# A Scalable Password-based Group Key Exchange Protocol in the Standard Model

Michel Abdalla and David Pointcheval

Departement d'Informatique, École normale supérieure, CNRS
{Michel.Abdalla,David.Pointcheval}@ens.fr
http://www.di.ens.fr/~{mabdalla,pointche}

**Abstract.** This paper presents a secure constant-round password-based group key exchange protocol in the common reference string model. Our protocol is based on the group key exchange protocol by Burmester and Desmedt and on the 2-party password-based authenticated protocols by Gennaro and Lindell, and by Katz, Ostrovsky, and Yung. The proof of security is in the standard model and based on the notion of smooth projective hash functions. As a result, it can be instantiated under various computational assumptions, such as decisional Diffie-Hellman, quadratic residuosity, and $N$-residuosity.

**Keywords.** Smooth Projective Hash Functions, Password-based Authentication, Group Key Exchange.

## 1 Introduction

Key exchange is one of the most useful tools in public-key cryptography, allowing users to establish a common secret which they can then use in applications to achieve both privacy and authenticity. Among the examples of key exchange protocols, the most classical one is the Diffie-Hellman protocol [22]. Unfortunately, the latter only works between two players and does not provide any authentication of the players.

**Group Key Exchange.** Group key exchange protocols are designed to provide a pool of players communicating over an open network with a shared secret key which may later be used to achieve cryptographic goals like multicast message confidentiality or multicast data integrity. Secure virtual conferences involving up to one hundred participants is an example.

Due to the usefulness of group key exchange protocols, several papers have attempted to extend the basic Diffie-Hellman protocol to the group setting. Nonetheless, most of these attempts were rather informal or quite inefficient in practice for large groups. To make the analyses of such protocols more formal, Bresson et al. [11, 16] introduced a formal security model for group key exchange protocols, in the same vein as [6, 7, 4]. Moreover, they also proposed new protocols, referred to as group Diffie-Hellman protocols, using a ring structure for the communication, in which each player has to wait for the message from his predecessor before producing his own. Unfortunately, the nature of their communication structure makes their protocols quite impractical for large groups since the number of rounds of communication is linear in the number of players.

A more efficient and practical approach to the group key exchange problem is the one proposed by Burmester and Desmedt [17, 18], in which they provide a *constant-round* Diffie-Hellman variant. Their protocol is both scalable and efficient, even for large groups, since it only requires 2 rounds of broadcasts. Thus, with reasonable time-out values, one could always quickly decide whether or not a protocol has been successfully executed. Furthermore, their protocol has also been formally analyzed, in the above security framework [30].

**Password-Based Authenticated Key Exchange.** The most classical way to add authentication to key exchange protocols is to sign critical message flows. In fact, as shown by Katz and Yung [30] in the context of group key exchange protocols, this technique can be made quite general and efficient, converting any scheme that is secure against passive adversaries into one that is secure against active ones. Unfortunately, such techniques require the use of complex infrastructures to handle public keys and certificates. One way to avoid such infrastructures is to use passwords for authentication. In the latter case, the pool of players who wants to agree on a common secret key only needs to share a low-entropy password —a 4-digit pin-code, for example— against which an exhaustive search is quite easy to perform. In password-based protocols, it is clear that an outsider attacker can always guess a password and attempt to run the protocol. In case of failure, he can try again with a different guess. After each failure, the adversary can erase one password. Such an attack, known as "on-line exhaustive search" cannot be avoided, but the damage it may cause can be mitigated by other means such as limiting the number of failed login attempts. A more dangerous threat is the "off-line exhaustive search", also known as "dictionary attack". It would mean that after one failure, or even after a simple eavesdropping, the adversary can significantly reduce the number of password candidates.

In the two-party case, perhaps the most well known Diffie-Hellman variant is the encrypted key exchange protocol by Bellovin and Merritt [8]. However, its security analyses [4, 10, 13, 14] require ideal models, such as the random oracle model [5] or the ideal cipher model. The first practical password-based key exchange protocol, without random oracles, was proposed by Katz et al. [28] in the common reference string model and it is based on the Cramer-Shoup cryptosystem [19]. Their work was later extended by Gennaro and Lindell [24] using the more general smooth projective hash function primitive [19–21].

In the group key exchange case, very few protocols have been proposed with password authentication. In [12, 15], Bresson et al. showed how to adapt their group Diffie-Hellman protocols to the password-based scenario. However, as the original protocols on which they are based, their security analyses require ideal models and the total number of rounds is linear in the number of players, making their schemes impractical for large groups. More recently, several constant-round password-based group key exchange protocols have been proposed in the literature by Abdalla et al. [1], by Dutta and Barua [23], and by Kim, Lee, and Lee [31]. All of these constructions are based on the Burmester and Desmedt protocol [17, 18] and are quite efficient, but their security analyses usually require the random oracle and/or the ideal cipher models.[1] Independently of and concurrently to our work, a new constant-round password-based group key exchange protocol has been proposed by Bohli et al. [9]. Their protocol is more efficient than ours and also enjoys a security proof in the standard model.

**Contributions.** In this paper, we propose the first password-based authenticated group key exchange protocol in the standard model. To achieve this goal, we extend the Gennaro-Lindell framework [24] to the group setting, using ideas similar to those used in the Burmester-Desmedt protocol [17, 18]. In doing so, we take advantage of the smooth projective hash function primitive [20] to avoid the use of ideal models. Our protocol has several advantages. First, it is efficient both in terms of communication, only requiring 5 rounds, and in terms of computation, with a per-user computational load that is linear in the size of the group. Second, like the Burmester-Desmedt protocol, our protocol is also contributory since each member contributes equally to the generation of the common session key. Such property, as pointed out by Steiner, Tsudik and Waidner [33], may be essential for certain distributed applications. Finally, as in the Gennaro-Lindell framework [24], our protocol works in the common reference string model and is quite general, being built in a modular way from four cryptographic primitives: a labeled encryption scheme secure against chosen-ciphertext attacks, a signature scheme, a family of smooth projective hash functions, and a family of universal hash functions. Thus, it can be instantiated under various computational assumptions, such as decisional Diffie-Hellman, quadratic residuosity, and $N$-residuosity (see [24]). In particular, the Diffie-Hellman variant (based on the Cramer-Shoup cryptosystem [19]) can be seen as a generalization of the KOY protocol [28] to the group setting.

## 2 Security Model

The security model for password-based group key exchange protocols that we present here is the one by Bresson et al. [15], which is based on the model by Bellare et al. [4] for 2-party password-based key exchange protocols.

**Protocol participants.** Let $\mathcal{U}$ denote the set of potential participants in a password-based group key exchange protocol. Each participant $U \in \mathcal{U}$ may belong to several subgroups $\mathcal{G} \subseteq \mathcal{U}$, each of which has a unique password $\mathsf{pw}_{\mathcal{G}}$ associated to it. The password $\mathsf{pw}_{\mathcal{G}}$ of a subgroup $\mathcal{G}$ is known to all the users $U_i \in \mathcal{G}$.

[1] In fact, in [1], Abdalla et al. showed that the protocols by Dutta and Barua [23] and by Kim, Lee, and Lee are insecure by presenting concrete attacks against these schemes.

