# Password-Authenticated Constant-Round Group Key Establishment with a Common Reference String

Jens-Matthias Bohli[1⋆], María Isabel González Vasco[2], and Rainer Steinwandt[3]

[1] NEC Laboratories Europe,
Kurfürsten-Anlage 36, 69115 Heidelberg, Germany;
bohli@nw.neclab.eu
[2] Departamento de Matemática Aplicada, Universidad Rey Juan Carlos,
c/ Tulipán s/n, 28933 Madrid, Spain;
mariaisabel.vasco@urjc.es
[3] Department of Mathematical Sciences, Florida Atlantic University,
777 Glades Road, Boca Raton, FL 33431, USA;
rsteinwa@fau.edu

**Abstract.** A provably secure password-authenticated protocol for group key establishment in the common reference string (CRS) model is presented. Our construction assumes the participating users to share a common password and combines projective hashing as introduced by Cramer and Shoup with a construction of Burmester and Desmedt. Our protocol is constant-round. Namely, it is a three-round protocol that can be seen as generalization of a two-party proposal of Gennaro and Lindell.

**Keywords:** group key establishment, password-based authentication

## 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. 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 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 [10, 3, 5]. However, to the best of our knowledge, all suggested

⋆ Work partly done while the author was at Universität Karlsruhe (TH).

constructions base on the random oracle or the ideal cipher model. On the other hand, using work of Katz et al. [18] as starting point, for the two-party case, Gennaro and Lindell [14, 15] recently proposed a protocol in the Common Reference String (CRS) model. Their main technical tool are smooth projective hash functions, which were introduced by Cramer and Shoup in [13]. The protocol we propose below can be taken for a generalization of Gennaro and Lindell's construction to a group setting, although there are some relevant differences. For instance, we do not rely on a one-time signature scheme, and we use Cramer and Shoup's original definition of projective hash families. Generally speaking, we describe a password-based constant-round group key establishment protocol that neither uses the random oracle nor the ideal cipher model. We would like to point out that in independent work, Abdalla et al. [4] follow a different approach aiming at the same goal.

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 [18, 19], building in turn on [7, 6]. In the subsequent section we recall the basic components of the security framework, addressing specifics of password-based authentication. Thereafter, in Section 3 we recall the needed tools concerning strongly universal projective hash functions 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 [14], 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.[4] 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:

- $\mathsf{used}_i^{s_i}$ 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;[5]
- $\mathsf{acc}_i^{s_i}$ 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 [6].

*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.

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

[5] 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.

$\mathsf{Reveal}(U_i, s_i)$ 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.

### 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 [8]. 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 $\mathsf{Send}$ 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 $\mathsf{Test}$-query may be executed.

*Freshness.* A $\mathsf{Test}$-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 $\ell$, defined as

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

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

Now, to capture key secrecy we follow an approach of [14]. 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 [3] in the sense that we do not tolerate a constant number $> 1$ of online guesses per protocol instance:

**Definition 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* $\mathsf{Send}$*-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 Strongly Universal$_2$ Hashing and Non-Malleable Commitments

The design of our protocol mainly builds on two basic tools: projective hashing and non-malleable commitments. Our usage of these tools is to a large extent inherited from [14]. In this section we review the main definitions and results necessary for the sequel. This revision is deployed at a somewhat intuitive level, and we refer to [13, 14] for formal definitions and proofs.

### 3.1 Strongly Universal$_2$ Projective Hashing

Kurosawa and Desmedt introduced in [21] the notion of strongly universal$_2$ projective hash families, building on the previous work of Cramer and Shoup on different flavors of projective hashing [13]. Projective hash families are usually understood as related to hard subset membership problems and in this fashion serve as basis for several provably secure cryptographic constructions [13, 14, 16, 17, 21].

**Definition 2.** *A subset membership problem $\mathcal{I}$ is a specification of a collection of probability distributions $\{I_\ell\}_{\ell \in \mathbb{N}}$, where for each $\ell$, $I_\ell$ is a probability distribution over* instance descriptions*. An instance description $\Lambda$ specifies:*

1. *Two finite, non-empty sets $X_\ell, L_\ell \subseteq \{0,1\}^{\mathrm{poly}(\ell)}$ with $L_\ell \subseteq X_\ell$.*

2. *Two probability distributions* $D(L_\ell)$ *and* $D(X_\ell \setminus L_\ell)$ *over* $L_\ell$ *and* $X_\ell \setminus L_\ell$ *respectively.*
3. *A set* $W_\ell \subseteq \{0,1\}^{\mathrm{poly}(\ell)}$*, together with an NP-relation* $R_\ell \subseteq X_\ell \times W_\ell$ *such that* $x \in L_\ell$ *if and only if there exists* $w \in W_\ell$ *such that* $(x, w) \in R_\ell$*.*

