Password-based Group Key Exchange in a Constant Number of Rounds

Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, and David Pointcheval

[1] Departement d'Informatique, École normale supérieure
45 Rue d'Ulm, 75230 Paris Cedex 05, France
{Michel.Abdalla,David.Pointcheval}@ens.fr
http://www.di.ens.fr/~{mabdalla,pointche}
[2] Cryptology Department, CELAR, 35174 Bruz, France
Emmanuel.Bresson@polytechnique.org, http://www.di.ens.fr/~bresson
[3] Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
OChevassut@lbl.gov, http://www.dsd.lbl.gov/~chevassu

Abstract. With the development of grids, distributed applications are spread across multiple computing resources and require efficient security mechanisms among the processes. Although protocols for authenticated group Diffie-Hellman key exchange protocols seem to be the natural mechanisms for supporting these applications, current solutions are either limited by the use of public key infrastructures or by their scalability, requiring a number of rounds linear in the number of group members. To overcome these shortcomings, we propose in this paper the first provably-secure password-based constant-round group key exchange protocol. It is based on the protocol of Burmester and Desmedt and is *provably-secure* in the random-oracle and ideal-cipher models, under the Decisional Diffie-Hellman assumption. The new protocol is very efficient and fully scalable since it only requires four rounds of communication and four multi-exponentiations per user. Moreover, the new protocol avoids intricate authentication infrastructures by relying on passwords for authentication.

Keywords. Password-based Authentication, Group Key Exchange.

1 Introduction

Motivation. Modern distributed applications often need to maintain consistency of replicated information and coordinate the activities of many processes. Collaborative applications and distributed computations are both examples of these types of applications. With the development of grids [12], distributed computations are spread across multiple computing resources requiring efficient security mechanisms between the processes. Although protocols for group Diffie-Hellman key exchange [5, 7, 6, 8] provide a natural mechanism for supporting these applications, these protocols are limited in their scalability due to a number of rounds linear in the number of group members. An alternative is to use a protocol for group key exchange that runs in a constant number or rounds [11, 15, 16]. The two measures of a protocol's efficiency are the computational cost per member and the communication complexity (number of protocol rounds) of the given protocol. Since the Moore's laws has told us that computing power grows faster than communication power, it is therefore natural to trade communication power for computing power in a group key exchange protocol.

A password is the ideal authentication means to exchange a session key in the absence of public-key infrastructures or pre-distributed symmetric keys. In a group, the sharing of a password among the members greatly simplifies the setup of distributed applications [7, 11]. An example of distributed applications could simply be the networking of all the devices attached to a human. Low-entropy passwords are easy for humans to remember, but cannot of course guarantee the same level of security as high-entropy secrets such as symmetric or asymmetric keys. The most serious attack against a password-based protocol is the so-called *dictionary attack*: the attacker recovers the password and uses it to impersonate the legitimate user. The low-entropy feature makes the job of the attacker easier since the attacker (off-line) runs through all the possible passwords in order to obtain partial information and to maximize his success probability. The minimum required from a protocol is security against this attack.

Contributions. In the present paper, we study the problem of scalable protocols for authenticated group Diffie-Hellman key exchange. Many researchers have studied and found solutions to this problem in the context of a Public-Key Infrastructure (PKI), yet a (secure) solution had to be found in the context of a (short) password shared among the members of the group. Two attempts in this direction are due to Dutta and Barua [11] and to Lee, Hwang, and Lee [17]. Unfortunately, adding authentication services to a group key exchange protocol is a not trivial since redundancy in the flows of the protocol can open the door to different forms of attacks. In fact, in Section 3, we briefly describe attacks against the schemes of Dutta and Barua [11] and of Lee, Hwang, and Lee [17]. Then, in Section 4, we show how to add password-authentication services to the Burmester and Desmedt scheme [9, 10]. Our protocol is *provably secure* in the random-oracle [4] and ideal-cipher models [3] under the Decisional Diffie-Hellman assumption.