**Protocol execution.** The interaction between an adversary $\mathcal{A}$ and the protocol participants only occurs via oracle queries, which model the adversary capabilities in a real attack. During the execution of the protocol, the adversary may create several instances of a participant and several instances of the same participant may be active at any given time. Let $U^{\langle i \rangle}$ denote the instance $i$ of a participant $U$ and let $b$ be a bit chosen uniformly at random. The query types available to the adversary are as follows:

- *Execute*$(U_1^{\langle i_1 \rangle}, \ldots, U_n^{\langle i_n \rangle})$: This query models passive attacks in which the attacker eavesdrops on honest executions among the participant instances $U_1^{\langle i_1 \rangle}, \ldots, U_n^{\langle i_n \rangle}$. It returns the messages that were exchanged during an honest execution of the protocol.
- *Send*$(U^{\langle i \rangle}, m)$: This query models an active attack, in which the adversary may tamper with the message being sent over the public channel. It returns the message that the participant instance $U^{\langle i \rangle}$ would generate upon receipt of message $m$.
- *Reveal*$(U^{\langle i \rangle})$: This query models the misuse of session keys by a user. It returns the session key held by the instance $U^{\langle i \rangle}$.
- *Test*$(U^{\langle i \rangle})$: This query tries to capture the adversary's ability to tell apart a real session key from a random one. It returns the session key for instance $U^{\langle i \rangle}$ if $b = 1$ or a random key of the same size if $b = 0$.

**Partnering.** Following [30], we define the notion of partnering via session and partner identifiers. Let the session identifier $\mathsf{sid}^i$ of a participant instance $U^{\langle i \rangle}$ be a function of all the messages sent and received by $U^{\langle i \rangle}$ as specified by the group key exchange protocol. Let the partner identifier $\mathsf{pid}^i$ of a participant instance $U^{\langle i \rangle}$ is the set of all participants with whom $U^{\langle i \rangle}$ wishes to establish a common secret key. Two instances $U_1^{\langle i_1 \rangle}$ and $U_2^{\langle i_2 \rangle}$ are said to be partnered if and only if $\mathsf{pid}_1^{i_1} = \mathsf{pid}_2^{i_2}$ and $\mathsf{sid}_1^{i_1} = \mathsf{sid}_2^{i_2}$.

**Freshness.** Differently from [30], our definition of freshness does not take into account forward security as the latter is out of the scope of the present paper. Let $\mathsf{acc}^i$ be $\mathsf{true}$ if an instance $U^{\langle i \rangle}$ goes into an accept state after receiving the last expected protocol message and $\mathsf{false}$ otherwise. We say that an instance $U^{\langle i \rangle}$ is fresh if $\mathsf{acc}^i = \mathsf{true}$ and no *Reveal* has been asked to $U^{\langle i \rangle}$ or to any of its partners.

**Correctness.** For a protocol to be correct, it should always be the case that, whenever two instances $U_1^{\langle i_1 \rangle}$ and $U_2^{\langle i_2 \rangle}$ are partnered and have accepted, both instances should hold the same non-null session key.

**Indistinguishability.** Consider an execution of the group key exchange protocol $P$ by an adversary $\mathcal{A}$, in which the latter is given access to the *Reveal*, *Execute*, *Send*, and *Test* oracles and asks a single *Test* query to a *fresh* instance, and outputs a guess bit $b'$. Let $\textsc{Succ}$ denote the event $b'$ correctly matches the value of the hidden bit $b$ used by the *Test* oracle. The AKE-IND advantage of an adversary $\mathcal{A}$ in violating the indistinguishability of the protocol $P$ and the advantage function of the protocol $P$, when passwords are drawn from a dictionary $\mathcal{D}$, are respectively $\mathbf{Adv}^{\text{ake-ind}}_{P,\mathcal{D}}(\mathcal{A}) = 2 \cdot \Pr[\,\textsc{Succ}\,] - 1$ and $\mathbf{Adv}^{\text{ake-ind}}_{P,\mathcal{D}}(t,R) = \max_{\mathcal{A}}\{\mathbf{Adv}^{\text{ake-ind}}_{P,\mathcal{D}}(\mathcal{A})\}$ , where maximum is over all $\mathcal{A}$ with time-complexity at most $t$ and using resources at most $R$ (such as the number of queries to its oracles). The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the experiments defining the security plus the code size.

We say that a password-based group key exchange protocol $P$ is secure if the advantage of any polynomial-time adversary is only negligibly larger than $O(q/|\mathcal{D}|)$, where $q$ is number of different protocol instances to which the adversary has asked *Send* queries. Given that the dictionary size can be quite small in practice, the hidden constant in the big-$O$ notation should be as small as possible (preferably 1) for a higher level of security.

## 3 Building blocks

### 3.1 Universal Hash Function Families

One of the tools used in our protocol is a family of universal hash functions. A family $\mathcal{UH}$ of universal hash function is a map $\mathbf{K} \times \mathbf{G} \mapsto \mathbf{R}$, where $\mathbf{K}$ is the key or seed space, $\mathbf{G}$ is the domain of the hash function, and $\mathbf{R}$ is the range. For each seed or key $k \in \mathbf{K}$, we can define a particular instance $\mathsf{UH}_k : \mathbf{G} \mapsto \mathbf{R}$ of the family by fixing the key being used in the computation of the function. For simplicity, we sometimes omit the seed $k$ from the notation when referring to a particular instance of the family. Let $\mathsf{UH}_k$ be a universal hash function chosen at random from a family $\mathcal{UH}$. One of the properties of universal hash function families in which we are interested is the one that says that, if an element $g$ is chosen uniformly at random from $\mathbf{G}$, then the output distribution of $\mathsf{UH}_k(g)$ is statistically close to uniform in $\mathbf{R}$ [26].

### 3.2 Signatures

The signature scheme used in our protocol is the standard one introduced by Goldwasser, Micali, and Rivest [25]. A standard signature scheme $\mathcal{SIG} = (\mathsf{SKG}, \mathsf{Sign}, \mathsf{Ver})$ is composed of three algorithms. The key generation algorithm $\mathsf{SKG}$ takes as input $1^k$, where $k$ is a security parameter, and returns a pair $(sk, vk)$ containing the secret signing key and the public verification key. The signing algorithm $\mathsf{Sign}$ takes as input the secret key $sk$ and a message $m$ and returns a signature $\sigma$ for that message. The verification algorithm $\mathsf{Ver}$ on input $(vk, m, \sigma)$ returns 1 if $\sigma$ is a valid signature for the message $m$ with respect to the verification key $vk$.

The security notion for signature schemes needed in our proofs is strong existential unforgeability under chosen-message attacks [25]. More precisely, let

$(sk, vk)$ be a pair of secret and public keys for a signature scheme $\mathcal{SIG}$, let $\textsc{Sign}(\cdot)$ be a signing oracle which returns $\sigma = \mathsf{Sign}(sk, m)$ on input $m$, and let $\mathcal{F}$ be an adversary. Then, consider the experiment in which the adversary $\mathcal{F}$, who is given access to the public key $vk$ and to the signing oracle $\textsc{Sign}(\cdot)$, outputs a pair $(m, \sigma)$. Let $\{(m_i, \sigma_i)\}$ denote the set of queries made to the signing oracle with the respective responses and let $\textsc{Succ}$ denote the event in which $\mathsf{Ver}(vk, m', \sigma') = 1$ and that $(m', \sigma') \notin \{(m_i, \sigma_i)\}$. The SIG-SUF-CMA-*advantage* of an adversary $\mathcal{F}$ in violating the chosen message security of the signature scheme $\mathcal{SIG}$ is defined as $\mathbf{Adv}^{\text{sig-suf-cma}}_{\mathcal{SIG},\mathcal{F}}(k) = \Pr[\,\textsc{Succ}\,]$. A signature scheme $\mathcal{SIG}$ is said to be SIG-SUF-CMA-*secure* if this advantage is a negligible function in $k$ for all polynomial time adversaries (PTAs) $\mathcal{F}$ asking a polynomial number of queries to their signing oracle.