The above definition is taken from [14], and deviates slightly from that of [13]. Again following [14], we will only be interested in subset membership problems that are efficiently samplable, that is, for which probabilistic polynomial-time algorithms for the following tasks are available:

1. Upon input $1^\ell$, sample an instance $\Lambda$ from $I_\ell$,
2. Upon input $1^\ell$ and an instance $\Lambda$, sample $x \in L_\ell$ according to $D(L_\ell)$, together with a witness $w \in W_\ell$ for $x$.
3. Upon input $1^\ell$ and an instance $\Lambda$, sample a value $x \in X_\ell \setminus L_\ell$ according to $D(X_\ell \setminus L_\ell)$.

Our definition of a *hard subset membership problem* is identical to the one in [14] and basically says that within $X_\ell$ distinguishing random elements inside and outside $L_\ell$ is hard:

**Definition 3.** *Let* $\mathcal{I}$ *be a subset membership problem as above. Then we say that* $\mathcal{I}$ *is a* hard subset membership problem*, provided that the ensembles* $\{(\Lambda_\ell, x_\ell)\}_{\ell\in\mathbb{N}}$ *and* $\{(\Lambda_\ell, \hat{x}_\ell)\}_{\ell\in\mathbb{N}}$ *are computationally indistinguishable for* $\Lambda_\ell$*,* $x_\ell$ *and* $\hat{x}_\ell$ *sampled according to* $I_\ell, D(L_\ell)$ *and* $D(X_\ell \setminus L_\ell)$ *respectively.*

Subsequently, we make use of subset membership problems, where the set $X_\ell$ comes along with a certain type of partition:

**Definition 4.** *Let* $\mathcal{I}$ *be a subset membership problem as above and suppose that* $X_\ell = C_\ell \times \mathcal{D}_\ell$*. Further, for each* $pw \in \mathcal{D}_\ell$ *denote by* $X_\ell(pw)$ *(resp.* $L_\ell(pw)$*) the set of pairs* $(c, pw) \in X_\ell$*, (resp.* $(c, pw) \in L_\ell$*). The distributions induced by* $D(L_\ell)$ *and* $D(X_\ell \setminus L_\ell)$ *in* $X_\ell(pw)$ *and* $L_\ell(pw)$ *are denoted by* $D(L_\ell(pw))$ *and* $D(X_\ell(pw) \setminus L_\ell(pw))$*.*

*We say that* $\mathcal{I}$ *is a* hard partitioned subset membership problem*, provided that for every* $pw \in \mathcal{D}_\ell$*, the ensembles* $\{(\Lambda_\ell, x_\ell)\}_{\ell\in\mathbb{N}}$ *and* $\{(\Lambda_\ell, \hat{x}_\ell)\}_{\ell\in\mathbb{N}}$ *are computationally indistinguishable for* $\Lambda_\ell$*,* $x_\ell$ *and* $\hat{x}_\ell$ *being sampled according to* $I_\ell$*,* $D(L_\ell(pw))$ *and* $D(X_\ell(pw) \setminus L_\ell(pw))$ *respectively.*

This definition of hard partitioned subset membership problems is taken from [14] and captures the situation where each set $X_\ell$ can actually be partitioned into disjoint sets of hard problems. As Gennaro and Lindell do in [14], we stress here that the projective hash functions considered in the sequel will not take this partitioning into account. Moreover, in accordance with [13] (and differing from [14]) we use a definition of projective hash families where the projection function $\alpha$ has only one argument:

**Definition 5.** *Let* $X$*,* $\Pi$ *be finite non-empty sets and* $K$ *some finite index set. Consider a family* $H = \{H_k : X \longrightarrow \Pi\}_{k\in K}$ *of mappings from* $X$ *into* $\Pi$*, 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).*

*Then, given a subset $L \subseteq X$, we refer to the tuple $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$, as* projective hash family *(PHF) for $(X, L)$ if for all $k \in K$, $x \in L$ the value $H_k(x)$ is determined by $\alpha(k)$.*

We are mainly interested in a special type of projective hash families, which in [21] are called *strongly universal$_2$*:

**Definition 6.** *Let $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ be a PHF. Then we refer to $\mathbf{H}$ as* strongly universal$_2$ *if for $k \in K$ chosen uniformly at random, for any $x, x^* \in X \setminus L$, $x \neq x^*$ the random variables*

*- $\xi_k := H_k(x)$, conditioned to $\alpha(k)$*
*- $\eta_k$, the variable $\xi_k$ conditioned to both $\alpha(k)$ and $H_k(x^*)$*

*are statistically close to the uniform distribution over $\Pi$.*

In the sequel, we will consider only projective hash families which are *efficient* in the sense of [14], i. e., there are efficient algorithms available for sampling uniformly at random elements from $K$, computing $\alpha$ and evaluating $H_k$ at a given $x \in X$ provided that

- either $k$ is given as an input, or
- $x \in L$ and $(x, w)$, $\alpha(k)$ are given as input, where $w$ is a witness for $x$.

