Password-Authenticated Group Key Establishment from Smooth Projective Hash Functions

Jens-Matthias Bohli[a], María Isabel González Vasco[b*] and Rainer Steinwandt[c]
[a]Hochschule Mannheim, Germany,
j.bohli@hs-mannheim.de
[b]MACIMTE, Área de Matemática Aplicada, Universidad Rey Juan Carlos
C/ Tulipán s/n. 28933, Móstoles, Madrid, Spain
mariaisabel.vasco@urjc.es
[c]Department of Mathematical Sciences, Florida Atlantic University,
777 Glades Road, Boca Raton, FL 33431, USA,
rsteinwa@fau.edu

March 1, 2018

Abstract

Password-authenticated key exchange (PAKE) protocols allow users sharing a password to agree upon a high entropy secret. In this paper, a provably secure password-authenticated protocol for group key establishment in the common reference string (CRS) model is presented. Our protocol is quite efficient, as regardless of the number of involved participants it can be implemented with only three communication rounds. We use a (by now classical) trick of Burmester and Desmedt for deriving group key exchange protocols using a two-party construction as main building block. In our case, the two party PAKE used as a base is a one-round protocol by Katz and Vaikuntanatan, which in turn builds upon a special kind of smooth projective hash functions (KV-SPHFs). As evidenced by Benhamouda et al., KV-SPHFs can be instantiated on Cramer-Shoup ciphertexts, thus yielding very efficient (and pairing free) constructions.

Group Key Exchange, Password Authentication, Smooth Projective Hashing

1 Introduction

In distributed applications, low-entropy passwords are still a dominating tool for authentication. Reflecting this, significant research efforts are currently devoted to the exploration of password-authenticated key establishment protocols, through which participants sharing initially a short password aim at agreeing upon a high entropy secret for securing subsequent communication. In this contribution we focus on group key establishment involving $n \geq 2$ users. In the password setting, different scenarios can be considered depending on the application context. E. g., it can be

* contact author

plausible to assume that a dedicated server is available, and each user has an individual password shared with this server. A different scenario does not involve a server, and assumes all users involved in the key establishment to share a common password. In this paper we consider the latter approach. It seems well-suited for small user groups without a centralized server or for applications where the legitimate protocol participants are devices controlled by a single human user.

Several group key establishment protocols for such a scenario have been proposed, including [1, 2, 3, 4]. Many of these initial constructions are based on the random oracle or the ideal cipher model, while few constructions rely on standard assumptions [5, 6]. Aiming at eluding idealized assumptions, we presented in [7] a provably secure password-authenticated protocol for group key establishment in the common reference string (CRS) model. This work can be considered as a refined and corrected final version of [7], yet here we use a different 2-party protocol as main building block. Namely, instead of using the protocol by Gennaro and Lindell from [8], we build upon a one-round construction by Katz and Vaikuntanatan [9] that can (as proven in [10]) be implemented from Cramer-Shoup ciphertexts.

1.1 Key establishment: from 2-party to group

The design of key establishment protocols for two participants has been extensively studied during the last decades, both in the password setting [11, 12, 13, 14] or using stronger authentication means (signatures) [15, 16, 17]. A standard strategy for breaking down the design task of a group key establishment into conceptually simpler steps are protocol compilers that build on the security of a given 2-party solution. Indeed, a number of such generic constructions have been discussed in the literature, including [18, 19, 20]. Many of these compilers are inspired in the classical construction of Burmester and Desmedt [18], where the trick of establishing a group key from pairwise agreed keys among the group principals was first introduced. We sketch their idea in Figure 1, where AKE stands for two-party authenticated key exchange protocol, and thus $\mathsf{AKE}(A, B)$ denotes the execution of AKE by users A and B.

The Burmester-Desmedt trick goes as follows: assume participants to be arranged in a cycle. In a first round, each participant exchanges two-party keys with his left and right neighbour. Once these two-party key establishments have been completed, each participant broadcasts the XOR-value (or the quotient) of the two keys he shares with his neighbors. This allows everyone in the cycle to retrieve each of the 2-party keys that have been exchanged previously, from which a shared session key may be derived. Intuitively, if an adversary has not been able to compromise the security of any of the 2-party protocol executions involved, neither will he be able to retrieve any information about the resulting group session key (for XORs of "randomly looking" elements should look as well random to him), even though some precautions must be taken in order to prevent him mixing up the messages exchanged in the last rounds.

Remarkably, whether designed following the above idea or built from scratch, most group key exchange protocols that can be found in the literature rely on high-entropy secrets for achieving security against active adversaries. Using instead password-based authentication, the first such construction in the standard model is due to Abdalla et al. [21, 22] where a 2-party solution is extended to the 3-party case. Shortly after, in [6], a group protocol is introduced. Further, in [5] a generic compiler that enables the derivation of an authenticated group key establishment protocol from an arbitrary authenticated 2-party key establishment is proposed. This compiler is indeed

Phase 1: Two-party key establishments.

AKE: All indices are to be taken in a cycle, i. e., $U_{n+1} = U_1$, etc.
For $i = 1, \ldots, n$, execute $\mathsf{AKE}(U_i, U_{i+1})$ —as a result, each user U_i holds two keys $\overrightarrow{K}_i$, $\overleftarrow{K}_i$ shared with U_{i+1} respectively U_{i-1}.

Phase 2: Group key establishment.

Computation: Each U_i computes

$$X_i := \overrightarrow{K}_i \oplus \overleftarrow{K}_i$$

Broadcast: Each U_i broadcasts (U_i, X_i)

Computation: Each U_i sets $K_i := \overleftarrow{K}_i$ and computes the $n-1$ values

$$K_{i-j} := \overleftarrow{K}_i \oplus X_{i-1} \oplus \cdots \oplus X_{i-j} \quad (j = 1, \ldots, n-1),$$

defines a master key

$$K := (K_1, \ldots, K_n)$$

and extracts a session key sk_i from K.

Figure 1: Burmester-Desmedt construction: high-level idea

inspired in the above Burmester-Desmedt rationale, but adds extra features and attains very strong security guarantees. In particular, from a password-authenticated 2-party key establishment the compiled protocol is a password-authenticated group key establishment, that has thus been derived without adding idealizing assumptions or high-entropy secrets for authentication. The construction suggested in [5] builds on the use of non-interactive and non-malleable commitments, which in the CRS model are known to be implementable through IND-CCA2 secure encryption schemes.

1.2 Our Contribution. Paper Outline.

In our work [7], we aimed at a neat combination of the 2-party protocol of Gennaro and Lindell [23, 8] and the compiler from [5]. The two-party protocol of Gennaro and Lindell [8, 23] stems from an earlier proposal of Katz et al. [12] and can be proven secure in the CRS model. Their main technical tools are a special type of smooth projective hash functions, which we will refer to as GL-SPHFs.

Smooth projective hash functions (SPHFs) were first introduced by Cramer and Shoup in [24] as a valuable cryptographic primitive for deriving provable secure encryption schemes, and have later proven useful in many other scenarios. Our goal in [7] was to describe a password-based constant-round group key establishment protocol with strong provable security properties which neither used the random oracle nor the ideal cipher model.[1] Unfortunately, the security properties

[1] In independent work, Abdalla et al. [6] followed a different approach aiming at the same goal.

we aimed at in [7] cannot be achieved using the definition of smooth projective hash functions included therein (which was the original by Cramer and Shoup, and will be later referred to as CS-SPHFs). The technical problem from [7] can however be circumvented using a stronger type of SPHF, introduced by Katz and Vaikuntanathan in [9] for building a one-round password based two party key exhange protocol.[2] We thus propose in this work a construction that can be taken for a generalization of Katz and Vaikuntanathan's scheme to a group setting, compiled through the ideas of [5] which in turn build on the Burmester-Desmedt rationale.

The three-round protocol we propose considers a fully asynchronous network with an active adversary. The theoretical model underlying our proof is basically adapted from [12, 25], building in turn on [26, 27]. In the subsequent section we recall the basic components of the security framework, addressing specifics of password-based authentication. Thereafter, in Section 3 we describe the basic tools needed for our construction: smooth projective hashing, labeled public-key encryption and and non-malleable commitments. Finally, in Section 4 we present our password-authenticated constant-round protocol for group key establishment along with a security proof in the CRS model.

2 Security Model and Security Goals

We assume that a common reference string CRS is available that, similarly as in [23], encodes

i) the information needed for implementing a non-malleable commitment scheme,

ii) a uniformly at random chosen element from a family of universal hash functions,

iii) and two values v_0, v_1 that will serve as input for a pseudorandom function.

Also, a dictionary $\mathcal{D} \subseteq \{0,1\}^*$ is assumed to be publicly known. We model the dictionary $\mathcal{D}$ to be efficiently recognizable and of constant or polynomial size. In particular, we must assume that a polynomially bounded adversary is able to exhaust $\mathcal{D}$. The polynomial-sized set $\mathcal{U} = \{U_1, \ldots, U_n\}$ of users is assumed to share a common password $pw \in \mathcal{D}$. Further users, not contained in $\mathcal{U}$ and not knowing the shared password, can be simulated by the adversary. For the sake of simplicity, we adopt the common assumption that pw has been chosen uniformly at random from $\mathcal{D}$, therewith slightly simplifying the formalism.