### 3.3 Labeled Encryption

The notion of labeled encryption, first formalized in the ISO 18033-2 standard [32], is a variation of the usual encryption notion that takes into account the presence of labels in the encryption and decryption algorithms. More precisely, in a labeled encryption scheme, both the encryption and decryption algorithms have an additional input parameter, referred to as a label, and the decryption algorithm should only correctly decrypt a ciphertext if its input label matches the label used to create that ciphertext.

Formally, a labeled encryption scheme $\mathcal{LPKE} = (\mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms. Via $(pk, sk) \stackrel{\$}{\leftarrow} \mathsf{LKG}(1^k)$, where $k \in \mathbb{N}$ is a security parameter, the randomized key-generation algorithm produces the public and secret keys of the scheme. Via $c \stackrel{\$}{\leftarrow} \mathsf{Enc}(pk, l, m; r)$, the randomized encryption algorithm produces a ciphertext $c$ for a label $l$ and message $m$ using $r$ as the randomness. Via $m \leftarrow \mathsf{Dec}(sk, l, c)$, the decryption algorithm decrypts the ciphertext $c$ using $l$ as the label to get back a message $m$.

The security notion for labeled encryption is similar to that of standard encryption schemes. The main difference is that, whenever the adversary wishes to ask a query to his Left-or-Right encryption oracle, in addition to providing a pair of messages $(m_0, m_1)$, he also has to provide a target label $l$ in order to obtain the challenge ciphertext $c$. Moreover, when chosen-ciphertext security (LPKE-IND-CCA) is concerned, the adversary is also allowed to query his decryption oracle on any pair $(l, c)$ as long as the ciphertext $c$ does not match the output of a query to his Left-or-Right encryption oracle whose input includes the label $l$. As shown by Bellare et al. in the case of standard encryption schemes [3], one can easily show that the Left-or-Right security notion for labeled encryption follows from the more standard Find-Then-Guess security notion (in which the adversary is only allowed a single query to his challenging encryption oracle).

### 3.4 Smooth Projective Hash Functions

The notion of projective hash function families was first introduced by Cramer and Shoup [20] as a means to design chosen-ciphertext secure encryption schemes.

Later, Gennaro and Lindell [24] showed how to use such families to build secure password-based authenticated key exchange protocols. One of the properties that makes these functions particularly interesting is that, for certain points of their domain, their values can be computed by using either a *secret* hashing key or a *public* projective key. While the computation using *secret* hashing key works for all the points in the domain of the hash function, the computation using a public *projective* key only works for a specified subset of the domain. A projective hash function family is said to be smooth if the value of the function on inputs that are outside the particular subset of the domain are independent of the projective key. In [24], the notion of smooth hash functions was presented in the context of families of hard (partitioned) subset membership problems. Here we follow the same approach.

Hard partitioned subset membership problems. Let $k \in \mathbb{N}$ be a security parameter. In a family of hard (partitioned) subset membership problem, we first specify two sets $\mathbf{X}(k)$ and $\mathbf{L}(k)$ in $\{0,1\}^{\mathbf{poly}(k)}$ such that $\mathbf{L}(k) \subseteq \mathbf{X}(k)$ as well as two distributions $D(\mathbf{L}(k))$ and $D(\mathbf{X}(k)\backslash\mathbf{L}(k))$ over $\mathbf{L}(k)$ and $\mathbf{X}(k)\backslash\mathbf{L}(k))$ respectively. Next, we specify a witness set $\mathbf{W}(k) \subseteq \{0,1\}^{\mathbf{poly}(k)}$ and a NP-relation $\mathbf{R}(k) \subseteq \mathbf{X}(k) \times \mathbf{W}(k)$ such that $x \in \mathbf{L}(k)$ if and only if there exists a witness $w \in \mathbf{W}(k)$ such that $(x, w) \in \mathbf{R}(k)$. Then, we say that a family of subset membership problems is hard if $(\mathbf{X}(k), \mathbf{L}(k), D(\mathbf{L}(k)), D(\mathbf{X}(k) \setminus \mathbf{L}(k)), \mathbf{W}(k), \mathbf{R}(k))$ instances can be efficiently generated, that a member element $x \in \mathbf{L}(k)$ can be efficiently sampled according to $D(\mathbf{L}(k))$ along with a witness $w \in \mathbf{W}(k)$ to the fact that $(x, w) \in \mathbf{R}(k)$, that non-member elements $x \in \mathbf{X}(k) \setminus \mathbf{L}(k)$ can be efficiently sampled according to $D(\mathbf{X}(k) \setminus \mathbf{L}(k))$, and that the distributions of member and non-member elements cannot be efficiently distinguished. The definition of hard *partitioned* subset membership problem is an extension of the one given above in which the set $\mathbf{X}(k)$ is partitioned in disjoint subsets $\mathbf{X}(k, i)$ for some index $i$ and for which for all $i$ it remains hard to distinguish an element $x \in \mathbf{L}(k, i)$ chosen according to a distribution $D(\mathbf{L}(k, i))$ from an element $x \in \mathbf{X}(k, i) \setminus \mathbf{L}(k, i)$ chosen according to a distribution $D(\mathbf{X}(k, i) \setminus \mathbf{L}(k, i))$.

Hard partitioned subset membership problems from labeled encryption. The families of hard partitioned subset membership problems in which we are interested are those based on LPKE-IND-CCA-secure labeled encryption schemes. More precisely, let $\mathcal{LPKE} = (\mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ be a LPKE-IND-CCA-secure labeled encryption scheme and let $pk$ be a public key outputted by the $\mathsf{LKG}$ algorithm for a given security parameter $k$. Let $\mathsf{Enc}(pk)$ denote an efficiently recognizable superset of the space of all ciphertexts that may be outputted by the encryption algorithm $\mathsf{Enc}$ when the public key is $pk$ and let $\mathbf{L}$ and $\mathbf{M}$ denote efficiently recognizable supersets of the label and message spaces. Using these sets, we can define a family of hard partitioned subset membership problems as follows. First, we define the sets $\mathbf{X}$ and $\mathbf{L}$ for the family of hard subset membership problems as $\mathbf{X}(pk) = \mathsf{Enc}(pk) \times \mathbf{L} \times \mathbf{M}$ and $\mathbf{L}(pk) = \{(c, l, m) \mid \exists r \text{ s.t. } c = \mathsf{Enc}(pk, l, m; r)\}$. Next, we define the partitioning of the sets $\mathbf{X}$ and $\mathbf{L}$ with respect to the message and label used in the encryption as $\mathbf{X}(pk, l, m) = \mathsf{Enc}(pk) \times l \times m$ and $\mathbf{L}(pk, l, m) = \{(c, l, m) \mid \exists r \text{ s.t. } c = \mathsf{Enc}(pk, l, m; r)\}$. The distribution $D(\mathbf{L}(pk, l, m))$ can then be defined by choosing a random $r \in \mathbf{R}$ and outputting the triple $(\mathsf{Enc}(pk, l, m; r), l, m)$ with $r$ as a witness. Likewise, the distribution $D(\mathbf{X}(pk, l, m)\backslash\mathbf{L}(pk, l, m))$ can be defined by choosing a random $r \in \mathbf{R}$ and outputting the triple $(\mathsf{Enc}(pk, l, m'; r), l, m)$, where $m'$ is a dummy message different from $m$ but of the same length. Finally, we define the witness set $\mathbf{W}(pk)$ to be $r$ and the NP-relation $\mathbf{R}(pk)$ in a natural way. It is easy to see that the hardness of distinguishing non-members from members follows from the LPKE-IND-CCA security of the labeled encryption scheme.