Related Work. Following the work of Bresson et al. on the group Diffie-Hellman key exchange problem [5, 7, 6, 8], several researchers have developed similar protocols but that run in a constant number of rounds. Katz and Yung [15] added authentication services to the original Burmester and Desmedt's protocol [9, 10]. Later, Kim, Lee and Lee extended the work of Katz and Yung to take into account the notion of dynamicity in the membership [16]. The problem of adding password-authentication services followed shortly after. In [7], Bresson et al. proposed the first solution to the group Diffie-Hellman key exchange problem in the password-based scenario. Their protocol, however, has a total number of rounds which is linear in the total number of players in the group. In [11, 17], two different password-based versions of Burmester-Desmedt protocol were proposed along with proofs in the random-oracle and ideal-cipher models. Unfortunately, the latter two schemes are not secure.

Outline of the paper. The paper is organized as follows. In Section 2, we recall the security model usually used for password-based group Diffie-Hellman key exchange. This model was previously defined in [7], but also takes advantage of [1]. In Section 3 we recall Burmester-Desmedt scheme and describe attacks against the schemes of Dutta and Barua [11] and of Lee, Hwang, and Lee [17]. In Section 4, we describe the mechanics behind our protocol. In Section 5, we show that our protocol is *provably-secure* in the random-oracle and ideal-cipher models under the Decisional Diffie-Hellman assumption.

2 Security Model

2.1 Password-Based Authentication

In the password-based authentication setting, we assume each player holds a password pw drawn uniformly at random from the dictionary $\mathsf{Password}$ of size N. This secret of low-entropy (N is often assumed to be small, *i.e.* typically less than a million) will be used to authenticate the parties to each other

Unfortunately, one cannot prevent an adversary to choose randomly a password in the dictionary and to try to impersonate a player. However such on-line exhaustive search (even if N is not so large) can easily be *limited* by requiring a minimal time interval between successive failed attempts or locking an account after a threshold of failures. Security against such active attacks is measured in the number of passwords the adversary can "erase" from the candidate list after a failure.

On the other hand, off-line exhaustive search cannot be limited by such practical behaviors or computational resources considerations. Hopefully, they can be *prevented* if the protocol is carefully designed and ensures that no information about the password can leak from passively eavesdropped transcripts, but also from active attacks.

2.2 Formal Definitions

We denote by $U_1, \ldots, U_n$ the parties that can participate in the key exchange protocol P. Each of them may have several *instances* called oracles involved in distinct, possibly concurrent, executions of P. We denote U_i instances by U_i^j. The parties share a low-entropy secret pw which is uniformly drawn from a small dictionary $\mathsf{Password}$ of size N.

The key exchange algorithm P is an interactive protocol between the U_i's that provides the instances with a session key sk. During the execution of this protocol, the adversary has the entire control of the network, and tries to break the privacy of the key.

Remark 1 In the "constant-round" protocols that we will study, simultaneous broadcasts are intensively used. However we do not make any assumption about the correctness of the latter primitive: it is actually a multi-cast, in which the adversary may delay, modify, or cancel the message sent to each recipient independently.

In the usual security model [7], several queries are available to the adversary to model his capability. We however enhance it with the Real-or-Random notion for the semantic security [1] instead of the Find-then-Guess. This notion is strictly stronger in the password-based setting. And actually, since we focus on the semantic security only, we can assume that each time a player accepts a key, the latter is revealed to the adversary, either in a real way, or in a random one (according to a bit b). Let us briefly review each query:

- $\mathsf{Send}(U_i^j, m)$: This query enables to consider active attacks by having $\mathcal{A}$ sending a message to any instance U_i^j. The adversary $\mathcal{A}$ gets back the response U_i^j generates in processing the message m according to the protocol P. A query $\mathsf{Send}(\mathtt{Start})$ initializes the key exchange algorithm, and thus the adversary receives the initial flows sent out by the instance.
- $\mathsf{Test}^b(U_i^j)$: This query models the misuse of the session key by instance U_i (*known-key attacks*). The query is only available to $\mathcal{A}$ if the attacked instance actually "holds" a session key. It either releases the actual key to $\mathcal{A}$, if $b = 1$ or a random one, if $b = 0$. The random keys must however be consistant between users in the same session. Therefore, a random key is simulated by the evaluation of a random function on the view a user has of the session: all the partners have the same view, they thus have the same random key (but independent of the actual view.)

  Remark 2 Note that it has been shown [1] that this query is indeed enough to model *known-key attacks* —where Reveal queries, which always answer with the real keys, are available—, and makes the model even stronger. Even though their result has only been proven in the two-party and three-party scenarios, one should note that their proof can be easily extended to the group scenario.