It is worth noting here that, in combination with a hard subset membership problem, the strongly universal$_2$ property guarantees that, for $x \neq x^* \in X$ and $\alpha(k), H_k(x^*)$ the value $H_k(x)$ is indistinguishable from random unless a corresponding witness is known.[6]

### 3.2 Strongly Universal$_2$ Hashing from Non-Malleable Commitments

Another essential component of Gennaro and Lindell's construction and of our proposal are 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 $\nu$, he must not be able to output a commitment to a value $\beta$ related to $\nu$ in a known way (that is, it must achieve *non-malleability for multiple commitments*).

[6] Smooth projective hashing would not suffice to guarantee independence with arbitrary different inputs $x \neq x^* \in X$.

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).

We briefly recall Gennaro and Lindell's proposal for constructing smooth projective hash families, given a suitable commitment scheme as above: Let $\mathcal{C}$ be a commitment scheme fulfilling the conditions above (thus, we are in the common reference string model). Let $\mathcal{D}$ a fixed message (password) space. We denote by $C_\rho(pw; r)$ a commitment to $pw \in \mathcal{D}$ using randomness $r$ and common reference string $\rho$. By $C_\rho$, let us denote the set of all strings that may be output by $\mathcal{C}$ when the common reference string is $\rho$. For an efficiently recognizable superset $C'_\rho \supseteq C_\rho$, define $X_\rho := C'_\rho \times \mathcal{D}$ and let

$$L_\rho := \{(c, pw) \in C_\rho \times \mathcal{D} \,|\, \exists\, r : c = C_\rho(pw; r)\} \subseteq X_\rho.$$

We consider a subset membership problem defined as follows. For each $\ell \in \mathbb{N}$ a common reference string $\rho$ (of polynomial size in $\ell$) is selected. Further, for each $pw \in \mathcal{D}$ define $D(X_\rho(pw) \setminus L_\rho(pw))$ respectively $D(L_\rho(pw))$ as the distribution induced by choosing random $r$ and computing $(C_\rho(0^{|pw|}; r), pw)$ respectively $(C_\rho(pw; r), pw))$. As it is argued in [14], it is easy to see that the hiding property of the commitment scheme yields the following

**Proposition 1.** *Let $\mathcal{C}$ be a non-interactive and non-malleable perfectly binding commitment scheme. Consider the above subset membership problem $\mathcal{I}$, where for each $\rho$ the set $X_\rho$ is partitioned by the sets $\{C'_\rho \times \{pw\}\}_{pw \in \mathcal{D}}$. Then, $\mathcal{I}$ is a hard partitioned subset membership problem.*

Now, assume we have a strongly universal$_2$ projective hash family defined with respect to $(X_\rho, L_\rho)$ as follows: Let $K$ be the key space, and for every $k \in K$ define

$$H_k : C'_\rho \times \mathcal{D} \longrightarrow G,$$

where $G$ is a finite abelian group of superpolynomial size. For the security proof of our protocol we need an analog of [14, Lemma 3.1]. Namely, we need that given a projection $\alpha(k) \in S$ and two valid commitments $c_1$ and $c_2$ on *the same* password $pw$, the values $H_k(c_1, pw)$ and $H_k(c_2, pw)$ are computationally indistinguishable from random (independent) values, provided appropriate witnesses are not known. Note that if the commitments are invalid (and hence $(c_1, pw)$ and $(c_2, pw)$ are outside $L$), this follows trivially from the definition of strongly universal$_2$. For valid commitments, this is a consequence of the hard subset membership problem.

**Lemma 1.** *Let $\mathcal{I}$ be the hard partitioned subset membership problem described above. To each instance $\Lambda = (X, D(X \setminus L), L, D(L), W, R)$, associate the above strongly universal$_2$ projective hash family $\mathbf{H} = (H, K, X, L, G, S, \alpha)$ for $(X, L)$. Let $M$ be a ppt oracle machine, and define the following experiments:*

Exp-Hash$(M)$: *An instance* $\Lambda = (X, D(X \setminus L), L, D(L), W, R)$ *is selected from* $I_\ell$. *Then* $M$ *is given access to three oracles* $\Omega_L$, Project *and* Hash *:*

$\Omega_L$: *When queried with a value* $pw \in \mathcal{D}$, *it outputs* $C_\rho(pw, r)$ *with the pair* $(C_\rho(pw, r), pw)$ *being selected according to* $D(L(pw))$ *from* $L(pw)$.