Smooth projective hash functions. Let $\mathcal{HLPKE}(pk) = (\mathbf{X}(pk), \mathbf{L}(pk), D(\mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m)), D(\mathbf{L}(pk, l, m)), \mathbf{W}(pk), \mathbf{R}(pk))$ be a family of hard (partitioned) subset membership problems based on a LPKE-IND-CCA-secure labeled encryption scheme $\mathcal{LPKE}$ with security parameter $k$. A family of smooth projective hash functions $\mathcal{HASH}(pk) = (\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ associated with $\mathcal{HLPKE}$ consists of four algorithms. Via $hk \stackrel{\$}{\leftarrow} \mathsf{HashKG}(pk)$, the randomized key-generation algorithm produces hash keys $hk \in \mathbf{HK}(pk)$, where $k \in \mathbb{N}$ is a security parameter and $pk$ is the public key of a labeled encryption scheme $\mathcal{LPKE}$. Via $phk \stackrel{\$}{\leftarrow} \mathsf{ProjKG}(hk, l, c)$, the randomized key projection algorithm produces projected hash keys $phk \in \mathbf{PHK}(pk)$ for a hash key $hk$ with respect to label $l$ and ciphertext $c$. Via $g \leftarrow \mathsf{Hash}(hk, c, l, m)$, the hashing algorithm computes the hash value $g \in \mathbf{G}(pk)$ of $(c, l, m)$ using the hash key $hk$. Via $g \leftarrow \mathsf{ProjHash}(phk, c, l, m; r)$, the projected hashing algorithm computes the hash value $g \in \mathbf{G}(pk)$ of $(c, l, m)$ using the projected hash key $phk$ and a witness $r$ to the fact that $c$ is a valid encryption of message $m$ with respect to the public-key $pk$ and label $l$.

Properties. The properties of smooth projective hash functions in which we are interested are correctness, smoothness, and pseudorandomness.

**Correctness.** Let $\mathcal{LPKE}$ be a labeled encryption scheme and let $pk$ be a public key outputted by the $\mathsf{LKG}$ algorithm for a given security parameter $k$. Let $c = \mathsf{Enc}(pk, l, m; r)$ be the ciphertext for a message $m$ with respect to public key $pk$ and label $l$ computed using $r$ as the randomness. Then, for any hash key $hk \in \mathbf{HK}(pk)$ and projected hash key $phk \stackrel{\$}{\leftarrow} \mathsf{ProjKG}(hk, l, c)$, the values $\mathsf{Hash}(hk, c, l, m)$ and $\mathsf{ProjHash}(phk, c, l, m, r)$ are the same.

**Smoothness.** Let $hk \in \mathbf{HK}(pk)$ be a hash key and let $phk \in \mathbf{PHK}(pk)$ be a projected hash key for $hk$ with respect to $l$ and $c$. Then, for every triple $(c, l, m)$ for which $c$ is *not* a valid encryption of message $m$ with respect to the public-key $pk$ and label $l$ (i.e., $(c, l, m) \in \mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m)$), the hash value $g = \mathsf{Hash}(hk, c, l, m)$ is statistically close to uniform in $\mathbf{G}$ and independent of the values $(phk, c, l, m)$.

**Pseudorandomness.** Let $\mathcal{LPKE}$ be a LPKE-IND-CCA-secure labeled encryption scheme, let $pk$ be a public key outputted by the $\mathsf{LKG}$ algorithm for a given security parameter $k$, and let $(l, m) \in \mathbf{L} \times \mathbf{M}$ be a message-label pair. Then, for uniformly chosen hash key $hk \in \mathbf{HK}(pk)$ and randomness $r \in \mathbf{R}(pk)$, the distributions $\{c = \mathsf{Enc}(pk, l, m; r), l, m, phk \stackrel{\$}{\leftarrow} \mathsf{ProjKG}(hk, l, c), g \leftarrow \mathsf{Hash}(hk, c, l, m)\}$ and $\{c = \mathsf{Enc}(pk, l, m; r), l, m, phk \stackrel{\$}{\leftarrow} \mathsf{ProjKG}(hk, l, c), g \stackrel{\$}{\leftarrow} \mathbf{G}\}$ are computationally indistinguishable.

Examples. To provide the reader with an idea of how efficient smooth projective hash functions are, we recall here the example given in [24] based on the Cramer-Shoup encryption scheme [19].

The labeled version of the Cramer-Shoup scheme works as follows. Let $G$ be a cyclic group of prime order $q$ where $q$ is large. The key generation algorithm chooses two additional random generators $g_1$, $g_2$ in $G$, a collision-resistant hash function $H$, and random values $z$, $\tilde{z}_1$, $\tilde{z}_2$, $\hat{z}_1$, $\hat{z}_2$ in $Z_q$ with $z \neq 0$. The secret key is set to $(z, \tilde{z}_1, \tilde{z}_2, \hat{z}_1, \hat{z}_2)$ and the public key is defined to be $(h, \tilde{h}, \hat{h}, g_1, g_2, H)$, where $h = g_1^z$, $\tilde{h} = g_1^{\tilde{z}_1} g_2^{\tilde{z}_2}$, and $\hat{h} = g_1^{\hat{z}_1} g_2^{\hat{z}_2}$. To encrypt a message $m \in G$ with respect to label $l$, the sender chooses $r \in Z_q$ , and computes $u_1 = g_1^r$ , $u_2 = g_2^r$ , $e = h^r \cdot m$, $\theta = H(l, u_1, u_2, e)$ and $v = (\tilde{h}\hat{h}^\theta)^r$. The ciphertext is $c = (u_1, u_2, e, v)$. To decrypt a ciphertext $c = (u_1, u_2, e, v)$ with respect to label $l$, the receiver computes $\theta = H(l, u_1, u_2, e)$ and tests if $v$ equals $u_1^{\tilde{z}_1+\theta\hat{z}_1} u_2^{\tilde{z}_2+\theta\hat{z}_2}$. If equality does not hold, it outputs $\perp$; otherwise, it outputs $m = eu_1^{-z}$.

The smooth projective hashing for the labeled Cramer-Shoup encryption scheme is then defined as follows. The hash key generation algorithm $\mathsf{HashKG}$ simply sets the key $hk$ to be the tuple $(a_1, a_2, a_3, a_4)$ where each $a_i$ is a random value in $Z_q$. The key projection function $\mathsf{ProjKG}$, on input $(hk, l, c)$, first computes $\theta = H(l, u_1, u_2, e)$ and outputs $phk = g_1^{a_1} g_2^{a_2} h^{a_3} (\tilde{h}\hat{h}^\theta)^{a_4}$. The hash function $\mathsf{Hash}$ on input $(hk, c, l, m)$ outputs $u_1^{a_1} u_2^{a_2} (e/m)^{a_3} v^{a_4}$. The projective hash function $\mathsf{ProjHash}$ on input $(phk, c, l, m, r)$ simply outputs $phk^r$.