2.1 Communication Model and Adversarial Capabilities

Users are modeled as probabilistic polynomial time (ppt) Turing machines.[3] Each user $U \in \mathcal{U}$ may execute a polynomial number of protocol *instances* in parallel. To refer to instance s_i of a user $U_i \in \mathcal{U}$ we use the notation $\Pi_i^{s_i}$ $(i \in \mathbb{N})$.

Protocol instances A single instance $\Pi_i^{s_i}$ can be taken for a process executed by U_i. To each instance we assign seven variables:

[2] We will consistently refer to these functions as KV-SPHFs.

[3] All our proofs hold for both uniform and non-uniform machines.

$\mathsf{used}_i^{s_i}$ indicates whether this instance is or has been used for a protocol run. The $\mathsf{used}_i^{s_i}$ flag can only be set through a protocol message received by the instance due to a call to the Send-oracle (see below);

$\mathsf{state}_i^{s_i}$ keeps the state information needed during the protocol execution;

$\mathsf{term}_i^{s_i}$ shows if the execution has terminated;

$\mathsf{sid}_i^{s_i}$ denotes a possibly public session identifier that can serve as identifier for the session key $\mathsf{sk}_i^{s_i}$;

$\mathsf{pid}_i^{s_i}$ stores the set of identities of those users that $\Pi_i^{s_i}$ aims at establishing a key with—including U_i himself;[4]

$\mathsf{acc}_i^{s_i}$ indicates if the protocol instance was successful, i. e., the user accepted the session key;

$\mathsf{sk}_i^{s_i}$ stores the session key once it is accepted by $\Pi_i^{s_i}$. Before acceptance, it stores a distinguished NULL value.

For more details on the usage of the variables we refer to the work of Bellare et al. in [27].

Communication network Arbitrary point-to-point connections among the users are assumed to be available. The network is non-private, however, and fully asynchronous. More specifically, it is controlled by the adversary, who may delay, insert and delete messages at will.

Adversarial capabilities We restrict to ppt adversaries. The capabilities of an adversary $\mathcal{A}$ are made explicit through a number of *oracles* allowing $\mathcal{A}$ to communicate with protocol instances run by the users:

$\mathsf{Send}(U_i, s_i, M)$ This sends message M to the instance $\Pi_i^{s_i}$ and returns the reply generated by this instance. If $\mathcal{A}$ queries this oracle with an unused instance $\Pi_i^{s_i}$ and M being the string "Start", the $\mathsf{used}_i^{s_i}$-flag is set, and the initial protocol message of $\Pi_i^{s_i}$ is returned.

$\mathsf{Execute}(\{\Pi_{u_1}^{s_{u_1}}, \ldots, \Pi_{u_\mu}^{s_{u_\mu}}\})$ This executes a complete protocol run among the specified unused instances of the respective users. The adversary obtains a transcript of all messages sent over the network. A query to the Execute oracle is supposed to reflect a passive eavesdropping. In particular, no online-guess for the secret password can be implemented with this oracle.

$\mathsf{Reveal}(U_i, s_i)$ yields the session key $\mathsf{sk}_i^{s_i}$.

$\mathsf{Test}(U_i, s_i)$ Only one query of this form is allowed for an active adversary $\mathcal{A}$. Provided that $\mathsf{sk}_i^{s_i}$ is defined, (i. e., $\mathsf{acc}_i^{s_i} = \mathsf{true}$ and $\mathsf{sk}_i^{s_i} \neq \text{NULL}$), $\mathcal{A}$ can execute this oracle query at any time when being activated. Then with probability 1/2 the session key $\mathsf{sk}_i^{s_i}$ and with probability 1/2 a uniformly chosen random session key is returned.

[4] Dealing with authentication through a shared password exclusively, we do not consider key establishments among strict subsets of $\mathcal{U}$. With $\mathsf{pid}_i^{s_i} := \mathcal{U}$ being the only case of interest, in the sequel we do not make explicit use of $\mathsf{pid}_i^{s_i}$ when defining partnering, integrity, etc.

2.2 Correctness, Integrity and Secrecy

Before we define correctness, integrity and secrecy, we introduce *partnering* to express which instances are associated in a common protocol session.

Partnering We adopt the notion of partnering from [28]. Namely, we refer to instances $\Pi_i^{s_i}$, $\Pi_j^{s_j}$ as being *partnered* if both $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ and $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$.

To avoid trivial cases, we assume that an instance $\Pi_i^{s_i}$ always accepts the session key constructed at the end of the corresponding protocol run if no deviation from the protocol specification occurs. Moreover, all users in the same protocol session should come up with the same session key, and we capture this in the subsequent notion of correctness.

Correctness We call a group key establishment protocol $\mathcal{P}$ *correct*, if in the presence of a passive adversary $\mathcal{A}$—i. e., $\mathcal{A}$ must not use the Send oracle—the following holds: for all i, j with both $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ and $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$, we have $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j} \neq \text{NULL}$.

Key integrity While correctness takes only passive attacks into account, *key integrity* does not restrict the adversary's oracle access: a correct group key establishment protocol fulfills *key integrity*, if with overwhelming probability all instances of users that have accepted with the same session identifier $\mathsf{sid}_j^{s_j}$ hold identical session keys $\mathsf{sk}_j^{s_j}$. Next, for detailing the security definition, we will have to specify under which conditions a Test-query may be executed.

Freshness A Test-query should only be allowed to those instances holding a key that is not for trivial reasons known to the adversary. To this aim, an instance $\Pi_i^{s_i}$ is called *fresh* if the adversary never queried $\mathsf{Reveal}(U_j, s_j)$ with $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ being partnered.

The idea here is that revealing a session key from an instance $\Pi_i^{s_i}$ trivially yields the session key of all instances partnered with $\Pi_i^{s_i}$, and hence this kind of "attack" will be excluded in the security definition.

Security/key secrecy Because of the polynomial size of the dictionary $\mathcal{D}$, we cannot prevent an adversary from correctly guessing the shared secret $pw \in \mathcal{D}$ with non-negligible probability. Our goal is to restrict the adversary $\mathcal{A}$ to online-verification of password guesses. For a secure group key establishment protocol, we have to impose a corresponding bound on the adversary's *advantage*: The advantage $\mathsf{Adv}_{\mathcal{A}}(\ell)$ of a ppt adversary $\mathcal{A}$ in attacking protocol $\mathcal{P}$ is a function in the security parameter ℓ, defined as

$$\mathsf{Adv}_{\mathcal{A}} := |2 \cdot \mathsf{Succ} - 1|.$$

Here Succ is the probability that the adversary queries Test on a fresh instance $\Pi_i^{s_i}$ and guesses correctly the bit b used by the Test oracle in a moment when $\Pi_i^{s_i}$ is still fresh.

Now, to capture key secrecy we follow an approach of [23]. The intuition behind the definition is that the adversary must not be able to test (online) more than one password per protocol instance. This approach is stricter than the one taken in [2] in the sense that we do not tolerate a constant number > 1 of online guesses per protocol instance:

Definition 2.1 *A password-authenticated group key establishment protocol $\mathcal{P}$ provides* key secrecy*, if for every dictionary $\mathcal{D}$ and every ppt adversary $\mathcal{A}$ querying the* Send*-oracle with at most q different protocol instances, the following inequality holds for some negligible function* $\mathrm{negl}(\ell)$*:*

$$\mathsf{Adv}_{\mathcal{A}} \leq \frac{q}{|\mathcal{D}|} + \mathrm{negl}(\ell)$$

3 Smooth projective hashing and CCA-labeled encryption

Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup [24], and have resulted in a central tool for several provable secure constructions, including [24, 23, 29, 30, 31, 10, 32]. Informally, consider a family of functions $\{H_k : X \longrightarrow \mathbb{G}\}_{k \in K}$ indexed by a countable set K, which acts on the elements of a set X, for a given group $\mathbb{G}$, both being finite. Now, given a distinguished language L defined over X, the above family defines a SPHF for (X, L) if there are four efficient algorithms, which in turn allow for selecting a hashing key k, computing a projection $\alpha(k)$ of it, and evaluating H_k on any $x \in X$ either from the hashing key k or from a tuple $(w, \alpha(k))$ where w is a witness that evidences $x \in L$.

As a correctness requirement, indeed whenever w is a valid L-witness for x, the hashing values computed directly from k and from $(w, \alpha(k))$ must coincide. However, the smoothness property implies that if x is not in L, $H_k(x)$ must be statistically indistinguishable from an element selected uniformly at random in the range of $\mathbb{G}$ even knowing the projection key $\alpha(k)$. There have been different definitions for SPHF, depending essentially on:

- how the projection key $\alpha(k)$ is defined: adaptively (depending both on k and x) or non adaptively (depending only on k);
- whether x may be computed adaptively from $\alpha(k)$.

In this work, we will stick to the definition of Katz and Vaikuntanathan [9], for which $\alpha(k)$ will only depend on the hash key k and yet the word x may be chosen from $\alpha(k)$ in an adaptive way. In the sequel, we introduce the main notions needed for our construction following essentially [9], therefore not aiming at full generality. Most definitions in this section are verbatim taken from [9].