Project: *Chooses a key* $k \in K$ *uniformly at random and returns* $\alpha(k)$.

Hash: *When queried with input* $(pw, c, \alpha(k))$, *it first checks if* $c$ *was output by* $\Omega_L$ *on input* $pw$, *and* $\alpha(k)$ *has been output by* Project. *If at least one of this is the case,* Hash *outputs* $H_k(c, pw)$. *Otherwise* Hash *outputs nothing.*

*The output of the experiment is the output of* $M$.

Exp-Unif$(M)$: *Exactly as above, except that the* Hash *oracle is substituted by an oracle* Unif *which first checks whether the input* $c$ *was output by* $\Omega_L$ *on input* $pw$, *and that* $\alpha(k)$ *has been output by* Project. *If both are true,* Unif *outputs a uniformly at random chosen* $g \in G$; *moreover, for two different calls with the same input* $\alpha(k)$, *the corresponding two random group elements will be selected independently. If only one of* $c$ *and* $\alpha(k)$ *was output by an oracle,* Unif *outputs* $H_k(c, pw)$, *otherwise,* Unif *outputs nothing.*

*Then, the above experiments are computationally indistinguishable, that is, for any ppt oracle machine* $M$, *for any value* $v$ *it may output,*

$$|\Pr[\textsf{Exp-Unif}(M) = v] - \Pr[\textsf{Exp-Hash}(M) = v]|$$

*is negligible in the security parameter* $\ell$.

*Proof.* This proof is a straightforward variation of the proof of [14, Lemma 3.1]. As they do, we define the experiments $\textsf{Exp-Unif}_{X\setminus L}$ and $\textsf{Exp-Hash}_{X\setminus L}$ by replacing the oracle $\Omega_L$ by an oracle $\Omega_{X\setminus L}$ defined in the obvious way. Now, as we are dealing with a hard partitioned subset membership problem, both

$$|\Pr[\textsf{Exp-Hash}_{X\setminus L}(M) = v] - \Pr[\textsf{Exp-Hash}(M) = v]|$$

and

$$|\Pr[\textsf{Exp-Unif}_{X\setminus L}(M) = v] - \Pr[\textsf{Exp-Unif}(M) = v]|$$

are negligible. Furthermore,

$$|\Pr[\textsf{Exp-Hash}_{X\setminus L}(M) = v] - \Pr[\textsf{Exp-Unif}_{X\setminus L}(M) = v]|$$

is also negligible by the definition of strongly universal$_2$, and putting it all together we have

$$\begin{aligned}
&|\Pr[\textsf{Exp-Unif}(M) = v] - \Pr[\textsf{Exp-Hash}(M) = v]| \\
&\leq |\Pr[\textsf{Exp-Hash}(M) = v] - \Pr[\textsf{Exp-Hash}_{X\setminus L}(M) = v]| \\
&+|\Pr[\textsf{Exp-Hash}_{X\setminus L}(M) = v] - \Pr[\textsf{Exp-Unif}_{X\setminus L}(M) = v]| \\
&+|\Pr[\textsf{Exp-Unif}_{X\setminus L}(M) = v] - \Pr[\ \textsf{Exp-Unif}(M) = v]|,
\end{aligned}$$

from which the desired result follows. □

## 4 A Group Key Establishment Protocol

The protocol we propose builds on a non-interactive non-malleable commitment scheme $\mathcal{C}$ and a strongly universal$_2$ projective hash family $H = \{H_k\}_{k \in K}$ 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 $G$. Furthermore, we 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 $G$ onto a superpolynomial sized set $T$ of cardinality $|T| \leq \sqrt{|G|}$. Similarly as Bresson et al. [9], 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 $\mathrm{UH}'$ as input, has no more than a negligible probability to predict $\mathrm{UH}'(g)$ for an (unknown) uniformly at random chosen $g \in G$.

*Example 1.* Let $G := \mathbb{Z}/p\mathbb{Z}$ be the additive group of integers modulo an $\ell$-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 [12] 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., [12, 22]). Moreover, as the case $a = 0$ is excluded in the affine maps considered, for a uniformly at random chosen $g \in G$, also $a \cdot g + b$ is uniformly at random distributed in $G$, 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 $\mathcal{UH}$. 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 [20]. 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 [20] for more details). As in [20] 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 $pw$.

Our protocol is symmetric in the sense that all users perform the same steps. Figure 1 shows the three rounds of our protocol. For the sake of readability, we do not explicitly refer to instances $s_i$ of users.