## 4 A scalable password-based group key exchange protocol

In this section, we finally present our password-based group key exchange protocol. Our protocol is an extension of the Gennaro-Lindell password-based key exchange protocol [24] to the group setting and uses ideas similar to those used in the Burmester-Desmedt group key exchange protocol [18]. The Gennaro-Lindell protocol itself is an abstraction of the password-based key exchange protocol of Katz, Ostrovsky, and Yung [28, 29]. Like the Gennaro-Lindell protocol, our protocol is built in a modular way from four cryptographic primitives: a LPKE-IND-CCA-secure labeled encryption scheme, a signature scheme, a family of smooth projective hash functions, and a family of universal hash functions. Thus, our protocol enjoys efficient instantiations based on the decisional Diffie-Hellman, quadratic residuosity, and $N$-residuosity assumptions (see [24]). Like the Burmester-Desmedt group key exchange protocol, our protocol only requires a constant number of rounds and low per-user computation.

As done in the Gennaro-Lindell protocol, we also assume the existence of a mechanism to allow parties involved in the protocol to differentiate between concurrent executions as well as identify the other parties with which they are interacting. As in their case, this requirement is only needed for the correct operation of the protocol. No security requirement is imposed on this mechanism.

### 4.1 Protocol Description

Overview. As in the Burmester-Desmedt protocol, our protocol assumes a ring structure for the users so that we can refer to the predecessor and successor of a user. Moreover, we associate each user with an index $i$ between 1 and $n$, where $n$ is the size of the group. After deciding on the order of the users, our protocol works as follows. First, each user in the group executes two correlated instances of the Gennaro-Lindell protocol, one with his predecessor and one with his successor so each user can authenticate his neighbors (this accounts for the first 3 rounds of the protocol). However, instead of generating a single session key in each of these instances, we modify the original Gennaro-Lindell protocol so that two independent session keys are generated in each session (this requires an extra hash key and an extra projection key per user). We then use the first one of these as a test key to authenticate the neighbor with whom that key is shared and we use the other one to help in the computation of the group session key, which is defined as the product of these latter keys. To do so, we add one more round of communication like in the Burmester-Desmedt protocol, so that each user computes and broadcasts the ratio of the session keys that he shares with his predecessor and successor. After this round, each user is capable of computing the group session key. However, to ensure that all users agree on the same key, a final round of signatures is added to the protocol to make sure that all users compute the group session key based on the same transcript. The key used to verify the signature of a user is the same one transmitted by that user in the first round of the Gennaro-Lindell protocol.

For a pictorial description of our protocol, please refer to Fig. 1. The formal description follows.

Description. Let $\mathcal{LPKE} = (\mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ be a labeled encryption scheme, let $\mathcal{SIG} = (\mathsf{SKG}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme, and let $\mathcal{HASH}(pk) = (\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ be a family smooth projective hash functions based on $\mathcal{LPKE}$. Let $\mathsf{UH} : \mathbf{G} \mapsto \{0,1\}^{2l}$ and $\mathsf{UH}' : \mathbf{G} \mapsto \{0,1\}^{l}$ be two universal hash functions chosen uniformly at random from the families $\mathcal{UH}$ and $\mathcal{UH}'$ and let $\mathsf{UH}_1(g)$ and $\mathsf{UH}_2(g)$ refer to the first and second halves of $\mathsf{UH}(g)$. Let $U_1, \ldots, U_n$ be the users wishing to establish a common secret key and let $\mathsf{pw}$ be their joint password chosen uniformly at random from a dictionary $\mathbf{Dict}$ of size $N$. We assume $\mathsf{pw}$ either lies in the message space $\mathbf{M}$ of $\mathcal{LPKE}$ or can be easily mapped to it. Our protocol has a total of five rounds of communication and works as follows.

**Initialization.** A trusted server runs the key generation algorithm $\mathsf{LKG}$ on input $1^k$, where $k \in \mathbb{N}$ is a security parameter, to obtain a pair $(pk, sk)$ of secret and public keys and publishes the public key $pk$ along with randomly selected universal hash function $\mathsf{UH}$ and $\mathsf{UH}'$ from the families $\mathcal{UH}$ and $\mathcal{UH}'$.

**Round 1.** In this first round, each player $U_i$ for $i = 1, \ldots, n$ starts by setting the partner identifier $\mathsf{pid}_i$ to $\{U_1, \ldots, U_n\}$. Then, each player $U_i$ generates a pair $(sk_i, vk_i)$ of secret and public keys for a signature scheme and a label $l_i = vk_i \parallel U_1 \parallel \ldots \parallel U_n$. Next, each player encrypts the joint group password $\mathsf{pw}$ using the encryption algorithm $\mathsf{Enc}$ with respect to the public key $pk$ and label $l_i$ using $r_i^{\mathtt{R}}$ as the randomness. Let $c_i^{\mathtt{R}}$ denote the resulting ciphertext (i.e., $c_i^{\mathtt{R}} = \mathsf{Enc}(pk, l_i, \mathsf{pw}; r_i^{\mathtt{R}})$). At the end of this round, each player $U_i$ broadcasts the pair $(l_i, c_i^{\mathtt{R}})$.

**Round 2.** In this second round, each player $U_i$ for $i = 1, \ldots, n$ encrypts once more the joint group password $\mathsf{pw}$ using the encryption algorithm $\mathsf{Enc}$ with respect to the public key $pk$ and label $l_i$ using $r_i^{\mathtt{L}}$ as the randomness. Let $c_i^{\mathtt{L}}$ denote the resulting ciphertext (i.e., $c_i^{\mathtt{L}} = \mathsf{Enc}(pk, l_i, \mathsf{pw}; r_i^{\mathtt{L}})$). Next, each player $U_i$ chooses a hash key $hk_i^{\mathtt{L}}$ uniformly at random from $\mathbf{HK}(pk)$ for the smooth projective hash function and then generates a projection key $phk_i^{\mathtt{L}}$ for it with respect to the pair $(c_{i-1}^{\mathtt{R}}, l_{i-1})$. That is, $phk_i^{\mathtt{L}} \xleftarrow{\$} \mathsf{ProjKG}(hk_i^{\mathtt{L}}, l_{i-1}, c_{i-1}^{\mathtt{R}})$. Here and in other parts of the protocol, the indices are taken modulo $n$. At the end of this round, each player $U_i$ broadcasts the pair $(c_i^{\mathtt{L}}, phk_i^{\mathtt{L}})$.