We start by stating what we mean by a labeled public-key encryption scheme, which fits applications in which both plaintexts and ciphertexts are tagged by labels in a consistent manner (see [33]). The different security notions for public-key encryption can easily be adapted for labeled schemes (see, for instance [6]). Informally, the main point for adapting security definitions is to modify the challenge construction phase assuming the adversary must provide not only two plaintexts m_0 and m_1, but also a label l. As a result, when considering CCA security, he will not be allowed to query his decryption oracle on any pair (label, c) where c has been output by the encryption oracle on an input involving the label l.

Definition 3.1 (Labeled PKE) *A* labeled public-key encryption scheme *is a tuple of probabilistic polynomial time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *such that*

- Gen*, the key-generation algorithm, takes as input a security parameter 1^n and returns a pair of (public, secret) keys (pk, sk). This is denoted by $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.*

- Enc, *the encryption algorithm, takes as input a public key* pk, *a label* label *and a plaintext* m *and returns a ciphertext* C. *We write:* $C \leftarrow \mathsf{Enc}(pk, \mathsf{label}, m)$, *or* $C \leftarrow \mathsf{Enc}(pk, \mathsf{label}, m, r)$, *if we want to make explicit the randomness* r *possibly involved in the computation.*

- Dec, *the decryption algorithm, takes as input a secret key* sk, *a label* label *and a ciphertext* C, *and returns a plaintext* m *or an error message* $\perp$. *In symbols;* $m \leftarrow \mathsf{Dec}(sk, \mathsf{label}, C)$.

Moreover, we assume that for all (pk, sk) *output by* $\mathsf{Gen}(1^n)$, *any* label, *any plaintext* m, *and any* C *output by* $\mathsf{Enc}(pk, \mathsf{label}, m)$, *it holds* $\mathsf{Dec}(sk, \mathsf{label}, C) = m$.

At this, plaintexts, ciphertexts, and labels are assumed to be bitstrings of length polynomial in n. *The plaintext and label spaces are supposed to be implicitly defined (and may depend on the public key).*

Building on a CCA-secure labeled encryption scheme, as in [9], we define a *hard subset-membership problem* that will serve as a basis for the SPHF which is the main tool of our construction. We refer to [8] for the general definition of hard subset-membership problems.

Let $\mathcal{D} \subseteq \{0,1\}^*$ be a fixed dictionary and $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ a CCA-secure labeled encryption scheme. For any given public key output by Gen, pk, we denote by C_{pk} the set of valid pairs (label, C) with respect to the public key pk (thus, C is a ciphertext), and assume this set to be efficiently recognizable. Now, consider:

- $X = \{(\mathsf{label}, C, pw) \mid (\mathsf{label}, C) \in C_{pk} \text{ and } pw \in \mathcal{D}\}$
- $L_{pw} = \{(\mathsf{label}, \mathsf{Enc}(pk, \mathsf{label}, pw), pw) \text{ where } \mathsf{label} \in \{0,1\}^*\}$.

Note that $L = \bigcup_{pw \in \mathcal{D}} L_{pw} \subseteq X$. As the encryption scheme is CCA, it can be proven that (X, L) define a hard subset-membership problem: for any probabilistic polynomial-time $\mathcal{A}$, he can only win with negligible probability the following experiment:

- `Phase 1: Setup.` $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and $b \leftarrow \{0,1\}$ is selected uniformly at random. The adversary is given the public key and is also granted access to a b-encryption oracle, which, on any input $(\mathsf{label}, m_0, m_1)$, with m_0 and m_1 of the same bit size, will output $\mathsf{Enc}(pk, \mathsf{label}, m_b)$.
- `Phase 2: Challenge.` $\mathcal{A}$ selects two passwords pw_0 and pw_1 from $\mathcal{D}$ (assumed to be of the same bit size), and a label label. He is presented with an encryption $C \leftarrow \mathsf{Enc}(pk, \mathsf{label}, pw_b)$.
- `Phase 3: Output.` On top of the b-encryption oracle, $\mathcal{A}$ has now access to a decryption oracle holding the secret key sk, which he cannot query with the input (label, C). He outputs a guess b' for the bit b.

Now, the above ingredients will be used to define a $\mathsf{KV\text{-}SPHF}$ for any given public key pk.

Definition 3.2 ($\mathsf{KV\text{-}SPHF}$) *Let* X *be a non-empty set,* $\mathbb{G}$ *be a group and* K *some index set (all finite). Consider a family* $H = \{H_k : X \longrightarrow \mathbb{G}\}_{k \in K}$ *of mappings from* X *into* $\mathbb{G}$, *and let* $\alpha : K \longrightarrow S$ *be a map from* K *into some finite non-empty set* S *(which may be seen as a projection). Given a subset* $L \subseteq X$, *we refer to the tuple* $\mathbf{H} = (H, K, X, L, G, S, \alpha)$, *as* smooth projective hash family *(*SPHF*) for* (X, L) *if there are efficient algorithms*

- HashKG*: selects uniformly at random a hash key* $k \in K$
- $\mathsf{ProjKG}(k)$*: computes a projection key* $\alpha(k)$,
- $\mathsf{Hash}(k, x)$*: outputs* $H_k(x)$ *computed from the hashing key* k.
- $\mathsf{ProjHash}(\alpha(k), x, w)$*: outputs* $H_k(x)$ *from the projection key* $\alpha(k)$*, provided that* w *is a valid witness evidencing* $x \in L$.

Moreover, for any function $f : S \longrightarrow X \setminus L$*, the following distributions have statistical difference negligible in the security parameter* n :

$$\{k \leftarrow \mathsf{HashKG}, s \leftarrow \mathsf{ProjKG}(k) : (s, H_k(f(s)))\}$$

and

$$\{k \leftarrow \mathsf{HashKG}, s \leftarrow \mathsf{ProjKG}(k), g \leftarrow G : (s, g)\}.$$

Remark 3.3 *As neatly explained in [10], the main difference between the above definition of* smoothness *and previous ones is that*

- KV-SPHF *[9]: the projection key does not depend on the word* C *and furthermore the smoothness condition holds even if* C *is constructed knowing* $\alpha(k)$,
- CS-SPHF*[24]: the projection key* $\alpha(k)$ *does not depend on* C*, but* C *must not depend on* $\alpha(k)$,
- GL-SPHF*[8]:* $\alpha(k)$ *may depend on* C.

In [10], a new KV-SPHF is constructed from labeled Cramer-Shoup encryption. Recall that KV-SPHFs were designed with the goal of achieving one-round PAKE. In order to do with just one round, the ciphertext and the projection key for verifying the correctness of the partner's ciphertext should be sent together, and thus be independent. Moreover, the smoothness property must hold in a scenario where the adversary can wait until it receives the partner's projection key before generating the ciphertext.

Let us go back to the concrete instance of a hard subset membership problem as explicited above. At this, note that $\mathsf{ProjHash}(\alpha(k), \mathsf{label}, C, pw, r)$ will output $H_k(\mathsf{label}, \mathsf{C}, \mathsf{pw})$ if and only if r is a valid witness of (label, C, pw), namely, if and only if $C \leftarrow \mathsf{Enc}(pk, \mathsf{label}, pw, r)$. We will make use of the following technical lemma taken from [9], which in turn is a refinement of Lemma 3.4 of [34]. It roughly states that seeing many projection keys will not help in distinguishing (properly constructed) hashes from elements selected at random from $\mathbb{G}$, if appropiate witnesses are not known.

Lemma 3.4 *Let* $\mathsf{LENC} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a CCA-labeled public-key encryption scheme,* $\rho = \rho(n)$ *be a fixed polynomial function and* $\mathcal{A}$ *a probabilistic polynomial time adversary. For* $b \in \{0, 1\}$*, we define the experiment* Exp_b *as*

- *`Phase 1: Setup.` Execute* $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$*, fix* $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ *a smooth projective hash function for* pk *as above and forward this public key to* $\mathcal{A}$

- *Phase 2: Challenge. Execute* HashKG ρ *times, retrieving as output* $k_1, \ldots, k_\rho$ *which are in turn fed to* ProjKG$(\cdot)$. *Feed the corresponding outputs* $s_1, \ldots, s_\rho$ *to* $\mathcal{A}$.
- *Phase 3: Output. During this phase,* $\mathcal{A}$ *is granted access to two oracles:*
  - $\mathcal{O}_{\mathsf{Enc}}$*: a modified encryption oracle, which on input* (label, pw) *for any* $pw \in \mathcal{D}$ *outputs*
    - *If* $b = 0$*:* $H_{k_i}(\mathsf{label}, C, pw)$, *for* $i = 1, \ldots, \rho$, *where* $C \leftarrow \mathsf{Enc}(pk, \mathsf{label}, pw)$,
    - *Else, if* $b = 1$*:* ρ *values selected uniformly at random from* $\mathbb{G}$.
  - $\mathcal{O}_{\mathsf{Dec}}$ *: a CCA-decryption oracle for* LENC, *namely, this oracle may not be queried with any pair* (label, C) *where* C *was obtained from the encryption oracle on query* label, pw.