As already noticed, the aim of the adversary is to break the privacy of the session key (a.k.a., semantic security). This security notion takes place in the context of executing P in the presence of the adversary $\mathcal{A}$. One first draws a password pw from Password, flips a coin b, provides coin tosses to $\mathcal{A}$, as well as access to the Test^b and Send oracles.

The goal of the adversary is to guess the bit b involved in the Test queries, by outputting this guess b'. We denote the **AKE advantage** as the probability that $\mathcal{A}$ correctly guesses the value of b. More precisely we define $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$. The protocol P is said to be (t, ϵ)-**AKE-secure** if $\mathcal{A}$'s advantage is smaller than ϵ for any adversary $\mathcal{A}$ running with time t.

2.3 On the Simplification of the Model

In previous models, Execute queries were introduced to model passive eavesdropping. However, they can easily be simulated using the Send queries. In our analysis, we refine the way to deal with the adversary possible behaviors. We will denote by q_{active} the number of messages the adversary produced by himself (thus without including those he has just forwarded). This number upper-bounds the number of on-line "tests" the adversary performs to guess the password. And we denote by q_{session} the total number of sessions the adversary has initiated: nq_{session}, where n is the size of the group, upper-bounds the total number of messages the adversary has sent in the protocol (including those he has built and those he has just forwarded). We emphasize that this is stronger than considering only Execute and Send queries: while being polynomially equivalent, the two models are not tightly equivalent, since the adversary does not need to know in advance if he will forward all the flows, or be active when a new session starts. Moreover, suppressing the Execute queries makes the model even simpler.

The best we can expect with such a scheme is that the adversary erases no more than 1 password for each *session* in which he plays actively (since there exists attacks which achieve that in any password-based scheme.) However, in our quite efficient scheme, we can just prevent the adversary from erasing more than 1 password for each *player* he tries to impersonate (we will even show our proof is almost optimal.)

3 Preliminaries

The best starting point for an efficient password-based group key exchange, and namely if one wants a constant-round protocol, is the scheme proposed by Burmester and Desmedt [9, 10] at Eurocrypt 94 and later formally analyzed by Katz and Yung in 2003 [15].

3.1 The Burmester and Desmedt Protocol

In the Burmester-Desmedt scheme, one considers a cyclic group $\mathbb{G}$ generated by g, in which the Decisional Diffie-Hellman (DDH) assumption holds. The protocol works as follows, where all the indices are taken modulo n (between 1 and n), and n is the size of the group:

- Each player U_i chooses a random exponent x_i and broadcasts $z_i = g^{x_i}$;
- Each player computes the $Z_i = z_{i-1}^{x_i}$ and $Z_{i+1} = z_i^{x_{i+1}} = z_{i+1}^{x_i}$, and broadcasts $X_i = Z_{i+1}/Z_i$;
- Each player computes his session key as $K_i = Z_i^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2}$.

It is easy to see that for any i, we have $K_i = \prod_{j=1}^{j=n} Z_j = g^{x_1x_2+x_2x_3+\cdots+x_nx_1}$.

3.2 A Naive Password-Based Approach

We immediately note that encrypting values in the second round would lead to a trivial dictionary attack, since the product of all values is equal to 1. One may want to enhance the Burmester and Desmedt's protocol by using a password pw to "mask" the first round only. One then comes up to the simple protocole, using a mask of the form h^{pw}, where h is another generator of the group $\mathbb{G}$, whose discrete logarithm in the base g is unknown [2]:

- Each player U_i chooses a random exponent x_i, computes $z_i = g^{x_i}$ and broadcasts $z_i^\star = z_i h^{pw}$;
- Each player extracts z_{i-1} and z_{i+1}, and computes the $Z_i = z_{i-1}^{x_i}$ and $Z_{i+1} = z_i^{x_{i+1}} = z_{i+1}^{x_i}$. He then broadcasts $X_i = Z_{i+1}/Z_i$;
- Each player computes his secret as $K_i = Z_i^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2}$

Thereafter, one can add any key confirmation and/or any intricate key extraction (even in the random oracle model, such as $sk_i = \mathcal{H}(\mathsf{View}, K_i)$), but it does not help. Indeed, the homomorphic property of this "masking" technique allows active attacks from the adversary: Assume that the adversary impersonates players U_1 and U_3 and sends for the first round $z_1^\star = g^{u_1}$ and $z_3^\star = g^{u_3}$, for known values u_1 and u_3. On the second round, the adversary waits for receiving X_2 from player U_2:

$$X_2 = \left(\frac{z_3}{z_1}\right)^{x_2} = g^{x_2(u_3-u_1)} = \left(\frac{z_2}{h^{pw}}\right)^{u_3-u_1}.$$

Then one knows that $h^{pw} = z_2/X_2^{(u_1-u_3)^{-1}}$, which can be easily checked off-line: a dictionary attack.