**Round 3.** In this round, player $U_i$ first chooses two new hash keys $hk_i$ and $hk_i^{\mathtt{R}}$ uniformly at random from $\mathbf{HK}(pk)$ for the smooth projective hash function. Next, player $U_i$ generates two projection keys $phk_i$ and $phk_i^{\mathtt{R}}$ for the hash keys $hk_i$ and $hk_i^{\mathtt{R}}$, both with respect to the pair $(c_{i+1}^{\mathtt{L}}, l_{i+1})$. That is, $phk_i \xleftarrow{\$} \mathsf{ProjKG}(hk_i, l_{i+1}, c_{i+1}^{\mathtt{L}})$ and $phk_i^{\mathtt{R}} \xleftarrow{\$} \mathsf{ProjKG}(hk_i^{\mathtt{R}}, l_{i+1}, c_{i+1}^{\mathtt{L}})$. Then, player $U_i$ computes a test master key $X_i^{\mathtt{R}} = K_{i+1}^{\mathtt{L}} \cdot K_i^{\mathtt{R}}$ for its successor, where $K_i^{\mathtt{L}} \triangleq \mathsf{Hash}(hk_i^{\mathtt{L}}, c_{i-1}^{\mathtt{R}}, l_{i-1}, \mathsf{pw})$ and $K_i^{\mathtt{R}} \triangleq \mathsf{Hash}(hk_i^{\mathtt{R}}, c_{i+1}^{\mathtt{L}}, l_{i+1}, \mathsf{pw})$. Note that player $U_i$ can compute $K_i^{\mathtt{R}}$ using $hk_i^{\mathtt{R}}$ and $K_{i+1}^{\mathtt{L}}$ using $phk_{i+1}^{\mathtt{L}}$ and the witness $r_i^{\mathtt{R}}$ to the fact that $c_i^{\mathtt{R}}$ is a valid encryption of $\mathsf{pw}$ with respect to $pk$ and $l_i$. Finally, player $U_i$ computes a test key $test_i^{\mathtt{R}} = \mathsf{UH}_1(X_i^{\mathtt{R}})$, sets $T_i^{\mathtt{R}} = U_i \parallel U_{i+1} \parallel c_i^{\mathtt{R}} \parallel c_{i+1}^{\mathtt{L}} \parallel phk_i \parallel phk_i^{\mathtt{R}} \parallel phk_{i+1}^{\mathtt{L}} \parallel test_i^{\mathtt{R}}$, and computes a signature $\sigma_i^{\mathtt{R}}$ on $T_i^{\mathtt{R}}$ using $sk_i$. At the end of this round, player $U_i$ broadcasts the tuple $(phk_i, phk_i^{\mathtt{R}}, test_i^{\mathtt{R}}, \sigma_i^{\mathtt{R}})$.

**Round 4.** In this round, each player $U_i$ first verifies if the signature $\sigma_{i-1}^{\mathtt{R}}$ on the transcript $T_{i-1}^{\mathtt{R}}$ is correct using $vk_{i-1}$. If this check fails, then player $U_i$ halts and sets $\mathsf{acc}_i = \mathsf{false}$. Otherwise, player $U_i$ computes the values $K_i^{\mathtt{L}}$ and $K_{i-1}^{\mathtt{R}}$, using the hash key $hk_i^{\mathtt{L}}$ and the projection key $phk_{i-1}^{\mathtt{R}}$ along with the witness $r_i^{\mathtt{L}}$ to the fact that $c_i^{\mathtt{L}}$ is a valid encryption of $\mathsf{pw}$ with respect to $pk$ and $l_i$. That is, $K_i^{\mathtt{L}} = \mathsf{Hash}(hk_i^{\mathtt{L}}, c_{i-1}^{\mathtt{R}}, l_{i-1}, \mathsf{pw})$ and $K_{i-1}^{\mathtt{R}} = \mathsf{ProjHash}(phk_{i-1}^{\mathtt{R}}, c_i^{\mathtt{L}}, l_i, \mathsf{pw}, r_i^{\mathtt{L}})$. Next, player $U_i$ computes the test master key $X_i^{\mathtt{L}} = K_i^{\mathtt{L}} \cdot K_{i-1}^{\mathtt{R}}$ for its predecessor and verifies if $test_{i-1}^{\mathtt{R}} = \mathsf{UH}_1(X_i^{\mathtt{L}})$. Once again, if this test fails, then player $U_i$ halts and sets $\mathsf{acc}_i = \mathsf{false}$. If this test succeeds, then player $U_i$ computes a test key $test_i^{\mathtt{L}} = \mathsf{UH}_2(X_i^{\mathtt{L}})$ for its predecessor and an auxiliary key $X_i = K_i / K_{i-1}$, where $K_i \triangleq \mathsf{Hash}(hk_i, c_{i+1}^{\mathtt{L}}, l_{i+1}, \mathsf{pw})$. More precisely, player $U_i$ computes the value $K_i$ using the hash key $hk_i$ and the value $K_{i-1}$ using the projection key $phk_{i-1}$ along with the witness $r_i^{\mathtt{L}}$ to the fact that $c_i^{\mathtt{L}}$ is a valid encryption of $\mathsf{pw}$ with respect to $pk$ and $l_i$. Finally, each player $U_i$ broadcasts the pair $(X_i, test_i^{\mathtt{L}})$.

**Round 5.** First, each player $U_i$ checks whether $test_{i+1}^{\mathtt{L}} = \mathsf{UH}_2(X_i^{\mathtt{R}})$ and whether $\prod_{l=1}^{n} X_l = 1$. If any of these tests fails, then player $U_i$ halts and sets $\mathsf{acc}_i = \mathsf{false}$. Otherwise, each player $U_i$ sets $T_j = vk_j \parallel U_j \parallel c_j \parallel phk_j \parallel phk_j^{\mathtt{L}} \parallel phk_j^{\mathtt{R}} \parallel X_j \parallel X_j^{\mathtt{L}}$ for $j = 1, \ldots, n$ and $T = T_1 \parallel \ldots \parallel T_n$ and then signs it using $sk_i$ to obtain $\sigma_i$. Finally, each player $U_i$ broadcasts $\sigma_i$.

**Finalization.** Each player $U_i$ checks for $j \neq i$ whether $\sigma_j$ is a valid signature on $T$ with respect to $vk_j$. If any of these checks fails, then player $U_i$ halts and sets $\mathsf{acc}_i = \mathsf{false}$. Otherwise, player $U_i$ sets $\mathsf{acc}_i = \mathsf{true}$ and computes the master key $MSK = \prod_{j=1}^{n} K_j = K_i^n \cdot X_{i+1}^{n-1} \cdot X_{i+2}^{n-2} \cdot \ldots \cdot X_{i+n-3}^{2} \cdot X_{i+n-1}$, and the session key $SK = \mathsf{UH}'(MSK)$. Each player $U_i$ also sets the session identifier $\mathsf{sid}_i$ to $T$.

**Observation.** Let $K_i \triangleq \mathsf{Hash}(hk_i, c_{i+1}^{\mathtt{L}}, l_{i+1}, \mathsf{pw})$, $K_i^{\mathtt{R}} \triangleq \mathsf{Hash}(hk_i^{\mathtt{R}}, c_{i+1}^{\mathtt{L}}, l_{i+1}, \mathsf{pw})$, and $K_i^{\mathtt{L}} \triangleq \mathsf{Hash}(hk_i^{\mathtt{L}}, c_{i-1}^{\mathtt{R}}, l_{i-1}, \mathsf{pw})$ denote temporary keys. In a normal execution of the protocol, the temporary keys $K_i$ and $K_i^{\mathtt{R}}$ are known to both player $U_i$ (who knows $hk_i$ and $hk_i^{\mathtt{R}}$) and his successor $U_{i+1}$ (who knows $phk_i$, $phk_i^{\mathtt{R}}$, and the witness $r_{i+1}^{\mathtt{L}}$ to the fact that $c_{i+1}^{\mathtt{L}}$ is a valid encryption of $\mathsf{pw}$ with respect to $pk$ and $l_{i+1}$). Likewise, the temporary key $K_i^{\mathtt{L}}$ is known to both player $U_i$ (who knows $hk_i^{\mathtt{L}}$) and his predecessor $U_{i-1}$ (who knows $phk_i^{\mathtt{R}}$ and the witness $r_{i-1}^{\mathtt{R}}$ to the fact that $c_{i-1}^{\mathtt{R}}$ is a valid encryption of $\mathsf{pw}$ with respect to $pk$ and $l_{i-1}$).