### 4.1 Design Rationale

The basic design of the protocol follows the Burmester-Desmedt [11] construction where the Diffie-Hellman key exchanges are replaced by a simultaneous version of Gennaro-Lindell's [14] key exchange. A difference is the construction of the master key as

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

**Round 1:**

**Broadcast** Each $U_i$ chooses uniformly at random a value $k_i \in K$ and random nonces $r_i$. Then, $U_i$ constructs $c_i := C_\rho(pw;r_i)$, $S_i := \alpha(k_i)$, and broadcasts $M_i^1 := (U_i, S_i, c_i)$.

**Check** Each $U_i$ waits until messages $M_j^1$ for all $U_j$ arrived, and checks if the values $c_{i+1}$ and $c_{i-1}$ are contained in $C'_\rho$. If $c_{i+1} \notin C'_\rho$ or $c_{i-1} \notin C'_\rho$, then set $\mathsf{acc}_i := \mathsf{false}$ and terminate the protocol execution.

**Round 2:**

**Computation** Each $U_i$ computes

$$Z_{i,i+1} := H_{k_i}(pw, c_{i+1}) \cdot H_{k_{i+1}}(pw, c_i),$$

$$Z_{i,i-1} := H_{k_i}(pw, c_{i-1}) \cdot H_{k_{i-1}}(pw, c_i).$$

Each $U_i$ sets $X_{i,0} := Z_{i,i+1} \cdot Z_{i,i-1}^{-1}$ and chooses a random $X_{i,1} \in G$. Furthermore, $U_i$ chooses random values $r'_{i,0}$, $r'_{i,1}$ to compute commitments $C_\rho(U_i, X_{i,0};r'_{i,0})$ and $C_\rho(U_i, X_{i,1};r'_{i,1})$ and chooses at random $\mathrm{UH}'_i \in \mathcal{UH}'$ to compute 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, and checks if $\mathrm{UH}'_{i+1}(Z_{i,i+1}) = \mathsf{test}_{i+1}$. If the check succeeds, set $(X_i, r'_i) := (X_{i,0}, r_{i,0})$ otherwise $(X_i, r'_i) := (X_{i,1}, r_{i,1})$.

**Round 3:**

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

**Check** Each $U_i$ checks that $X_1 \cdots X_n = 1$ and 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 the protocol execution.

**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 $\mathsf{acc}_i := \mathsf{true}$.

**Fig. 1.** A password-authenticated 3-round protocol for group key establishment.

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 [8]. 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 $\mathsf{mk}$ (and thus, identical $\mathsf{sid}_i$, $\mathsf{sk}_i$). Note that with the construction of $\mathsf{mk}$ proposed above, both $\mathsf{sid}_i$ and $\mathsf{sk}_i$ will be indistinguishable from random if a sole honest user is involved in the protocol run.

One might also wonder about the additional Round 2 where commitments to the quotients $X_i$ are broadcast. This is motivated by the following online attack on the protocol consisting only of Round 1 and Round 3, that allows to test two passwords using only one instance $\Pi_i^{s_i}$:

- The adversary $\mathcal{A}$ chooses two candidate passwords $pw_1$ and $pw_2$. Then $\mathcal{A}$ initializes a protocol run of $U_i$ via $\mathsf{Send}$ and tries to impersonate all users to $U_i$.
- In the name of the neighboring users $U_{i-1}$ and $U_{i+1}$, $\mathcal{A}$ sends the respective messages

$$M_{i-1}^1 = (U_{i-1}, \alpha(k_{i-1}), C_\rho(pw_1; r_{i-1}))$$
$$M_{i+1}^1 = (U_{i+1}, \alpha(k_{i+1}), C_\rho(pw_2; r_{i+1})),$$

  with honestly generated $k_{i-1}, k_{i+1}, r_{i-1}, r_{i+1}$. The other users' messages are not relevant as $U_i$ will ignore them.
- In the following round (we assume it will be Round 3) $\mathcal{A}$ can make up values $X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n$ such that $X_1 \cdots X_n = 1$ and send the messages $M_j^3 = (U_j, X_j)$ for $j = 1, \ldots, n$, $j \neq i$. (Note that the message part $r'_j$ is only part of the full protocol that includes Round 2.)
- The adversary will now compute two candidate views of $U_i$ for the values $Z_{\mu,\nu}$. Assuming $pw_1$ was correct and so knowing $U_i$'s value $Z_{i,i-1}$, $\mathcal{A}$ computes $Z_{i-1,i-2}^{pw_1}$, $Z_{i-2,i-3}^{pw_1}, \ldots, Z_{i,i+1}^{pw_1}$ as in the protocol. In a similar way $\mathcal{A}$ computes the values $Z_{i-2,i-1}^{pw_2}, \ldots, Z_{i,i+1}^{pw_2}$ starting from $Z_{i,i+1}$. Then $\mathcal{A}$ can compute candidate master keys

$$\mathsf{mk}_{pw_b} = (Z_{1,2}^{pw_b}, \ldots, Z_{n,1}^{pw_b}) \quad (b = 1, 2)$$

  and finally two candidate session keys.
- The adversary will now query $\mathsf{Reveal}(U_i, s_i)$ giving him the session key $\mathsf{sk}_i$ and compare it with his candidate keys.