Furthermore, one can be easily convinced that any mechanism such as proof of knowledge, commitments, etc. to "enforce" the adversary to properly construct his values are useless against this attack, since in the above attack, the adversary plays "honestly".

3.3 The Dutta and Barua Protocol

Dutta and Barua [11] proposed a variant of the Kim-Lee-Lee protocol [16] presented at Asiacrypt '04. It makes use of the ideal-cipher model, instead of a simple mask as above, and is claimed to be secure against dictionary attacks:

- Each player U_i chooses a random exponent x_i, as well as a random key k_i, computes $z_i = g^{x_i}$, and broadcasts $z_i^\star = \mathcal{E}_{pw}(z_i)$;
- Each player extracts z_{i-1} and z_{i+1}, and computes the $K_i^L = \mathcal{H}(z_{i-1}^{x_i}) = \mathcal{H}(g^{x_{i-1}x_i})$ and $K_i^R = \mathcal{H}(z_i^{x_{i+1}}) = \mathcal{H}(z_{i+1}^{x_i}) = \mathcal{H}(g^{x_i x_{i+1}})$. For $i = 1, \ldots, n-1$, U_i computes $X_i = K_i^L \oplus K_i^R$, while U_n computes $X_n = k_n \oplus K_n^R$; For $i = 1, \ldots, n-1$, U_i broadcasts $\mathcal{E}'_{pw}(k_i \| X_i)$, while U_n broadcasts $\mathcal{E}''(X_n)$;
- After decryption, they can all recover all the k_i, and then the common session key is set as $sk = \mathcal{H}(k_1 \| \ldots \| k_n)$.

Unfortunately, their protocol contains another source of redundancy that can be exploited by an attacker: the encryption algorithm of all users use the password as their encryption key. Therefore, a simple attack against their scheme runs as follows: the adversary plays the role of user U_3, with honest users U_1 and U_2. When the adversary receives $z_1^\star = \mathcal{E}_{pw}(z_1)$ and $z_2^\star = \mathcal{E}_{pw}(z_2)$, he sets $z_3^\star = Z_1^\star$, sends it to users U_1 and U_2, and waits for their responses. Note that setting $z_3^\star = Z_1^\star$ implicitly sets $x_3 = x_1$. At this point, the adversary knows that $K_2^L = \mathcal{H}(g^{x_1 x_2})$ and $K_2^R = \mathcal{H}(g^{x_2 x_3}) = \mathcal{H}(g^{x_1 x_2})$, and thus $X_2 = 0^k$ (where k is the output length of the function $\mathcal{H}$). Upon receiving $\mathcal{E}'_{pw}(k_2\|X_2)$ from U_2, he can perform an off-line dictionary attack that immediately leads to the correct password, since this will be the only one decrypting this value to $k_2\|0^k$.

This confirms the fact that converting a provably-secure scheme into a password-based protocol is not a simple task. The main problem we observe with the above scheme is the unique way in which the initial messages of all users are encrypted, allowing attacks where one player can easily replay messages from another player. Thus, to avoid problems such as these, one should at least make sure that the encryption key used by each user is unique to that user. In fact, this is one of the features of the protocol that we present in the next section.

