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] Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe, 76128 Karlsruhe, Germany;
bohli@ira.uka.de

[2] Departamento de Matemática Aplicada, Universidad Rey Juan Carlos, c/ Tulipán, s/n, 28933 Madrid, Spain;
migonzalez@escet.urjc.es

[3] Center for Cryptology and Information Security, Dept. 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 smooth 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 [8, 2, 4, 20]. However, to the best of our knowledge, all suggested constructions base on the random oracle or the ideal cipher model. On the other hand, using work of Katz et al. [16] as starting point, for the two-party case, Gennaro and Lindell [12, 13] 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 [11]. 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. [3] 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 [16, 17], building in turn on [6, 5]. 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 smooth 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 [12], 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 $\Pi_i^{s_i}$ ($i \in \mathbb{N}$).

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

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 [5].

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 $\mathsf{sk}_i^{s_i} \neq$ 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.

[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.

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 [7]. 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 $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j} \neq$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 [12]. 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 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 Non-Malleable Commitments

The design of our protocol mainly builds on two basic tools: smooth projective hashing and non-malleable commitments. Our usage of these tools is to a large extent inherited from [12]. 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 [11, 12] for formal definitions and proofs.

3.1 Smooth Projective Hashing

Cramer and Shoup introduced the notion of *Smooth Projective Hashing* in [11]. Smooth 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 [11, 12, 14, 15, 19].

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 ℓ, I_ℓ is a probability distribution over* instance descriptions*. An instance description Λ 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_ℓ 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 [12], and deviates slightly from that of [11]. Again following [12], 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^ℓ, sample an instance Λ from I_ℓ,
2. Upon input 1^ℓ and an instance Λ, sample $x \in L_\ell$ according to $D(L_\ell)$, together with a witness $w \in W_\ell$ for x.

3. Upon input 1^ℓ and an instance Λ, 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 [12] and basically says that within X_ℓ distinguishing random elements inside and outside L_ℓ 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 Λ_ℓ, x_ℓ 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_ℓ 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 Λ_ℓ, x_ℓ and $\hat{x}_\ell$ being sampled according to I_ℓ, $D(L_\ell(pw))$ and $D(X_\ell(pw) \setminus L_\ell(pw))$ respectively.*

This definition of hard partitioned subset membership problems is taken from [12] and captures the situation where each set X_ℓ can actually be partitioned into disjoint sets of hard problems. As Gennaro and Lindell do in [12], we stress here that the smooth projecting hash functions considered in the sequel will not take this partitioning into account. Moreover, in accordance with [11] (and differing from [12]) we use a definition of projective hash families where the projection function α has only one argument:

Definition 5. *Let X, Π 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 Π, 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 [12] are called *smooth* (this notion differs from the notion of *smoothness* in [11]):

Definition 6. *Let $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ be a PHF. Then we refer to $\mathbf{H}$ as* smooth *if for each $x \in X \setminus L$ the probability distributions of $(x, s, H_k(x))$ and (x, s, π) are statistically close, where k and π are chosen uniformly at random in K and Π, respectively, and $s = \alpha(k)$.*

In the sequel, we will consider smooth projective hash families which are *efficient* in the sense of [12], i. e., there are efficient algorithms available for sampling uniformly at random elements from K, computing α 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.

3.2 Smooth Projective 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 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).

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 ρ. By C_ρ, let us denote the set of all strings that may be output by $\mathcal{C}$ when the common reference string is ρ. 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 ρ (of polynomial size in ℓ) 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 [12], 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 ρ the set X_ρ 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 smooth projective hash family defined with respect to (X_ρ, L_ρ) 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 super-polynomial size. For the security proof of our protocol we need an analog of [12, Lemma 3.1]. Namely, we need that given a projection $\alpha(k) \in S$ and two valid commitments c_1 and c_2 of *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.

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 smooth 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_ℓ. Then M is given access to two oracles Ω_L and* Hash*:*
  - *Ω_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 $L(pw)$.*
  - Hash*: When queried with an input (pw, c_1, c_2), it first checks that both c_1 and c_2 were output by Ω_L on input pw, and if so it chooses uniformly at random a key $k \in K$ and returns the triple $(\alpha(k), H_k(c_1, pw), H_k(c_2, pw))$. 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 was output by the Ω_L oracle, and if so chooses uniformly at random a key $k \in K$ and returns the triple $(\alpha(k), g_1, g_2)$, where g_1, g_2 are chosen independently and uniformly at random in G. Otherwise, it 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 ℓ.

Proof. This proof is a straightforward variation of the proof of [12, 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 Ω_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

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

are negligible. Furthermore,

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

is also negligible by the definition of smooth hashing, and putting it all together we have

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

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 smooth 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 map elements from G^n onto a superpolynomial-sized set $\{0,1\}^L$. The CRS selects one universal hash function UH from this 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 [18]. 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 no ppt adversary can find two differenct indices $\lambda \neq \lambda' \in \{0,1\}^L$ such that $F_\lambda(v_0) = F_{\lambda'}(v_0)$ (see [18] for more details). As in [18] we use another public value v_1 (which, like v_0 can be included in the CRS) for deriving the session key.