  At the end of this phase, $\mathcal{A}$ *outputs her guess* b'.

Then, $|2\Pr[b = b'] - 1|$ *is negligible in the security parameter.*

The previous lemma thus states that distinguishing between the two experiments Exp_0 and Exp_1 defined above is *hard*, thus hashes and random group elements are hard to distinguish even when having access to many projections.

4 A Group Key Establishment Protocol

The protocol we propose builds on a CCA-labeled encryption scheme and a KV-SPHF $H = \{H_k\}_{k \in K}$ as described in the previous section. In particular, we assume the image of the hash functions H_k to be contained in a finite Abelian group $\mathbb{G}$. Furthermore, we will use a family of universal hash functions $\mathcal{UH}$ that maps elements from G^n onto a superpolynomial-sized set $\{0,1\}^L$, and a family of universal hash functions $\mathcal{UH}'$ that map elements from $\mathbb{G}$ onto a superpolynomial sized set T of cardinality $|T| \leq \sqrt{|\mathbb{G}|}$. Similarly as Bresson et al. [1], we impose an additional restriction on $\mathcal{UH}'$, saying that there are no "bad indices" into the family $\mathcal{UH}'$. Namely, for *each* $\mathrm{UH}' \in \mathcal{UH}'$ we require the following to hold: any ppt algorithm having UH' as input, has no more than a negligible probability to predict $\mathrm{UH}'(g)$ for an (unknown) uniformly at random chosen $g \in \mathbb{G}$.

Example 1 *Let* $\mathbb{G} := \mathbb{Z}/p\mathbb{Z}$ *be the additive group of integers modulo an* ℓ*-bit prime* p*, and let* $L' := \lfloor \ell/2 \rfloor$. *Choosing* $T := \{0,1\}^{L'}$ *to be the set of bitstrings of length* L'*, the following family* $\mathcal{UH}'$ *considered by Carter and Wegman [35] contains no "bad indices":*

$$\mathcal{UH}' := \{g \mapsto [a \cdot g + b]_{0 \to L'-1} : a, b \in \mathbb{Z}/p\mathbb{Z} \text{ with } a \neq 0\},$$

where $[\,\cdot\,]_{0 \to L'-1}$ *denotes selection of the* L' *least significant bits ("mod* $2^{L'}$*"). The universality of* $\mathcal{UH}'$ *is well-known (cf., e. g., [35]). Moreover, as the case* $a = 0$ *is excluded in the affine maps considered, for a uniformly at random chosen* $g \in \mathbb{G}$*, also* $a \cdot g + b$ *is uniformly at random distributed in* $\mathbb{G}$*, and the probability of predicting the correct value* $\mathrm{UH}'_{a,b}(g) = [a \cdot g + b]_{0 \to L'-1}$ *is negligible.*

The CRS selects one universal hash function UH from the family $\mathcal{UH}$.

A collision-resistant pseudorandom function family We use UH to select an index within a collision-resistant pseudorandom function family $\mathcal{F} = \{F^\ell\}_{\ell\in\mathbb{N}}$ as used by Katz and Shin [36]. We assume $F^\ell = \{F^\ell_\eta\}_{\eta\in\{0,1\}^L}$ to be indexed by $\{0,1\}^L$ and denote by $v_0 = v_0(\ell)$ a publicly known value such that no ppt adversary can find two different indices $\lambda \neq \lambda' \in \{0,1\}^L$ with $F_\lambda(v_0) = F_{\lambda'}(v_0)$ (see [36] for more details). As in [36] we use another public value v_1 (which, like v_0 can be included in the CRS) for deriving the session key. The family $\mathcal{UH}'$ is used for confirming the auxiliary two-party keys $Z_{i,i-1}$ in Round 2 of our protocol without jeopardizing the password pw.

Commitments Round 2 of our protocol uses another essential component already present in Gennaro and Lindell's construction: non-interactive and non-malleable commitment schemes. Roughly speaking, they should fulfill the following requirements:

1. Every commitment c defines at most one value ($\mathsf{decommit}(c)$) (i. e., the scheme must be *perfectly binding*).
2. If an adversary receives several commitments to a value ν, he must not be able to output a commitment to a value β related to ν in a known way (that is, it must achieve *non-malleability for multiple commitments*).

In the common reference string model, the above commitment schemes can be constructed from any public-key encryption scheme that is non-malleable and secure for multiple encryptions (in particular, from any IND-CCA2 secure public-key encryption scheme) (see, for instance, [8]).[5]

Now we are ready to introduce our proposed construction. Our protocol is symmetric in the sense that all users perform the same steps. Figures 2, 3 and 4 show the three rounds of our protocol. For the sake of readability, we do not explicitly refer to instances s_i of users.

5 Design comments

The basic design of the protocol follows the Burmester-Desmedt [18] construction where the Diffie-Hellman key exchanges are replaced by the Katz-Vaikuntanatan one-round protocol [9] key exchange. As in [7], a basic trick of our design is the construction of the master key as

$$\mathsf{mk} = (Z_{1,2}, Z_{2,3}, \ldots, Z_{n-1,n}, Z_{n,1}).$$

The original construction $\mathsf{mk} = \prod_{i=1,\ldots,n} Z_{i,i+1}$ can be determined by two malicious users as pointed out in [28]. Thus, if an adversary guesses the password, he would be able to provoke pathological behaviors such that each protocol run ends up with exactly the same mk (and thus, identical sid_i, sk_i). Note that with the construction of mk proposed above, both sid_i and sk_i will be indistinguishable from random if a sole honest user is involved in the protocol run.

Let us further comment a bit about Round 2. At this stage, the idea is to broadcast commitments to the quotients X_i. As in [7], this is to prevent an online attack on the protocol consisting

[5] Actually, the encrypted values $c_{i,i+1}, c_{i,i-1}$ from Round 1 could as well have been defined as commitments constructed from such a commitment scheme. We preferred the encryption formulation to keep our formulation closer to that of [9]

Round 1:

`Broadcast:` Each U_i

- chooses uniformly at random k_i from K;
- derives a corresponding projection key $S_i = \alpha(k_i)$;
- selects random nonces $r_{i,i+1}, r_{i,i-1}$;
- defines labels $\mathsf{label}_{i,i+1} := (U_i, U_{i+1}, S_i)$ and $\mathsf{label}_{i,i-1} = (U_i, U_{i-1}, S_i)$;
- computes

$$c_{i,i+1} := \mathsf{Enc}(pk, \mathsf{label}_{i,i+1}, pw, r_{i,i+1})$$

and

$$c_{i,i-1} := \mathsf{Enc}(pk, \mathsf{label}_{i,i-1}, pw, r_{i,i-1});$$

- broadcasts $M_i^1 := (U_i, S_i, c_{i,i+1}, c_{i,i-1})$.

`Check:` Each U_i

- waits until messages M_j^1 for all U_j arrived;
- checks if the values $c_{i+1,i}$ and $c_{i-1,i}$ are valid encryptions;[a]
- If any of the checks fails, set $\mathsf{acc}_i := \mathsf{false}$ and terminate.

[a] At this, valid means a valid encryption of pw with the expected pk and label

Figure 2: Round 1: A password-authenticated 3-round protocol for group key establishment.

only of Round 1 and Round 3, that allows to test two passwords using only one instance $\Pi_i^{s_i}$ (see Section 4.1 of [7]).

Further, note that the test_i-values in Round 2 addresses attacks where one party did not receive the correct projection but rather a bogus one, inserted by the adversary. Note that this is needed despite assumptions on the projective hash function, for we have no guarantees if projections are not constructed from randomly selected elements $k \in K$ (for details see Game 4 and Game 5 in the proof of Theorem 5.1). Finally, the random value $X_{i,1}$ is needed if the check of a test_i-value fails. In this case, the true $X_{i,0}$ must not be revealed. On the other hand, an adversary should not recognize if the check fails, as this would give a hint if the respective hash values and therefore a password that the adversary may have used was correct. This is again, to prevent that two passwords may be tested with one instance running the protocol.

5.1 Security Analysis

Theorem 5.1 *With the prerequisites as described above, the protocol depicted in Figures 2, 3 and 4 is correct and achieves key secrecy and key integrity.*

Proof. It is easy to see that the above protocol fulfills correctness and integrity, and the main part of our proof is devoted to key secrecy:

Correctness and Integrity. Owing to the collision-resistance of the family $\mathcal{F}$, all oracles that accept with identical session identifier use with overwhelming probability the same index value $\mathrm{UH}(\mathsf{mk})$ and therewith also derive the same session key.