3.4 The Lee-Hwang-Lee Protocol

In [17], Lee, Hwang, and Lee proposed another password-based version of the Burmester-Desmedt protocol, which makes use of the random-oracle and ideal-cipher models. Let $\mathcal{E}$ be an ideal cipher and let $\mathcal{H}$ and $\mathcal{H}'$ be random oracles. Their protocol works as follows:

- Each player U_i chooses a random exponent x_i, computes $z_i = g^{x_i}$, and broadcasts $(U_i, z_i^\star = \mathcal{E}_{pw}(z_i))$;
- Each player U_i extracts z_{i-1} and z_{i+1}, computes $K_i = \mathcal{H}(z_{i+1}^{x_i}) = \mathcal{H}(g^{x_i x_{i+1}})$, $K_{i-1} = \mathcal{H}(z_{i-1}^{x_i}) = \mathcal{H}(g^{x_{i-1} x_i})$, $w_i = K_{i-1} \oplus K_i$, and broadcasts (U_i, w_i).
- Each player U_i first computes the values $K_j = \mathcal{H}(g^{x_{j-1} x_j})$ for $j = 1, \ldots, n$, using the values w_j that were broadcasted in the second round. Next, each player U_i sets $sk = \mathcal{H}'(\mathcal{H}(g^{x_1 x_2})\| \ldots \|\mathcal{H}(g^{x_{n-1} x_n})\|\mathcal{H}(g^{x_n x_1}))$ as the common session key.

To show that the protocol above is not secure, we present the following simple attack against the semantic security of the session key. First, we start two sessions with player U_1 using $\{U_1, \ldots, U_4\}$ as the group. Let x_1 and x'_1 be the corresponding values chosen by the two instances of player U_1 in each of these sessions and let $(U_1, z_1^\star = \mathcal{E}_{pw}(g^{x_1}))$ and $(U_1, {z'}_1^\star = \mathcal{E}_{pw}(g^{x'_1}))$ be the corresponding values outputted by these instances. For the instance that outputted $(U_1, z_1^\star)$, we provide to it the values $(U_2, {z'}_1^\star)$, $(U_3, z_1^\star)$, and $(U_4, {z'}_1^\star)$, as the first-round messages of players U_2, U_3, and U_4. This implicitly makes $K_1 = K_2 = K_3 = K_4 = \mathcal{H}(g^{x'_1 x_1})$. Likewise, for the instance that outputted $(U_1, z_1'^\star)$, we provide to it the values $(U_2, z_1^\star)$, $(U_3, {z'}_1^\star)$, and $(U_4, z_1^\star)$, as the first-round messages of players U_2, U_3, and U_4. This implicitly makes $K'_1 = K'_2 = K'_3 = K'_4 = \mathcal{H}(g^{x'_1 x_1})$. As a result, $w_1 = w_2 = w_3 = w_4 = 0$ and $w'_1 = w'_2 = w'_3 = w'_4 = 0$ and, thus, we can easily compute the appropriate second-round messages for players U_2, U_3, and U_4 in both sessions. Moreover, the session keys of these two sessions are the same. Thus, we can ask test queries to both instances of player U_1 and check whether we get back the same value. This should be the case whenever the output of test oracle is the actual session key.

4 Our protocol

As above, we use the ideal-cipher model. The latter considers a family of random permutations $\mathcal{E}_k : \mathbb{G} \rightarrow \mathbb{G}$ indexed by a $\ell_{\mathcal{H}}$-bit key k which are accessible (as well as their inverses) through oracle queries ($\mathcal{E}$ and $\mathcal{D}$). Here we use the password, together with nonces, and the index of the user, to encrypt the values in the first round. Other values are sent in the clear. Also a preliminary round is used during which each player chooses random nonces to be used. This will be crucial to define sessions, and then link the encrypted values all together.

Key generations (for the symmetric encryption $\mathcal{E}$, and for the session key) will make use of hash functions $\mathcal{H} : \{0,1\}^\star \rightarrow \{0,1\}^{\ell_{\mathcal{H}}}$ and $\mathcal{G} : \{0,1\}^\star \rightarrow \{0,1\}^{\ell_{\mathcal{G}}}$. Key confirmations will apply the function $\mathcal{A}uth : \{0,1\}^\star \rightarrow \{0,1\}^{\ell_{\mathcal{A}uth}}$.