Through this attack, the adversary could check two passwords per session and the protocol would not fulfill the tight bound of security in Definition 1.

*Remark 1.* In principle, this observation applies to the proposal of Abdalla et al. [3], too. However, in their model a constant number of password checks per faked message is allowed.

The $\mathsf{test}_i$-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 the projective hash function being strongly universal$_2$,

as this gives 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 1).

Finally, the random value $X_{i,1}$ is needed if the check of a $\mathsf{test}_i$-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.

### 4.2 Security Analysis

**Theorem 1.** *With the prerequisites as described above, the protocol in Figure 1 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 UH($\mathsf{mk}$) and therewith also derive the same session key.

*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 $\ell$. Furthermore, we will index the $\mathsf{Send}$ oracle, denoting by $\mathsf{Send}_0$ the $\mathsf{Send}$ query that initializes a protocol run and by $\mathsf{Send}_i$ a $\mathsf{Send}$ 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 $\mathsf{Send}_3$ 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 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 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 commitments were oracle-generated, i. e., honestly transmitted or replayed from other instances. Distinction between this game and Game 5 reduces to distinguishing between Exp-Hash and Exp-Unif from Lemma 1.
– Game 7 deals with the case in which the adversary may insert invalid commitments. The simulator, detecting an invalid commitment, will choose $X_{i,0}$ at random. This modification is by definition of strongly universal$_2$ not detectable by the adversary from the messages exchanged.
– Game 8 deals with the case of valid commitments generated by the adversary, in which case he wins. This corresponds to a correct guess for the password.
– Game 9 aborts in case any 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 ruled out all malicious Send-queries. First, in Game 10, commitments 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 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 $G$. 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 $\mathsf{mk}$ will be a randomly selected element from $G^n$.

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-Unif and Exp-Hash from Lemma 1. Indeed, for a fixed common reference string and password the adversary cannot distinguish between Exp-Unif and Exp-Hash, for $i = 1, \ldots, n$. That means, seeing commitments $c_{i-1}, c_{i+1}$ and the projection $\alpha(k_i)$, he cannot tell $H_{k_i}(c_{i-1}, pw)$ and $H_{k_i}(c_{i+1}, pw)$ 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 Execute oracle is again modified, so that a password $\widehat{pw}$ is chosen uniformly at random from $\mathcal{D}$. Further, define each $c_i$ accordingly as $c_i = C_\rho(\widehat{pw}; r_i)$ for randomly selected nonces $r_i$. Due to the hiding property of the commitment scheme, 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 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 Execute oracle returns only random values, independent of the password, and instances used by an Execute-query hold only random session keys. The following games will deal with the Send oracle.

We will in the following call a commitment that was generated by the simulator *oracle-generated* and in accordance a commitment that was generated by the adversary *adversary-generated*. This can be checked efficiently by keeping a list of all commitments the simulator generates. Furthermore, we call the commitment *valid* if it is indeed a commitment for the password $pw$ and *invalid* otherwise. This cannot be checked efficiently, but as the commitment scheme is perfectly binding it is information-theoretically computable. Note that also commitments that are replayed by the adversary are *oracle-generated*.

**Game 4**. In this experiment all commitments 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 the respective commitment $c_{i-1}$ is *oracle-generated*, the respective 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 $H_{k_{i-1}}(pw, c_i)$ may be known to the adversary. However, as $k_i$ was honestly generated, $c_{i-1}$ oracle-generated, and the PHF is strongly universal$_2$, the hash $H_{k_i}(pw, c_{i-1})$ 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 $H_{k_i}(pw, c_{i-1})$. 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 $\mathsf{test}_i$ 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 $\mathsf{test}_i$ nor can he produce a $\mathsf{test}_i$ that will be accepted by $U_{i-1}$.