Round 2:

`Computation:` Each U_i

- sets
$$\mathsf{label}_{i+1,i} := (U_{i+1}, U_i, S_{i+1})$$
and
$$\mathsf{label}_{i-1,i} := (U_{i-1}, U_i, S_{i-1});$$
- derives two party keys:
$$Z_{i,i+1} := H_{k_i}(\mathsf{label}_{i+1,i}, pw, c_{i+1,i}) \cdot H_{k_{i+1}}(\mathsf{label}_{i,i+1}, pw, c_{i,i+1}),$$
$$Z_{i,i-1} := H_{k_{i-1}}(\mathsf{label}_{i,i-1}, pw, c_{i,i-1}) \cdot H_{k_i}(\mathsf{label}_{i-1,i}, pw, c_{i-1,i});$$
- sets $X_{i,0} := Z_{i,i+1} \cdot Z_{i,i-1}^{-1}$;
- chooses a random $X_{i,1} \in G$;
- chooses random values $r'_{i,0}$, $r'_{i,1}$;
- computes commitments $C_\rho(U_i, X_{i,0}; r'_{i,0})$ and $C_\rho(U_i, X_{i,1}; r'_{i,1})$;
- chooses at random $\mathrm{UH}'_i \in \mathcal{UH}'$ and
- computes a test value $\mathsf{test}_i := \mathrm{UH}'_i(Z_{i,i-1})$.

`Broadcast:` Each user U_i broadcasts for a random bit b

$$M_i^2 := (U_i, C_\rho(U_i, X_{i,b}; r'_{i,b}), C_\rho(U_i, X_{i,1-b}; r'_{i,1-b}), \mathsf{test}_i, \mathrm{UH}'_i).$$

`Check:` Each user U_i

- waits until messages M_j^2 for all j arrived
- checks if
$$\mathrm{UH}'_{i+1}(Z_{i,i+1}) = \mathsf{test}_{i+1};$$
- if the check succeeds, set
$$(X_i, r'_i) := (X_{i,0}, r_{i,0}) \text{ otherwise } (X_i, r'_i) := (X_{i,1}, r_{i,1}).$$

Figure 3: Round 2: A password-authenticated 3-round protocol for group key establishment.

Round 3:

`Broadcast:` Each user U_i broadcasts $M_i^3 := (U_i, X_i, r'_i)$.

`Check:` Each U_i

- checks that $X_1 \cdots X_n = 1$;
- checks the correctness of the commitments $C_\rho(U_j, X_j; r'_j)$;
- if at least one of theses checks fails, set $\mathsf{acc}_i := \mathsf{false}$ and terminate.

`Computation:` Each U_i computes the values

$$\begin{aligned} Z_{i-1,i-2} &:= Z_{i,i-1}/X_{i-1} \\ Z_{i-2,i-3} &:= Z_{i-1,i-2}/X_{i-2} \\ &\vdots \\ Z_{i,i+1} &:= Z_{i+1,i+2}/X_{i+1}, \end{aligned}$$

a master key

$$\mathsf{mk} := (Z_{1,2}, Z_{2,3}, \ldots, Z_{n-1,n}, Z_{n,1}),$$

and sets $\mathsf{sk}_i := F_{\mathrm{UH}(\mathsf{mk})}(v_1)$, $\mathsf{sid}_i := F_{\mathrm{UH}(\mathsf{mk})}(v_0)$ and $\mathsf{acc}_i := \mathsf{true}$.

Figure 4: Round 3: A password-authenticated 3-round protocol for group key establishment.

Key Secrecy. We imagine a simulator that simulates the oracles and instances for the adversary. The proof is set up in terms of several experiments or games, where from game to game the simulator's behavior somehow deviates from the previous. Following standard notation, we denote by $\mathsf{Adv}(\mathcal{A}, G_i)$ the advantage of the adversary when confronted with Game i. The security parameter is denoted by ℓ. Furthermore, we will index the Send oracle, denoting by Send_0 the Send query that initializes a protocol run and by Send_i a Send query that delivers a message of round i for $i \in \{1, 2, 3\}$. As we must consider the session identifiers known to the adversary, we assume them to be part of the output of the final Send_3 query.

For the sake of readability, we start by sketching an informal *proof roadmap* here:

– Game 0 is, as usual, modelling the real experiment faced by the adversary.

– Game 1 to Game 3 deal with the case of passive adversaries; thus, they progressively modify the $\mathsf{Execute}$ oracle: in Game 1 the two-party keys $Z_{i,j}$ are replaced by random bitstrings, then in Game 2 the "real" password is substituted by another one and finally the session key is chosen in Game 3 uniformly at random. The adversary is unable to notice these steps, due to the hiding property of the commitment scheme, the semantic security of the encryption scheme and the fact that the values $Z_{i,j}$ and X_i look anyway random to him.

Games 4 to 8 deal with adversaries that modify messages in Round 1. For all possible modifications, the recipient of the bogus message will randomize its value X_i, or the game is aborted. Also for correct messages, the values $Z_{i,i-1}$ and $Z_{i,i+1}$ are randomized. Table 1 gives an overview which game deals with which modifications caused by the adversary $\mathcal{A}$.

Game	S_{i-1}	S_{i+1}	c_{i-1}	c_{i+1}
4	replaced	$*$	oracle-generated	oracle-generated
5	$*$	replaced	oracle-generated	oracle-generated
6	✓	✓	oracle-generated	oracle-generated
7	$*$	$*$	invalid	$*$
	$*$	$*$	$*$	invalid
8	$*$	$*$	valid from $\mathcal{A}$	valid from $\mathcal{A}$

Table 1: Handling modified Round 1 messages in the proof of Theorem 5.1.

– Game 4 and 5 are concerned with the situation in which the adversary may insert projections in the first round. A malicious insertion of S_{i-1} results in U_i choosing $Z_{i,i-1}$ uniformly at random; in Game 5, if U_i gets an adversarially sent S_{i+1} the corresponding X_i is chosen uniformly at random.

  The adversary will not notice these changes in the simulation. In Game 4 the argument follows because inserting a projection will not help him distinguishing the $Z_{i,j}$ from values selected independently and uniformly at random, and thus messages from Round 2 will not help him detect the change. Furthermore, in both games the messages from Round 3 will not help the adversary in distinguishing. The adversary cannot prevent that with overwhelming probability the check in Round 2 will fail and thus in Round 3 uniform random values $X_{i,1}$ will be broadcast.

– Game 6. Once ruled out the possibility of inserted projections, the simulator will now generate the two-party keys $Z_{i,j}$ independently and uniformly at random if the encryptions in round one were oracle-generated, i. e., honestly transmitted or replayed from other instances. Distinction between this game and Game 5 reduces to distinguishing between Exp_0 and Exp_1 from Lemma 3.4.

  In the sequel, we will speak of "valid encryptions" to refer to encryptions of the correct pw with the public key and label as expected.

– Game 7 deals with the case in which the adversary may insert an invalid encryption in Round 1. The simulator, detecting an invalid encryption, will choose $X_{i,0}$ at random. This modification is due to the smoothness property not detectable by the adversary from the messages exchanged.

– Game 8 deals with the case of valid encryptions c_i generated by the adversary, in which case he wins. This corresponds to a correct guess for the password.

– Game 9 aborts in case any encryption, commitment, projection or X_i-value is inserted by the adversary. The advantage of the adversary can only vary negligibly, as due to the non-malleability of the commitment scheme and the condition $X_1 \cdots X_n = 1$, the protocol would anyway abort with overwhelming probability.

– Game 10 and 11 argue similarly as in the passive case, once all malicious Send-queries are ruled out. First, in Game 10, encryptions from Round 1 are constructed using a randomly selected password. To conclude, the key generation is modified in Game 11 in that the session key is chosen uniformly at random. The adversary can only win by having inserted a valid commitment he constructed; otherwise he will not be able to tell the difference, given that UH is a universal hash function and (at least) one of its inputs $Z_{i,k}$ is a random group element. This concludes the proof.

Having outlined the structure of the proof, we are left to fill in the details:

Game 0. All oracles are simulated as defined in the model. Thus, $\mathsf{Adv}(\mathcal{A}, G_0)$ is exactly $\mathsf{Adv}(\mathcal{A})$.

Game 1. In this game, the simulation of the $\mathsf{Execute}$ oracle is modified. Instead of computing the values $Z_{i,i-1}, Z_{i,i+1}$ for $i = 1, \ldots, n$ as specified in the protocol, they are chosen uniformly at random from $\mathbb{G}$. As a result, also the values X_i will be random though fulfill the property $X_1 \cdots X_n = 1$ and the master key mk will be a randomly selected element from $\mathbb{G}^n$.

Let us now reason that the probability an adversary has of distinguishing between the values X_i generated in Game 0 and the ones generated in Game 1 is no greater than the probability he has of distinguishing the experiments Exp_0 and Exp_1 from Lemma 3.4. Indeed, for a fixed common reference string and password the adversary cannot distinguish between Exp_0 and Exp_1, for $i = 1, \ldots, n$. That means, seeing values $c_{i,i-1}, c_{i,i+1}$ and the projection $\alpha(k_i)$, he cannot tell $H_{k_i}(\mathsf{label}_{\mathsf{i+1,i}}, \mathsf{pw}, \mathsf{c}_{\mathsf{i,i-1}})$ and $H_{k_i}(\mathsf{label}_{\mathsf{i-1,i}}, \mathsf{pw}, \mathsf{c}_{\mathsf{i-1,i}})$ apart from independent random values, thus, the same applies to each element X_i generated in Game 0.