4.1 Description

The protocol runs as follows:

1. Each player U_i chooses a random nonce N_i and broadcasts (U_i, N_i);
2. The session $S = U_1 \| N_1 \| \ldots \| U_i \| N_i \ldots \| U_n \| N_n$ is then defined, in which each player has a specific index i, and a specific symmetric key $k_i = \mathcal{H}(S, i, pw)$. Each player U_i chooses a random exponent x_i and broadcasts $z_i^\star = \mathcal{E}_{k_i}(z_i)$, where $z_i = g^{x_i}$;
3. Each player extracts $z_{i-1} = \mathcal{D}_{k_{i-1}}(z_{i-1}^\star)$ and $z_{i+1} = \mathcal{D}_{k_{i+1}}(z_{i+1}^\star)$, and computes the $Z_i = z_{i-1}^{x_i}$ and $Z_{i+1} = z_i^{x_{i+1}} = z_{i+1}^{x_i}$. He then broadcasts $X_i = Z_{i+1}/Z_i$;
4. Each player computes his secret as $K_i = Z_i^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2}$, and broadcasts his key confirmation $\mathcal{A}uth_i = \mathcal{A}uth(S, \{z_j^\star, X_j\}_j, K_i, i)$.
5. After having received and checked all the key confirmations, each player defined is session key as $sk_i = \mathcal{G}(S, \{z_j^\star, X_j, \mathcal{A}uth_j\}_j, K_i)$.

4.2 Security Theorem

Here we present the main security result of this paper, whose proof appears in Section 5.

Theorem 3 *Let P the above protocol in which the password is chosen in a dictionary of size N. Then for any adversary $\mathcal{A}$ running in time t, that makes at most q_{active} attempts within at most qSession sessions, his advantage in breaking the semantic security of the session key, in the ideal-cipher model, is upper-bounded by:*

$$\mathsf{Adv}_P^{\mathsf{ake}}(t) \leq \frac{2q_{\mathsf{active}}}{N} + 4q_{\mathsf{session}} n \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t) + \frac{2q_{\mathcal{G}}^2}{2^{\ell_{\mathcal{G}}}} + \frac{2q_{\mathcal{A}uth}^2}{2^{\ell_{\mathcal{A}uth}}} + \frac{8q_{\mathcal{G}} + 2q_{\mathcal{A}uth} + 2q_{\mathcal{D}} + 2nq_{\mathcal{E}}q_{\mathsf{session}} + (q_{\mathcal{E}} + q_{\mathcal{D}})^2}{|\mathbb{G}|} + \frac{2q_{\mathcal{H}}(q_{\mathcal{H}} + q_{\mathcal{D}})}{2^{\ell_{\mathcal{H}}}}$$

where $q_{\mathcal{G}}, q_{\mathcal{H}}, q_{\mathcal{A}uth}, q_{\mathcal{E}}, q_{\mathcal{D}}$ denote the number of oracle queries the adversary is allowed to make to the random oracles $\mathcal{G}$, $\mathcal{H}$ and $\mathcal{A}uth$, and to the ideal-cipher oracles $\mathcal{E}$ and $\mathcal{D}$, respectively.

This theorem states that the security of the session key is protected against dictionary attacks: the advantage of the adversary essentially grows linearly with the number of *active attempts* that the adversary makes (i.e., the number of messages that the adversary builds by himself). While the number of *sessions* includes both active attacks and passive ones (i.e., the session transcripts $\mathcal{A}$ passively eavesdropped), the theorem shows that these passive attacks are essentially negligible: a honest transcript does not help a computationally bounded adversary in guessing the password.

4.3 On the tightness of Theorem 3

Clearly, Theorem 3 ensures that when building a message by himself, the adversary cannot "test" more than one password per message. Actually, in the proof, we use q_{active} to upper-bound the number of players the adversary tries to impersonate and thus the number of different passwords he can inject. Hence, we achieve a stronger security result than the one claimed in Theorem 3. However, it leaves open the possibility of whether an adversary can test several passwords in the same session. Since one may wonder whether a security proof with a tighter reduction could be found, here we present an online dictionary attack against our scheme that shows that this is not the case. More precisely, we exhibit an online dictionary attack in which the adversary can test several passwords in the same session (but still no more than one password for each message!). The idea behind the attack is to create a session in which the number of dishonest players (whose roles are played by the adversary) is twice the number of honest players and to surround each of the honest players with two dishonest players.