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 [9] construction where the Diffie-Hellman key exchanges are replaced by a simultaneous version of Gennaro-Lindell's [12] key exchange. A difference is the construction of the master key as

$$K = (Z_{1,2}, Z_{2,3}, \dots, Z_{n-1,n}, Z_{n,1}).$$

The original construction $K = \prod_{i=1,\dots,n} Z_{i,i+1}$ can be determined by two malicious users as pointed out in [7]. Thus, if an adversary guesses the password,

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_j are in C'_ρ.

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 $X_i := Z_{i,i+1} \cdot Z_{i,i-1}^{-1}$ and chooses a random r'_i to compute a commitment $c_\rho(X_i, r'_i)$.

Broadcast Each user U_i broadcasts $M_i^2 := (U_i, c_\rho(X_i, r'_i))$.

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 $c_\rho(X_j, r'_j)$.

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

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

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

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

he would be able to provoke pathological behaviors such that each protocol run ends up with exactly the same K (and thus, identical sid_i, sk_i). Note that with the construction of K proposed above, both sid_i and 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 broadcasted. 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 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

$$M^1_{i-1} = (U_{i-1}, \alpha(k_{i-1}), c_\rho(pw_1, r_{i-1}))$$
$$M^1_{i+1} = (U_{i+1}, \alpha(k_{i+1}), c_\rho(pw_2, r_{i+1})),$$

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 $M^3_j = (U_j, X_j)$ 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. Assuming pw_1 was correct and so knowing U_i's value $Z_{i,i-1}$, $\mathcal{A}$ computes $Z^{pw_1}_{i-1,i-2}$, $Z^{pw_1}_{i-2,i-3}, \ldots, Z^{pw_1}_{i,i+1}$ as in the protocol. In a similar way $\mathcal{A}$ computes the set $Z^{pw_2}_{i-2,i-1}, \ldots, Z^{pw_2}_{i,i+1}$ starting from $Z_{i,i+1}$. Then $\mathcal{A}$ can compute candidate master keys

$$K_{pw_b} = (Z^{pw_b}_{1,2}, \ldots, Z^{pw_b}_{n,1}) \quad (b = 1, 2)$$

and finally two candidate session keys.
– The adversary will now query $\mathsf{Reveal}(U_i, s_i)$ giving him the session key 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. [2], too. However, in their model a constant number of password checks per faked message is allowed.

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 $\mathrm{UH}(K)$ 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 ℓ. Furthermore, for clarity, we will index the Send oracle, by the round in which the transmitted message was sent, beginning with Send_0 for the initialization. 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.

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 $i = 1, \dots, n$ 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 $\prod_{i=1,\dots,n} X_i = 1$ and the master key K 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 $i = 1, \dots, n$. 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 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

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

Game 2. At this, the Execute oracle is again modified, so that a password $\hat{pw}$ is chosen uniformly at random from $\mathcal{D}$. Further, define each c_i accordingly as $c_i = c_\rho(\hat{pw}, r_i)$ for randomly selected nonces r_i. Due to the hiding property of the commitment scheme, we again have

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

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

The master key $K = (Z_{1,2}, \dots, Z_{n,1})$ has, once the X_i are public, sufficient entropy so that the output of the pseudorandom function $F_{UH(K)}$ is distinguishable from a random $\mathsf{sk}_i^{s_i}$ with negligible probability only.

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

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* else. This cannot be checked efficiently, however, as the commitment scheme is perfectly binding it is information-theoretically computable.

Game 4. In this experiment, the simulator behaves as in Game 3, except that in Round 2's computation phase, following a Send_1 query, all received commitments are checked by the simulator w.r.t. the password. 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 $Z_{i,i-1}$ (respectively $Z_{i,i+1}$).

Note that the adversary has negligible probability of distinguishing between Game 4 and Game 3. If in Round 1 an invalid commitment c to a wrong password $\widetilde{pw}$ (that is, $(c, \widetilde{pw}) \notin L_{\rho},$) was sent to any instance, then by the definition of smooth projective hashing the distribution $(c, \widetilde{pw}, \alpha(k), H_k(c, \widetilde{pw}))$ is statistically close to $(c, \widetilde{pw}, \alpha(k), g)$ for a random group element $g \in G$. Thus, the corresponding $Z_{i,j}$ will look like a random group element for the adversary, who thus has only a negligible chance to detect the difference:

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

Game 5. In this experiment, again for the instances $\Pi_i^{s_i}$ that were modified in a Send_1 query in the previous game the simulator chooses *both* $Z_{i,i-1}$ and $Z_{i,i+1}$ at random. Moreover, $\Pi_i^{s_i}$ will not accept in Round 3.