Therefore, having a negligible probability of distinguishing between the two experiments we have

$$|\mathsf{Adv}(\mathcal{A}, G_1) - \mathsf{Adv}(\mathcal{A}, G_0)| \leq \mathrm{negl}(\ell).$$

Game 2. At this, the $\mathsf{Execute}$ oracle is again modified, so that a password $\widehat{pw}$ is chosen uniformly at random from $\mathcal{D}$. Further, each $c_{i,i+1}$ and $c_{i,i-1}$ are computed consistently. Due to the semantic security of the encryption scheme ENC, we again have

$$|\mathsf{Adv}(\mathcal{A}, G_2) - \mathsf{Adv}(\mathcal{A}, G_1)| \leq \mathrm{negl}(\ell).$$

Game 3. Let us consider a further modification of the $\mathsf{Execute}$ oracle. Namely, the simulator will assign to the instances a session key $\mathsf{sk}_i^{s_i} \in \{0,1\}^\ell$, chosen uniformly at random.

Note that even knowing all values X_i, still the value of at least one of the two-party keys $Z_{k,j}$ is indistinguishable from a random group element. Thus, with the leftover hash lemma, we see that the master key $\mathsf{mk} = (Z_{1,2}, \ldots, Z_{n,1})$ has sufficient entropy so that the output of the pseudorandom function $F_{\mathrm{UH}(\mathsf{mk})}$ is distinguishable from a random $\mathsf{sk}_i^{s_i}$ with negligible probability only.

$$|\mathsf{Adv}(\mathcal{A}, G_3) - \mathsf{Adv}(\mathcal{A}, G_2)| \leq \mathrm{negl}(\ell).$$

By now the $\mathsf{Execute}$ oracle returns only random values, independent of the password, and instances used by an $\mathsf{Execute}$-query hold only random session keys. The following games will deal with the Send oracle.

We will in the following call an encryption that was generated by the simulator *oracle-generated* and in accordance an encryption that was generated by the adversary *adversary-generated*. This can be checked efficiently by keeping a list of all encryptions the simulator generates. Furthermore, we call the encryption *valid* if it is indeed an encryption for the password pw and *invalid* otherwise. Note that also encryptions that are replayed by the adversary are *oracle-generated*.

Game 4. In this experiment all encryptions are oracle-generated and the simulator will keep a list for the projections S_i he generated for each user U_i in Round 1. Once an instance $\Pi_i^{s_i}$ has got all messages of the first round, the simulator checks if the received S_{i-1} is consistent with the one generated for the U_{i-1}. In case S_{i-1} was replaced and one of the respective $c_{i,i-1}$ or $c_{i-1,i}$ is *oracle-generated*, the corresponding key $Z_{i,i-1}$ is replaced by a random group element. If $Z_{i,i-1}$ was replaced, U_{i-1} will use $X_{i-1,1}$ in Round 3.

The replacement of $Z_{i,i-1}$ was caused by a replaced projection S_{i-1} and hence the value $H_{k_{i-1}}(\mathsf{label}_{\mathsf{i},\mathsf{i-1}}, pw, c_{i,i-1})$ may be known to the adversary. However, as k_i was honestly generated, $c_{i-1,i}$ oracle-generated, and the projective hash function is smooth the hash $H_{k_i}(\mathsf{label}_{\mathsf{i-1},\mathsf{i}}, pw, c_{i-1,i})$ computed by U_i is indistinguishable from an element chosen independently and uniformly in the group. Therefore $Z_{i,i-1}$ computed by U_i is for the adversary indistinguishable from an independently uniformly at random chosen element. This holds of course only, if U_{i-1} does not release any information about $H_{k_i}(\mathsf{label}_{\mathsf{i-1},\mathsf{i}}, pw, c_{i-1,i})$. Therefore U_{i-1} will use $X_{i-1,1}$ in Round 3.

It is left to show that U_{i-1}'s check of test_i will indeed fail. U_i will randomly choose $\mathrm{UH}_i' \in \mathcal{UH}'$ and compute and broadcast $\mathsf{test}_i = \mathrm{UH}_i'(Z_{i,i-1})$. Neither can the adversary recognize the replacement of $Z_{i,i-1}$ from test_i nor can he produce a test_i that will be accepted by U_{i-1}.

- Even with knowledge of all test_j, $j = 1 \ldots n$, the value $Z_{i,i-1}$ remains indistinguishable from an independently and uniformly at random chosen element in G, because the test_j carry only a negligible amount of information due to $|T| \leq \sqrt{|\mathbb{G}|}$.
- The adversary cannot produce test_i' that would be accepted by U_{i-1}: As UH_i' is chosen independent at random it is independent from $Z_{i-1,i}$ and the adversary has no prior information on the test_i-value expected by U_{i-1}, because $Z_{i,i-1}$ is indistinguishable from random for him. Suppose the adversary tries to insert both test_i' and UH_i''. He will only succeed if he is able to find h and UH_i'' so that $h = \mathrm{UH}_i''(Z_{i-1,i})$, where $Z_{i-1,i}$ is indistinguishable from random for him. By our assumption on the hash family $\mathcal{UH}'$ that there are no "bad indices" into $\mathcal{UH}'$, this is not possible, however. Thus, in either case, the adversary has only a negligible probability of success.

Therefore we have

$$|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_3)| \leq \mathrm{negl}(\ell).$$

Game 5. Again, the commitments are oracle-generated, but this experiment deviates from the previous one in that the simulator also checks if S_{i+1} is consistent with the one generated for the U_{i+1}. In case S_{i+1} was replaced and $c_{i,i+1}$ is *oracle-generated*, U_i will continue with $X_{i,1}$ in Round 3.

We show that U_i's check of test_{i+1} would indeed fail, so that this replacement makes no difference for the adversary. The argument is analogous as above: when U_{i+1} chooses $\mathrm{UH}_{i+1}' \in \mathcal{UH}'$, the adversary is unable to produce a test_{i+1} that will be accepted by U_i, as again the value $Z_{i,i+1}$ computed by U_{i+1} is indistinguishable from random for the adversary. Neither will the adversary

succeed in computing a pair $(\mathrm{UH}''_{i+1}, \mathsf{test}'_{i+1})$ that will convince U_i: due to our assumption on the hash family $\mathcal{UH}'$, the adversary has no a priori information on the test_{i+1} value expected by U_i.

Therefore we have

$$|\mathsf{Adv}(\mathcal{A}, G_5) - \mathsf{Adv}(\mathcal{A}, G_4)| \leq \mathrm{negl}(\ell).$$

Game 6. In this experiment, if the commitments were oracle-generated the simulator chooses the values $Z_{i,i-1}$ and $Z_{i,i+1}$ independently and uniformly at random from the group G. The simulator keeps a list for entries of the form

$$(c_{i,i-1}, S_{i-1}, c_{i-1,i}, S_i) \to Z_{i,i-1},$$

$$(c_{i,i+1}, S_i, c_{i+1,i}, S_{i+1}) \to Z_{i,i+1}.$$

The simulator behaves as in Game 5, except for the answer following a Send_1 query that delivers the last first round message to an instance $\Pi_i^{s_i}$.

Once an instance $\Pi_i^{s_i}$ has received all messages of the first round, the simulator checks if $c_{i-1,i}$ and $c_{i+1,i}$ were both *oracle-generated* (but all projections were unmodified). In this case, the simulator checks if $(c_{i,i-1}, S_{i-1}, c_{i-1,i}, S_i) \to Z_{i,i-1}$ or $(c_{i,i+1}, S_i, c_{i+1,i}, S_{i+1}) \to Z_{i,i+1}$ are already in the list, and uses the according values $Z_{i,i-1}$ respectively $Z_{i,i+1}$ for further computations. The values $Z_{i,i-1}$ or $Z_{i,i+1}$ that are not yet determined by the list are chosen at random from the group $\mathbb{G}$ and the assignment $(c_{i,i-1}, S_{i-1}, c_{i-1,i}, S_i) \to Z_{i,i-1}$ or $(c_{i,i+1}, S_i, c_{i+1,i}, S_{i+1}) \to Z_{i,i+1}$, respectively, is stored in the list to assure consistency between neighbored instances.

Given an adversary $\mathcal{A}$ able to distinguish between Game 5 and Game 6 we can construct a distinguisher D between Exp_0 and Exp_1. Thus, from Lemma 3.4 we can conclude that $\mathcal{A}$'s advantage between the two games differs at most negligibly.

The distinguisher D is either facing Exp_0 or Exp_1 from Lemma 3.4. D is constructed so that it will behave like the simulator from Game 5, except:

- commitments c are not computed but obtained by the $\mathcal{O}_{\mathsf{Enc}}$ oracle from Lemma 3.4 on input pw,
- if a Send-query of the adversary requires D to compute values $Z_{i,i-1}$ respectively $Z_{i,i+1}$, D will output hashes/random values from the respective values $c_{i,i-1}, c_{i-1,i}$ and $c_{i,i+1}, c_{i+1,i}$ if both were *oracle-generated*.

Now the view of $\mathcal{A}$ will be exactly as in Game 5 if D is facing Exp_0 and exactly as in Game 6 if D is facing Exp_1.

$$|\mathsf{Adv}(\mathcal{A}, G_6) - \mathsf{Adv}(\mathcal{A}, G_5)| \leq \mathrm{negl}(\ell).$$

For the following games, the simulator is given an $\mathsf{Extract}$ oracle, that checks if a given encryption is valid, i. e., an encryption of the password pw.