### 4.2 Correctness and Security

Correctness. In an honest execution of the protocol, it is easy to verify that all participants in the protocol will terminate by accepting and computing the same values for the partner identifier, session identifiers, and the session key. The session key in this case is equal to $\prod_{j=1}^{n} \mathsf{Hash}(hk_j, c_{j+1}, l_{j+1}, \mathsf{pw}) = \prod_{j=1}^{n} K_j$.

Security. The intuition behind the security of our protocol is quite simple. Due to the security properties of the underlying Gennaro-Lindell protocol, each user is able to authenticate its neighbors and safely share session keys with them. Due to the properties of the signature scheme, all users in the group are able to ensure that they had received the same messages and that they will generate the same group session key. As the following theorem shows, the $\mathcal{GPAKE}$ protocol described above and in Fig. 1 is a secure password-based authenticated group key exchange protocol as long as the primitives on which the protocol is based meet the appropriate security notion described in the theorem.

**Theorem 1.** *Let $\mathcal{LPKE}$ be a labeled encryption secure against chosen-ciphertext attacks, let $\mathcal{HASH}$ be a family of smooth projective hash functions, let $\mathcal{UH}$ and $\mathcal{UH}'$ be families of universal hash functions, and let $\mathcal{SIG}$ be a signature scheme that is unforgeable against chosen-message attacks. Let $\mathcal{GPAKE}$ denote the protocol built from these primitives as described above and let $\mathcal{A}$ be an adversary against $\mathcal{GPAKE}$. Then, the advantage function $\mathbf{Adv}_{\mathcal{GPAKE},\mathcal{A}}^{\text{ake-ind}}(k)$ is only negligibly larger than $O(q/N)$, where $q$ denotes the maximum number of different protocol instances to which $\mathcal{A}$ has asked* Send *queries and $N$ is the dictionary size.*

The proof can be found in the full version of this paper [2]. In it, we actually show that the security of our protocol is only negligibly larger than $(q_{\mathit{send}\text{-}1} + q_{\mathit{send}\text{-}2})/N$, where $q_{\mathit{send}\text{-}1}$ and $q_{\mathit{send}\text{-}2}$ represent the maximum number of *Send* queries that the adversary can ask with respect to the first and second round of communication and $N$ is dictionary size. Even though we believe this security level is good enough for groups of small to medium sizes, it may not be sufficient in cases where the number of users in a group is large and the dictionary size is small. In the latter case, it would be desirable to have a scheme whose security is only negligibly larger than the number of sessions (and not protocol instances) over the size of the dictionary. Unfortunately, the latter cannot be achieved by our protocol as it is possible for an active adversary to test in the same session a number of passwords that is linear in the total number of users, for instance by playing the role of every other user.

### 4.3 Efficiency

Our protocol is quite efficient, only requiring a small amount of computation by each user. In what concerns encryption and hash computations, each user only has to perform 2 encryptions, 3 projection key generations, 3 hash computations, 3 projected hash computations, and 5 universal hash computations. The most expensive part of our protocol, which is linear in the group size, is the number of signature verifications and the master session key computation. While the latter computation can be improved by using algorithms for multi-exponentiations, the former can be improved by using two-time signature schemes.

It is worth mentioning that, as done by Katz et al. [27] in the case of the KOY protocol [28], one could also improve the efficiency of our protocol by using two different encryption schemes when computing the ciphertexts $c_i^{\mathtt{R}}$ and $c_i^{\mathtt{L}}$ broadcasted in the first and second rounds. While the computation of the ciphertexts $c_i^{\mathtt{R}}$ would require a CCA-secure labeled encryption scheme, the computation of the ciphertexts $c_i^{\mathtt{L}}$ would only require a CPA-secure encryption scheme.

### 4.4 Future Work

One issue not addressed in the current paper is whether our protocol remains secure in the presence of *Corrupt* queries, through which the adversary can learn the values of the long-term secret keys held by a user. This is indeed a significant limitation of our security model which we expect to address in the full version of this paper. In fact, we do hope to be able to prove that our protocol achieves forward security according to the definition given in [30].

## Acknowledgements

The authors were supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by France Telecom R&D as part of the contract CIDRE, between France Telecom R&D and École normale supérieure.

User $U_i$

$$
\begin{array}{l}
\mathsf{pid}_i = \{U_1, \ldots, U_n\} \\
(sk_i, vk_i) \stackrel{\$}{\leftarrow} \mathsf{SKG}(1^k) \\
l_i = vk_i \,\|\, U_1 \,\|\, \ldots \,\|\, U_n \\
c_i^{\mathsf{R}} = \mathsf{Enc}(pk, l_i, \mathsf{pw}; r_i^{\mathsf{R}})
\end{array}
$$

$$\xrightarrow{l_i,\ c_i^{\mathsf{R}}}$$

$$
\begin{array}{l}
hk_i^{\mathsf{L}} \stackrel{\$}{\leftarrow} \mathbf{HK}(pk) \\
phk_i^{\mathsf{L}} \stackrel{\$}{\leftarrow} \mathsf{ProjKG}(hk_i^{\mathsf{L}}, l_{i-1}, c_{i-1}^{\mathsf{R}}) \\
c_i^{\mathsf{L}} = \mathsf{Enc}(pk, l_i, \mathsf{pw}; r_i^{\mathsf{L}})
\end{array}
$$

$$\xrightarrow{phk_i^{\mathsf{L}},\ c_i^{\mathsf{L}}}$$

$$
\begin{array}{l}
hk_i,\ hk_i^{\mathsf{R}} \stackrel{\$}{\leftarrow} \mathbf{HK}(pk) \\
phk_i \stackrel{\$}{\leftarrow} \mathsf{ProjKG}(hk_i, l_{i+1}, c_{i+1}^{\mathsf{L}}) \\
phk_i^{\mathsf{R}} \stackrel{\$}{\leftarrow} \mathsf{ProjKG}(hk_i^{\mathsf{R}}, l_{i+1}, c_{i+1}^{\mathsf{L}}) \\
K_{i+1}^{\mathsf{L}} = \mathsf{ProjHash}(phk_{i+1}^{\mathsf{L}}, c_i^{\mathsf{R}}, l_i, \mathsf{pw}, r_i^{\mathsf{R}}) \\
K_i^{\mathsf{R}} = \mathsf{Hash}(hk_i^{\mathsf{R}}, c_{i+1}^{\mathsf{L}}, l_{i+1}, \mathsf{pw}) \\
X_i^{\mathsf{R}} = K_{i+1}^{\mathsf{L}} \cdot K_i^{\mathsf{R}} \\
test_i^{\mathsf{R}} = \mathsf{UH}_1(X_i^{\mathsf{R}}) \\
\sigma_i^{\mathsf{R}} = \mathsf{Sign}(sk_i, T_i^{\mathsf{R}})
\end{array}
$$

