptography and Network Security (ACNS)*. LNCS, vol. 3089 (Springer, Berlin, 2004), pp. 220–232

[32] S. Jiang, G. Gong, Password based key exchange with mutual authentication, in *11th Annual International Workshop on Selected Areas in Cryptography (SAC)*. LNCS, vol. 3357 (Springer, Berlin, 2004), pp. 267–279

[33] J. Katz, P.D. MacKenzie, G. Taban, V.D. Gligor, Two-server password-only authenticated key exchange. *J. Comput. Syst. Sci. Int.* **78**(2), 651–669 (2012)

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

[35] J. Katz, V. Vaikuntanathan, Smooth projective hashing and password-based authenticated key exchange from lattices, in *Advances in Cryptology—Asiacrypt 2009*. LNCS, vol. 5912 (Springer, Berlin, 2009), pp. 636–652

[36] P.D. MacKenzie, S. Patel, R. Swaminathan, Password-authenticated key exchange based on RSA, in *Advances in Cryptology—Asiacrypt 2000*. LNCS, vol. 1976 (Springer, Berlin, 2000), pp. 599–613

[37] M. Naor, M. Yung, Public-key cryptosystems provably secure against chosen ciphertext attacks, in *22nd Annual ACM Symposium on Theory of Computing (STOC)* (ACM, New York, 1990), pp. 427–437

[38] M.-H. Nguyen, S. Vadhan, Simpler session-key generation from short random passwords. *J. Cryptol.* **21**(1), 52–96 (2008)

[39] T. Okamoto, Authenticated key exchange and key encapsulation in the standard model, in *Advances in Cryptology—Asiacrypt 2007*. LNCS, vol. 4833 (Springer, Berlin, 2007), pp. 474–484

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

[41] H. Shacham, A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint archive, report 2007/074