This can be done because the password is information-theoretically contained in the encryption. On input a value c, the $\mathsf{Extract}$ oracle exhausts all possible random choices r to check whether c is a commitment to pw or not. Indeed, the set of possible values r is of superpolynomial size in the security parameter; this is however allowed for the $\mathsf{Extract}$ oracle.

Game 7. In this experiment, the simulator behaves as in Game 6, except that in Round 2's computation phase, following a Send_1 query, the received $c_{i-1,i}$ and $c_{i+1,i}$ are checked by the simulator w.r.t. the password using the $\mathsf{Extract}$ oracle. Then, those instances $\Pi_i^{s_i}$ that received an invalid encryption will choose a random group element for $X_{i,0}$.

By a statistical argument, we see that the probability for the adversary to distinguish between Game 7 and Game 6 is negligible. If in Round 1 an invalid encryption c to a wrong password $\widetilde{pw}$ (that is, $(c, \widetilde{pw}) \notin L_\rho$,) was sent, then by the smoothness property of the hash proof system the distribution of $(c, \widetilde{pw}, \alpha(k), H_k(\mathsf{label}, \widetilde{pw}, c))$ is statistically close to the distribution of $(c, \widetilde{pw}, \alpha(k), g)$ for a random group element $g \in G$. Thus, the respective $Z_{i,i-1}$ or $Z_{i,i+1}$ and therefore $X_{i,0}$ will look like a random group element for the adversary, who thus has only a negligible chance to detect the difference.

As a result,

$$|\mathsf{Adv}(\mathcal{A}, G_7) - \mathsf{Adv}(\mathcal{A}, G_6)| \leq \mathrm{negl}(\ell).$$

By now, only such executions of Round 2 following a Send_1 query are unchanged where the encryptions from the neighboring users are both *valid* and at least one was *adversary-generated*. The following experiments will also modify this situation.

Game 8. Now the simulator will abort the game with a win of the adversary, if an instance $\Pi_i^{s_i}$ received from a Send_1-query *valid* commitments $c_{i-1,i}$ and $c_{i+1,i}$ of which at least one was *adversary-generated*.

This will only increase the success probability of the adversary, therefore:

$$\mathsf{Adv}(\mathcal{A}, G_8) \geq \mathsf{Adv}(\mathcal{A}, G_7).$$

Game 9. In this game, the simulation aborts if the adversary has inserted commitments, projections or X_i-values in Round 3:

Note that in Game 9, if a message in Round 1 to an instance of U_i was modified, as a result U_i individually chooses $Z_{i,i-1}$ or $Z_{i,i+1}$, respectively, uniformly at random. Therefore, U_i holds X_i unknown to anyone and expects commitments to values X_j such that $X_1 \cdots X_n = 1$. Now, in Round 2, U_i outputs a commitment $C_\rho(U_i, X_i; r_i')$ to a value X_i that only with negligible probability fulfills $X_1 \cdots X_n = 1$. To avoid users $U_1, \ldots, U_{i-1}, U_{i+1}, \ldots, U_n$ from aborting, therefore, the adversary needs to be able to construct a commitment $C_\rho(U_i, X_i^*; r^*)$ to a value X_i^* such that $X_1 \cdots X_i^* \cdots X_n = 1$. Again, as all X_j are unknown to the adversary, this can only succeed with negligible probability. Note that moreover, X_i is a random value and only $C_\rho(U_i, X_i; r_i')$ contains information about X_i. Thus, the non-malleability of the commitment scheme gives the adversary only a negligible probability to insert values X_j^* with $j = 1, \ldots, i-1, i+1, \ldots, n$ which would be accepted by $\Pi_i^{s_i}$ in Game 8. The above argument also demonstrates that the adversary cannot insert any value X_i in Round 3 without resulting in an abort (with overwhelming probability). Therefore,

$$|\mathsf{Adv}(\mathcal{A}, G_9) - \mathsf{Adv}(\mathcal{A}, G_8)| \leq \mathrm{negl}(\ell).$$

At this point, we have excluded all situations in which the adversary may have inserted encryptions, commitments, projections or X_i-values in Round 3: either he has guessed the password and inserted valid commitments to it (Game 8) or his attempts to insert rogue messages resulted in a

protocol abortion before the computation of a session key. Thus the only situation left to handle is that all queries to the Send-oracle contain messages faithfully constructed following the protocol specification. Here we can mimic the reasoning for the passive case.

Game 10. Now the simulation changes in that, for constructing the encryptions from Round 1, a password $\widehat{pw}$ is chosen uniformly at random from $\mathcal{D}$. This does not change anything, as in the previous game, these values were not used in any projective hash function anymore. All information about the password, which was available to the adversary, were encryptions sent in the first round. Due to the semantic security of the encryption scheme, the adversary detects the use of $\widehat{pw}$ instead of pw with negligible probability only. Now the adversary does not have any correct encryption of the password as input. But, due to the non-malleability of the encryption scheme, the adversary's probability to succeed in producing a new valid encryption of pw drops at most negligibly.

$$|\mathsf{Adv}(\mathcal{A}, G_{10}) - \mathsf{Adv}(\mathcal{A}, G_9)| \leq \mathrm{negl}(\ell).$$

Game 11. We modify now the computation of the session key. The simulator keeps a list of assignments $(Z_{1,2}, \ldots, Z_{n,1}, \mathsf{sk}_i^{s_i})$. Once an instance receives the last Send_3-query, the simulator computes $Z_{1,2}, \ldots, Z_{n,1}$ and checks if for the sequence $(Z_{1,2}, \ldots, Z_{n,1})$ a master key was already issued and assigns this key to the instance. If no such entry exists in the list, the simulator chooses a session key $\mathsf{sk}_i^{s_i} \in \{0,1\}^\ell$ uniformly at random.

The adversary can only detect the difference, if he knows the master key $\mathsf{mk} = (Z_{1,2}, \ldots, Z_{n,1})$. The master key has once the X_i are public, sufficient entropy because knowing all quotients X_i, still the value of at least one of the two-party keys $Z_{k,j}$ is indistinguishable from a random group element. Therefore the output of the function $F_{\mathrm{UH}(\mathsf{mk})}$ is only with negligible probability distinguishable from a random $\mathsf{sk}_i^{s_i}$.

$$|\mathsf{Adv}(\mathcal{A}, G_{11}) - \mathsf{Adv}(\mathcal{A}, G_{10})| \leq \mathrm{negl}(\ell).$$

Now the session keys are randomly distributed and independent from the password and the messages. Instances that hold the same master key computed the same $\mathrm{UH}(\mathsf{mk})$ and therefore hold identical session identifiers. Thus, those instances are partnered and the freshness definition renders the Reveal-oracle useless because instances that are not partnered have independently uniformly at random chosen session keys. Besides the $1/2$ probability of guessing the bit b right, the only way for the adversary to win is having sent a valid adversary-generated encryption to a neighbored instance that did not get an invalid commitment from the other neighbor. Thus, the adversary has just one try per instance to guess a password and the probability to win in Game 11 is

$$\mathsf{Succ}(\mathcal{A}, G_{11}) = \frac{q}{|\mathcal{D}|} + \frac{1}{2}\left(1 - \frac{q}{|\mathcal{D}|}\right) + \mathrm{negl}(\ell),$$

giving an advantage of

$$\mathsf{Adv}(\mathcal{A}, G_{11}) = \frac{q}{|\mathcal{D}|} + \mathrm{negl}(\ell).$$

Remember, that q only counts the number of *different* instances that were addressed by a Send-query.

Putting everything together, we have

$$\mathsf{Adv}(\mathcal{A}) \leq \frac{q}{|\mathcal{D}|} + \mathrm{negl}(\ell).$$

□

6 Conclusion

We bring together the design ideas depicted in [5] and [7], to come up with an implementation of a group PAKE building on the two-party protocol of Katz et al. [9]. Following [10], this scheme can be implemented from Cramer-Shoup CCA-encryption, resulting in a very efficient (pairing free) design. Our construction can be proven secure in the standard model, and provides quite strong guarantees; in particular, we evidence that adversaries can only perform one online password test per instance.

Acknowledgments

We are indebted to Michel Abdalla for numerous valuable comments and discussions. Moreover, we would like to thank an anonymous referee of [7] for insightful and valuable comments.

Funding

M.I. González Vasco is partially supported by research projects MTM2013-41426-R, and MTM2016-77213-R, both funded by the Spanish MINECO.

References

[1] E. Bresson, O. Chevassut, D. Pointcheval, Group Diffie-Hellman Key Exchange Secure Against Dictionary Attacks, in: Advances in Cryptology – Proceedings of ASIACRYPT '02, Vol. 2501 of Lecture Notes in Computer Science, Springer, 2002, pp. 497–514.