- Even with knowledge of all $\mathsf{test}_j$, $j = 1 \ldots n$, the value $Z_{i,i-1}$ remains indistinguishable from an independently and uniformly at random chosen element in $G$, because the $\mathsf{test}_j$ carry only a negligible amount of information due to $|T| \leq \sqrt{|G|}$.
- The adversary cannot produce $\mathsf{test}'_i$ that would be accepted by $U_{i-1}$: As $\mathrm{UH}'_i$ is chosen independent at random it is independent from $Z_{i-1,i}$ and the adversary has no prior information on the $\mathsf{test}_i$-value expected by $U_{i-1}$, because $Z_{i,i-1}$ is indistinguishable from random for him. Suppose the adversary tries to insert both $\mathsf{test}'_i$ and $\mathrm{UH}''_i$. He will only succeed if he is able to find $h$ and $\mathrm{UH}''_i$ 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 the respective commitment $c_{i+1}$ is *oracle-generated*, $U_i$ will continue with $X_{i,1}$ in Round 3.

We show that $U_i$'s check of $\mathsf{test}_{i+1}$ 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 $\mathsf{test}_{i+1}$ 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 $\mathsf{test}_{i+1}$ 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 $(c_{i-1}, S_{i-1}, c_i, S_i) \rightarrow Z_{i,i-1}$, $(c_i, S_i, c_{i+1}, S_{i+1}) \rightarrow Z_{i,i+1}$. The simulator behaves as in Game 5, except for the answer following a $\mathsf{Send}_1$ 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 $c_{i-1}$ and $c_{i+1}$ were both *oracle-generated* (but all projections were unmodified). In this case, the simulator checks if $(c_{i-1}, S_{i-1}, c_i, S_i) \rightarrow Z_{i,i-1}$ or $(c_i, S_i, c_{i+1}, S_{i+1}) \rightarrow Z_{i,i+1}$ 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 $G$ and the assignment $(c_{i-1}, S_{i-1}, c_i, S_i) \rightarrow Z_{i,i-1}$ and $(c_i, S_i, c_{i+1}, S_{i+1}) \rightarrow Z_{i,i+1}$, 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-Hash and Exp-Unif. Thus, from Lemma 1 we can conclude that $\mathcal{A}$'s advantage between the two games differs at most negligibly.

The distinguisher $D$ is either facing Exp-Hash or Exp-Unif from Lemma 1. $D$ is constructed so that it will behave like the simulator from Game 5, except:

- commitments $c$ are not computed but obtained by the $\Omega_L(pw)$ oracle,
- if a Send-query of the adversary requires $D$ to compute values $Z_{i,i-1}$ respectively $Z_{i,i+1}$, $D$ will query Hash/Unif with the respective values $c_{i-1}$ and $c_{i+1}$ if both were *oracle-generated*.

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

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

For the following games, the simulator is given an Extract oracle, that checks if a given commitment is valid, i. e., a commitment to the password $pw$. This can be done because the password is information-theoretically contained in the commitment. On input a commitment $c$, the 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 Extract oracle.

**Game 7**. In this experiment, the simulator behaves as in Game 6, except that in Round 2's computation phase, following a $\mathsf{Send}_1$ query, the received commitments $c_{i-1}$ and $c_{i+1}$ are checked by the simulator w.r.t. the password using the Extract oracle. Then, those instances $\Pi_i^{s_i}$ that received an invalid commitment $c_{i-1}$ or $c_{i+1}$ 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 commitment $c$ to a wrong password $\widetilde{pw}$ (that is, $(c, \widetilde{pw}) \notin L_\rho$,) was sent, then by the strongly universal$_2$ property of the PHF, the distribution of $(c, \widetilde{pw}, \alpha(k), H_k(c, \widetilde{pw}))$ 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 $\mathsf{Send}_1$ query are unchanged where the commitments 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 $\mathsf{Send}_1$-query *valid* commitments $c_{i-1}$ and $c_{i+1}$ 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 $C_\rho(X_i;r'_i)$ 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 $C_\rho(X_i^*;r^*)$ 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 $C_\rho(X_i;r'_i)$ 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 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 $\mathsf{Send}$-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 commitments 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, the commitments were not used in any projective hash function anymore. All information about the password, which was available to the adversary, were commitments of the form $C_\rho(pw;r)$ for a random value $r$. Due to the hiding property of the commitment scheme, the adversary detects the use of $\widehat{pw}$ instead of $pw$ with negligible probability only. Now the adversary does not have any correct commitments to the password as input. But, due to the non-malleability of the commitment scheme, the adversary's probability to succeed in producing a new commitment on $pw$ 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 $\mathsf{Send}_3$-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 UH($\mathsf{mk}$) and therefore hold identical session identifiers. Thus, those instances are partnered and the freshness definition renders the $\mathsf{Reveal}$-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 commitment 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 $\mathsf{Send}$-query.