$$\xrightarrow{phk_i,\ phk_i^{\mathsf{R}},\ test_i^{\mathsf{R}},\ \sigma_i^{\mathsf{R}}}$$

$$
\begin{array}{l}
\text{if } \mathsf{Ver}(vk_{i-1}, T_{i-1}^{\mathsf{R}}, \sigma_{i-1}^{\mathsf{R}}) = 0 \text{ then } \mathsf{acc}_i = \mathsf{false} \\
K_i^{\mathsf{L}} = \mathsf{Hash}(hk_i^{\mathsf{L}}, c_{i-1}^{\mathsf{R}}, l_{i-1}, \mathsf{pw}) \\
K_{i-1}^{\mathsf{R}} = \mathsf{ProjHash}(phk_{i-1}^{\mathsf{R}}, c_i^{\mathsf{L}}, l_i, \mathsf{pw}, r_i^{\mathsf{L}}) \\
X_i^{\mathsf{L}} = K_i^{\mathsf{L}} \cdot K_{i-1}^{\mathsf{R}} \\
\text{if } test_{i-1}^{\mathsf{R}} \neq \mathsf{UH}_1(X_i^{\mathsf{L}}) \text{ then } \mathsf{acc}_i = \mathsf{false} \\
test_i^{\mathsf{L}} = \mathsf{UH}_2(X_i^{\mathsf{L}}) \\
K_i = \mathsf{Hash}(hk_i, c_{i+1}^{\mathsf{L}}, l_{i+1}, \mathsf{pw}) \\
K_{i-1} = \mathsf{ProjHash}(phk_{i-1}, c_i^{\mathsf{L}}, l_i, \mathsf{pw}, r_i^{\mathsf{L}}) \\
X_i = K_i / K_{i-1}
\end{array}
$$

$$\xrightarrow{X_i,\ test_i^{\mathsf{L}}}$$

$$
\begin{array}{l}
\text{if } test_{i+1}^{\mathsf{L}} \neq \mathsf{UH}_2(X_i^{\mathsf{R}}) \text{ then } \mathsf{acc}_i = \mathsf{false} \\
\text{if } \prod_{l=1}^{n} X_l \neq 1 \text{ then } \mathsf{acc}_i = \mathsf{false} \\
T = T_1 \,\|\, \ldots \,\|\, T_n \\
\sigma_i = \mathsf{Sign}(sk_i, T)
\end{array}
$$

$$\xrightarrow{\sigma_i}$$

$$
\begin{array}{l}
\text{for } j = 1, \ldots, i-1, i+1, \ldots, n \\
\quad \text{if } \mathsf{Ver}(vk_j, T, \sigma_j) = 0 \text{ then } \mathsf{acc}_i = \mathsf{false} \\
MSK = K_i^n \cdot \prod_{j=1}^{n-1} X_{i+j}^{n-j} \\
SK = \mathsf{UH}'(MSK) \\
\mathsf{acc}_i = \mathsf{true} \\
\mathsf{sid}_i = T
\end{array}
$$

**Fig. 1.** An honest execution of the password-authenticated group key exchange protocol by player $U_i$ in a group $\{U_1, \ldots, U_n\}$, where $T_i^{\mathsf{R}} = U_i \,\|\, U_{i+1} \,\|\, c_i^{\mathsf{R}} \,\|\, c_{i+1}^{\mathsf{L}} \,\|\, phk_i \,\|\, phk_i^{\mathsf{R}} \,\|\, phk_{i+1}^{\mathsf{L}} \,\|\, test_i^{\mathsf{R}}$ and $T_i = vk_i \,\|\, U_i \,\|\, c_i \,\|\, phk_i \,\|\, phk_i^{\mathsf{L}} \,\|\, phk_i^{\mathsf{R}} \,\|\, X_i \,\|\, X_i^{\mathsf{L}}$ for $i = 1, \ldots, n$.

## References


1. M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. Password-based group key exchange in a constant number of rounds. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 427–442. Springer-Verlag, Berlin, Germany, Apr. 2006.
2. M. Abdalla and D. Pointcheval. A scalable password-based group key exchange protocol in the standard model. Full version of current paper. Available from authors' web pages.
3. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer-Verlag, Berlin, Germany, May 2000.
4. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer-Verlag, Berlin, Germany, May 2000.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
6. M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, Berlin, Germany, Aug. 1994.
7. M. Bellare and P. Rogaway. Provably secure session key distribution — the three party case. In *28th ACM STOC*, pages 57–66. ACM Press, May 1996.
8. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.
9. J.-M. Bohli, M. I. G. Vasco, and R. Steinwandt. Password-authenticated constant-round group key establishment with a common reference string. Cryptology ePrint Archive, Report 2006/214, 2006. `http://eprint.iacr.org/`.
10. V. Boyko, P. D. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer-Verlag, Berlin, Germany, May 2000.
11. E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group Diffie-Hellman key exchange – the dynamic case. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 290–309. Springer-Verlag, Berlin, Germany, Dec. 2001.
12. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman key exchange secure against dictionary attacks. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 497–514. Springer-Verlag, Berlin, Germany, Dec. 2002.
13. E. Bresson, O. Chevassut, and D. Pointcheval. Security proofs for an efficient password-based key exchange. In *ACM CCS 03*, pages 241–250. ACM Press, Oct. 2003.
14. E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. In F. Bao, R. Deng, and J. Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 145–158. Springer-Verlag, Berlin, Germany, Mar. 2004.
15. E. Bresson, O. Chevassut, and D. Pointcheval. A security solution for IEEE 802.11's ad-hoc mode: Password authentication and group Diffie-Hellman key exchange. *International Journal of Wireless and Mobile Computing*, 2005. To appear.
16. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *ACM CCS 01*, pages 255–264. ACM Press, Nov. 2001.

17. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system (extended abstract). In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 275–286. Springer-Verlag, Berlin, Germany, May 1994.
18. M. Burmester and Y. Desmedt. A secure and scalable group key exchange system. *Information Processing Letters*, 94(3):137–143, May 2005.
19. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, Berlin, Germany, Aug. 1998.
20. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer-Verlag, Berlin, Germany, Apr. / May 2002.
21. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
22. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
23. R. Dutta and R. Barua. Password-based encrypted group key agreement. *International Journal of Network Security*, 3(1):30–41, July 2006. `http://isrc.nchu.edu.tw/ijns`.
24. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer-Verlag, Berlin, Germany, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.
25. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.
26. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
27. J. Katz, P. D. MacKenzie, G. Taban, and V. D. Gligor. Two-server password-only authenticated key exchange. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 1–16. Springer-Verlag, Berlin, Germany, June 2005.
28. J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer-Verlag, Berlin, Germany, May 2001.
29. J. Katz, R. Ostrovsky, and M. Yung. Forward secrecy in password-only key exchange protocols. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 29–44. Springer-Verlag, Berlin, Germany, Sept. 2002.
30. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 110–125. Springer-Verlag, Berlin, Germany, Aug. 2003.
31. H.-J. Kim, S.-M. Lee, and D. H. Lee. Constant-round authenticated group key exchange for dynamic groups. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 245–259. Springer-Verlag, Berlin, Germany, Dec. 2004.
32. V. Shoup. ISO 18033-2: An emerging standard for public-key encryption. `http://shoup.net/iso/std6.pdf`, Dec. 2004. Final Committee Draft.
33. M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, Aug. 2000.