There exists only a difference to Game 4 when exactly one commitment was *invalid*. As in this situation one value of $Z_{i,i-1}$ and $Z_{i,i+1}$ was already chosen at random, so was already the quotient X_i. Thus, in this game the output distribution of the messages M_i^2 and M_i^3 does not differ from Game 4 and the adversary has no chance to detect it from the messages. The second change is the abort in Round 3. Before M_i^3 will be sent, only $c_{\rho}(X_i, r'_i)$ is known about X_i as this is a random value of instance $\Pi_i^{s_i}$. To pass the check in Round 3, $\Pi_i^{s_i}$ expects commitments $c_{\rho}(X_j, r'_j)$ for $j = 1, \ldots, n$ such that $X_1 \cdots X_n = 1$. As X_i is a random value of which only $c_{\rho}(X_i, r'_i)$ is known and the commitment scheme is non-malleable, the adversary's probability to pass the test and detect the difference is only negligible.

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

By now, only such executions of Round 2 following a Send_1 query are unchanged where the commitments from the neighboring users are both *valid*. The following experiments will also modify these situations.

Game 6. 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} and c_{i+1} of which at least one was *adversary-generated*.

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

$$\mathsf{Adv}(\mathcal{A}, G_6) \geq \mathsf{Adv}(\mathcal{A}, G_5).$$

Game 7. Once an instance $\Pi_i^{s_i}$ has got all messages of the first round and c_{i-1} and c_{i+1} were both *oracle-generated*, then the instance will set its values $Z_{i,i-1}$ and $Z_{i,i+1}$ to random values from the group G and keeps the assignments $(c_{i-1}, S_{i-1}, c_i, S_i) \rightarrow Z_{i,i-1}$, $(c_i, S_i, c_{i+1}, S_{i+1}) \rightarrow Z_{i,i+1}$ in a list to assure consistency between corresponding instances.

Given an adversary $\mathcal{A}$ able to distinguish between Game 6 and Game 7 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.

At first, D will receive the common reference string ρ and fix a password pw. We construct D such that it then behaves like the simulator in Game 6, except that 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 the Hash oracle with the respective values c_{i-1} and c_{i+1} if both were *oracle-generated*, and continue as in Game 6 otherwise.

Now the view of $\mathcal{A}$ will be exactly as in Game 6 if D interacts with Exp-Hash and exactly as in Game 7 if D interacts with Exp-Unif.

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

Game 8. In Game 8, the Send_0 oracle is modified, so that c_i is not chosen as a commitment to the correct password pw, but a random password $\widetilde{pw}$ from the dictionary is chosen and the value $c_i = c(\widetilde{pw}, r_i)$ with r_i chosen uniformly at random is broadcast. This will not influence the further protocol run, as so far, all instances choose $Z_{i,i-1}$ and $Z_{i,i+1}$ at random in any case, the hash function is never computed and thus the commitments c_i never needed.

The adversary cannot detect the difference with more than negligible probability due to the hiding property of the commitment scheme.

$$|\mathsf{Adv}(\mathcal{A}, G_8) - \mathsf{Adv}(\mathcal{A}, G_7)| \leq \mathrm{negl}(\ell).$$

Game 9. 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 master key $K = (Z_{1,2}, \ldots, Z_{n,1})$ has, once the X_i are public, sufficient entropy such that the output of the pseudorandom function $F_{UH(K)}$ is distinguishable from a random $\mathsf{sk}_i^{s_i}$ with negligible probability only.

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

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 $UH(K)$ 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. 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 and the probability to win in Game 9 is

$$\mathsf{Succ}(\mathcal{A}, G_9) = \frac{q}{|D|} + \frac{1}{2}\left(1 - \frac{q}{|D|}\right),$$

giving an advantage of

$$\mathsf{Adv}(\mathcal{A}, G_9) = \frac{q}{|D|}.$$

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

Putting everything together we have

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

□

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 [12], 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 [18]. 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.

As it is the case with 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 smooth 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. Building an specific example based on the Decisional Diffie-Hellman assumption is therefore straightforward using the encryption scheme of Cramer and Shoup [10].

Following an observation of Abdalla [1], in joint work we are currently exploring to what extent ideas in the above protocol are useful in another setting: We hope 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.

Acknowledgment

We thank Michel Abdalla for helpful discussions and bringing [3] to our attention.

References

1. Michel Abdalla. Personal communication, June 2006.
2. 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.
3. Michel Abdalla et al. A Scalable Password-based Group Key Exchange Protocol in the Standard Model. Submitted.
4. Ratna Dutta amd Rana Barua. Password-Based Encrypted Group Key Agreement. *International Journal of Network Security*, 3(1):23–34, July 2006.
5. 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.
6. 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.
7. Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. Secure Group Key Establishment Revisited. Cryptology ePrint Archive, Report 2005/395, 2005. `http://eprint.iacr.org/2005/395/`.
8. 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.

9. 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.
10. Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.
11. 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.
12. 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`.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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`.
18. Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. Cryptology ePrint Archive: Report 2005/163, 2005. Available at `http://eprint.iacr.org/2005/163`.
19. 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.
20. Su Mi Lee, Jung Yeon Hwang, and Dong Hoon Lee. Efficient Password-Based Group Key Exchange. In Sokratis Katsikas, Javier Lopez, and Günther Pernul, editors, *Trust and Privacy in Digital Business: First International Conference, TrustBus 2004*, volume 3184 of *Lecture Notes in Computer Science*, pages 191–199. Springer, 2004.