[2] M. Abdalla, E. Bresson, O. Chevassut, D. Pointcheval, Password-based Group Key Exchange in a Constant Number of Rounds, in: M. Yung, Y. Dodis, A. Kiayias, T. Malkin (Eds.), Public Key Cryptography – PKC 2006, Vol. 3958 of Lecture Notes in Computer Science, Springer, 2006, pp. 427–442.

[3] R. Dutta, R. Barua, Password-Based Encrypted Group Key Agreement, International Journal of Network Security 3 (1) (2006) 23–34.

[4] M. C. Gorantla, C. Boyd, J. M. González Nieto, M. Manulis, Generic One Round Group Key Exchange in the Standard Model, in: Information, Security and Cryptology – ICISC 2009: 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers, Vol. 5984 of Lecture Notes in Computer Science, Springer, 2010, pp. 1–15.

[5] M. Abdalla, J.-M. Bohli, M. I. González Vasco, R. Steinwandt, (Password) Authenticated Key Establishment: From 2-Party to Group, in: S. P. Vadhan (Ed.), Theory of Cryptography Conference – TCC 2007, Vol. 4392 of Lecture Notes in Computer Science, Springer, 2007, pp. 499–514.

[6] M. Abdalla, D. Pointcheval, A Scalable Password-based Group Key Exchange Protocol in the Standard Model, in: X. Lai, K. Chen (Eds.), Proceedings of ASIACRYPT 2006, Vol. 4284 of Lecture Notes in Computer Science, Springer, 2006, pp. 332–347.

[7] J.-M. Bohli, M. I. G. Vasco, R. Steinwandt, Password-authenticated constant-round group key establishment with a common reference string, Cryptology ePrint Archive, Report 2006/214, http://eprint.iacr.org/2006/214, last revised 27 Apr 2009 (2006).

[8] R. Gennaro, Y. Lindell, A Framework for Password-Based Authenticated Key Exchange (Extended Abstract), in: E. Biham (Ed.), Advances in Cryptology – EUROCRYPT 2003, Vol. 2656 of Lecture Notes in Computer Science, Springer, 2003, pp. 524–543.

[9] J. Katz, V. Vaikuntanathan, Round-optimal password-based authenticated key exchange, J. Cryptology 26 (4) (2013) 714–743.

[10] F. Ben Hamouda, O. Blazy, C. Chevalier, D. Pointcheval, D. Vergnaud, New smooth projective hash functions and one-round authenticated key exchange, IACR Cryptology ePrint Archive 2013 (2013) 34.
URL http://eprint.iacr.org/2013/034

[11] V. Boyko, P. MacKenzie, S. Patel, Provably secure password-authenticated key exchange using diffie-hellman, in: B. Preneel (Ed.), Advances in Cryptology — EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings, Vol. 1807 of Lecture Notes in Computer Science, Springer, 2000, pp. 156–171.

[12] J. Katz, R. Ostrovsky, M. Yung, Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords, in: B. Pfitzmann (Ed.), Advances in Cryptology – EUROCRYPT 2001, Vol. 2045 of Lecture Notes in Computer Science, Springer, 2001, pp. 475–494.

[13] M. Abdalla, D. Pointcheval, Simple password-based encrypted key exchange protocols, in: A. Menezes (Ed.), Topics in Cryptology – CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005. Proceedings, Vol. 3376 of Lecture Notes in Computer Science, Springer, 2005, pp. 191–208.

[14] M. Abdalla, F. Benhamouda, P. MacKenzie, Security of the J-PAKE password-authenticated key exchange protocol, in: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015, IEEE Computer Society, 2015, pp. 571–587. doi:10.1109/SP.2015.41.
URL https://doi.org/10.1109/SP.2015.41

[15] S. Blake-Wilson, A. Menezes, Authenticated Diffie-Hellman Key Agreement Protocols, in: Proceedings of the Selected Areas in Cryptography, SAC '98, Springer-Verlag, London, UK, UK, 1999, pp. 339–361.
URL http://dl.acm.org/citation.cfm?id=646554.694440

[16] M. Bellare, R. Canetti, H. Krawczyk, A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract), in: J. S. Vitter (Ed.), Proceedings

of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, ACM, 1998, pp. 419–428. doi:10.1145/276698.276854.
URL http://doi.acm.org/10.1145/276698.276854

[17] J. Katz, M. Yung, Scalable protocols for authenticated group key exchange, J. Cryptology 20 (1) (2007) 85–113.

[18] M. Burmester, Y. Desmedt, A Secure and Efficient Conference Key Distribution System, in: A. D. Santis (Ed.), Advances in Cryptology – EUROCRYPT'94, Vol. 950 of Lecture Notes in Computer Science, Springer, 1995, pp. 275–286.

[19] A. Mayer, M. Yung, Secure Protocol Transformation via "Expansion": From Two-party to Groups, in: Proceedings of the 6th ACM conference on Computer and Communications Security CCS '99, ACM Press, 1999, pp. 83–92.

[20] J. Y. Hwang, S.-M. Lee, D. H. Lee, Scalable key exchange transformation: from two-party to group, Electronic Letters 40 (12) (2004) 728–729.

[21] M. Abdalla, P.-A. Fouque, D. Pointcheval, Password-Based Authenticated Key Exchange in the Three-Party Setting, in: S. Vaudenay (Ed.), Public Key Cryptography – PKC 2005, Vol. 3386 of Lecture Notes in Computer Science, Springer, 2005, pp. 65–84.

[22] M. Abdalla, P.-A. Fouque, D. Pointcheval, Password-Based Authenticated Key Exchange in the Three-Party Setting, IEE Proceedings – Information Security 153 (1) (2006) 27–39.

[23] R. Gennaro, Y. Lindell, A Framework for Password-Based Authenticated Key Exchange, Cryptology ePrint Archive: Report 2003/032, available at http://eprint.iacr.org/2003/032 (2003).

[24] R. Cramer, V. Shoup, Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption, in: L. Knudsen (Ed.), Advances in Cryptology – EUROCRYPT 2002, Vol. 2332 of Lecture Notes in Computer Science, Springer, 2002, pp. 45–64.

[25] J. Katz, R. Ostrovsky, M. Yung, Efficient and Secure Authenticated Key Exchange Using Weak Passwords, at the time of writing available online at http://www.cs.umd.edu/~jkatz/papers/password.pdf (2006).

[26] M. Bellare, P. Rogaway, Entitiy Authentication and Key Distribution, in: D. R. Stinson (Ed.), Advances in Cryptology – CRYPTO '93, Vol. 773 of Lecture Notes in Computer Science, Springer, 1994, pp. 232–249.

[27] M. Bellare, D. Pointcheval, P. Rogaway, Authenticated Key Exchange Secure Against Dictionary Attacks, in: B. Preneel (Ed.), Advances in Cryptology – EUROCRYPT 2000, Vol. 1807 of Lecture Notes in Computer Science, Springer, 2000, pp. 139–155.

[28] J.-M. Bohli, M. I. González Vasco, R. Steinwandt, Secure Group Key Establishment Revisited, International Journal of Information Security 6 (4) (2007) 243–254.

[29] M. I. González Vasco, C. Martínez, R. Steinwandt, J. L. Villar, A new Cramer-Shoup like methodology for group based provably secure schemes, in: J. Kilian (Ed.), Proceedings of the 2nd Theory of Cryptography Conference TCC 2005, Vol. 3378 of Lecture Notes in Computer Science, Springer, 2005, pp. 495–509.

[30] Y. T. Kalai, Smooth Projective Hashing and Two-Message Oblivious Transfer, in: R. Cramer (Ed.), Advances in Cryptology – EUROCRYPT 2005, Vol. 3494 of Lecture Notes in Computer Science, Springer, 2005, pp. 78–95.

[31] K. Kurosawa, Y. Desmedt, A New Paradigm of Hybrid Encryption Scheme, in: M. Franklin (Ed.), Advances in Cryptology – CRYPTO 2004, Vol. 3152 of Lecture Notes in Computer Science, Springer, 2004, pp. 426–442.

[32] O. Blazy, C. Chevalier, Generic Construction of UC-Secure Oblivious Transfer, in: T. Malkin, V. Kolesnikov, A. B. Lewko, M. Polychronakis (Eds.), Applied Cryptography and Network Security: 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers, Vol. 9092 of Lecture Notes in Computer Science, Springer, 2015, pp. 65–86.

[33] V. Shoup, An emerging standard for public-key encryption, ISO ISO 18033-2, International Organization for Standardization, Geneva, Switzerland, http://www.shoup.net/iso/std6.pdf (2006).

[34] R. Gennaro, Y. Lindell, A framework for password-based authenticated key exchange[1], ACM Trans. Inf. Syst. Secur. 9 (2) (2006) 181–234. doi:10.1145/1151414.1151418.
URL `http://doi.acm.org/10.1145/1151414.1151418`

[35] L. Carter, M. N. Wegman, Universal classes of hash functions (extended abstract), in: J. E. Hopcroft, E. P. Friedman, M. A. Harrison (Eds.), Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA, ACM, 1977, pp. 106–112.

[36] J. Katz, J. S. Shin, Modeling Insider Attacks on Group Key-Exchange Protocols, Cryptology ePrint Archive: Report 2005/163, available at `http://eprint.iacr.org/2005/163` (2005).