Putting everything together, we have

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

□

## 5 Conclusion and Further Remarks

We have proposed a password-authenticated three-round protocol for group key establishment that achieves key secrecy, implicit key authentication and key integrity in the common reference string model. Moreover, our definition of key secrecy imposes that adversaries are not able to test more than one password at each session. To date, we are not aware of any other protocol fulfilling the above requirements and neither requiring random oracles nor ideal ciphers. Our construction can be seen as a generalization of the two-party protocol of Gennaro and Lindell from [14], diverging from the approach taken there in that

- we do not require the use of a one-time signature scheme,
- we make use of the original definition of projective hash families, for which projections determine the complete action of the corresponding hash function on $L$. Gennaro and Lindell's usage of smooth projective hashing is less demanding in this sense, as they only consider projection mappings that applied to pairs $(k, x)$ only determine the value of $H_k(x)$.

– we construct session keys and session identifiers via collision-resistant pseudorandom functions, following the approach of Katz and Shin from [20]. Note that if an adversary has guessed the password, in order to preserve the integrity of the protocol we have to guarantee he will not be able to find two different master keys yielding the same session identifier but two different session keys.

In the same fashion as Gennaro and Lindell's construction, instantiations of our protocol can be constructed from any IND-CCA2 secure encryption scheme that admits an efficient construction of strongly universal$_2$ projective hashing. Deriving the required non-malleable commitments via such an encryption scheme would actually yield a hard subset membership problem related to the language of pairs $(c, m)$ where $c$ is a valid encryption of $m$ using part of the common reference string as public key.

Following an observation of Abdalla [1], in joint work we have explored to what extent ideas in the above protocol are useful in another setting. In [2] we show that an appropriate use of commitments enables the efficient derivation of a constant-round group key establishment from any authenticated two-party key establishment without having to rely on signatures.

## Acknowledgments

We are indebted to Michel Abdalla for numerous valuable comments and discussions that have improved the original draft significantly, as well as kindly providing us a preprint of [4]. Moreover, we would like to thank an anonymous referee for insightful and valuable comments.

## References

1. Michel Abdalla. Personal communication, June 2006.
2. Michel Abdalla, Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. (Password) Authenticated Key Establishment: From 2-Party to Group. In Salil P. Vadhan, editor, *Theory of Cryptography Conference – TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2007.
3. Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Password-based Group Key Exchange in a Constant Number of Rounds. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography – PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 427–442. Springer, 2006.
4. Michel Abdalla and David Pointcheval. A Scalable Password-based Group Key Exchange Protocol in the Standard Model. In Xuejia Lai and Kefei Chen, editors, *Proceedings of ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 2006.
5. Ratna Dutta amd Rana Barua. Password-Based Encrypted Group Key Agreement. *International Journal of Network Security*, 3(1):23–34, July 2006.

6. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, 2000.
7. Mihir Bellare and Phillip Rogaway. Entitiy Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1994.
8. Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. Secure Group Key Establishment Revisited. *International Journal of Information Security*, 6(4):243–254, 2007.
9. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2002.
10. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Group Diffie-Hellman Key Exchange Secure Against Dictionary Attacks. In *Advances in Cryptology – Proceedings of ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 497–514. Springer, 2002.
11. Mike Burmester and Yvo Desmedt. A Secure and Efficient Conference Key Distribution System. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 1995.
12. J. Lawrence Carter and Mark N. Wegman. Universal Classes of Hash Functions. In *STOC '77: Ninth Annual ACM Symposium on Theory of Computing*, pages 106–112. ACM, 1977.
13. Ronald Cramer and Victor Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In Lars Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.
14. Rosario Gennaro and Yehuda Lindell. A Framework for Password-Based Authenticated Key Exchange. Cryptology ePrint Archive: Report 2003/032, 2003. Available at `http://eprint.iacr.org/2003/032`.
15. Rosario Gennaro and Yehuda Lindell. A Framework for Password-Based Authenticated Key Exchange (Extended Abstract). In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, 2003.
16. María Isabel González Vasco, Consuelo Martínez, Rainer Steinwandt, and Jorge L. Villar. A new Cramer-Shoup like methodology for group based provably secure schemes. In Joe Kilian, editor, *Proceedings of the 2nd Theory of Cryptography Conference TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 495–509. Springer, 2005.
17. Yael Tauman Kalai. Smooth Projective Hashing and Two-Message Oblivious Transfer. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95. Springer, 2005.
18. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, 2001.

19. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient and Secure Authenticated Key Exchange Using Weak Passwords, 2006. At the time of writing available online at `http://www.cs.umd.edu/~jkatz/papers/password.pdf`.
20. Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *12th ACM conference on Computer and communications security*, pages 180–189. ACM, 2005.
21. Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2004.
22. Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, second edition, 2008. At the time of writing available online at `http://www.shoup.net/ntb/ntb-v2.pdf`.