Let k be the number of honest players. The attack works as follows. First, the adversary starts a session in which all the honest players have indices of the form $3(i-1)+2$ for $i = 1, \ldots, k$. Then, let $\{pw_1, \ldots, pw_k\}$ be a list of candidate passwords that an adversary wants to try and let $i' = 3(i-1)$. To test whether pw_i for $i = 1, \ldots, k$ is the correct password, the adversary plays the role of players $U_{i'+1}$ and $U_{i'+3}$ and follows the protocol using pw_i as the password. That is, he chooses random exponents $x_{i'+1}$ and $x_{i'+3}$, computes the values $z_{i'+1} = g^{x_{i'+1}}$ and $z_{i'+3} = g^{x_{i'+3}}$, and then computes $z^\star_{i'+1}$ and $z^\star_{i'+3}$ from $z_{i'+1}$ and $z_{i'+3}$ using pw_i as the password. Let $X_{i'+2}$ be the value that the honest user $U_{i'+2}$ outputs in the third round of our protocol. To verify if his guess pw_i for the password is the correct one, the adversary computes $z_{i'+2}$ from $z^\star_{i'+2}$ using pw_i as the password and checks whether $z_{i'+2}^{x_{i'+3}-x_{i'+1}} = X_{i'+2}$. This should be the case whenever pw_i is equal to the actual password.

4.4 Computational Assumptions

Decisional Diffie-Hellman assumption: DDH. The DDH assumption states (roughly) that the distributions (g^u, g^v, g^{uv}) and (g^u, g^v, g^w) are computationally indistinguishable when u, v, w are indices chosen uniformly at random. This can be made more precise by defining two experiments, $\mathsf{DDH}^\star$ and $\mathsf{DDH}^\$$. In experiment $\mathsf{DDH}^\star$, the inputs given to the adversary are $U = g^u$, $V = g^v$, and

$W = g^{uv}$, where u and v are two random indices. In experiment $\mathsf{DDH}^{\$}$, the inputs given to the adversary are $U = g^u$, $V = g^v$, and $W = g^w$, where u, v, and w are random indices. The goal of the adversary is to guess a bit indicating the experiment he thinks he is in. A (t, ϵ)-distinguisher against DDH for $\mathbb{G}$ is a probabilistic Turing machine Δ with time-complexity t, which is able to distinguish these two distributions with an advantage $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\Delta)$ greater than ϵ. The **advantage function** $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)$ for the group $\mathbb{G}$ is then defined as the maximum value of $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\Delta)$ over all Δ with time-complexity at most t.

Parallel Decisional Diffie-Hellman assumption: PDDH. We define a variant of the DDH problem, we name it the *Parallel Decisional Diffie-Hellman* problem, which is equivalent to the usual DDH problem. To this aim, we define the two following distributions:

$$\mathsf{PDH}_n^{\star} = \{g^{x_1}, \ldots, g^{x_n}, g^{x_1 x_2}, \ldots, g^{x_{n-1}x_n}, g^{x_n x_1} \mid x_1, \ldots, x_n \in_R \mathbb{Z}_q\},$$
$$\mathsf{PDH}_n^{\$} = \{g^{x_1}, \ldots, g^{x_n}, g^{y_1}, \ldots, g^{y_n} \mid x_1, \ldots, x_n, y_1, \ldots, y_n \in_R \mathbb{Z}_q\}.$$

A (t, ϵ)-distinguisher against PDDH_n for $\mathbb{G}$ is a probabilistic Turing machine Δ with time-complexity t, which is able to distinguish these two distributions with an advantage $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{pddh}_n}(\Delta)$ greater than ϵ. The **advantage function** $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{pddh}_n}(t)$ for the group $\mathbb{G}$, is then defined as the maximum value of $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{pddh}_n}(\Delta)$ over all Δ with time-complexity at most t.

Lemma 4 (Equivalence between PDDH_n and DDH) *For any group $\mathbb{G}$ and any integer n, the PDDH_n and the DDH problems are equivalent: for any time bound T,*

$$\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(T) \leq \mathsf{Adv}_{\mathbb{G}}^{\mathsf{pddh}_n}(T) \leq n\,\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(T).$$

Proof. We omit the proof of this lemma in this version of the paper as it follows from a standard hybrid argument [13, 14] with $n+1$ hybrid experiments, in which the first i DDH values are replaced by random ones in the i-th hybrid experiment for $i \in \{0, \ldots, n\}$. In fact, a proof of this lemma was implicitly made in the proceedings version of the paper by Katz and Yung in Crypto 2003 [15] when showing an upper bound for the probability distance between the experiments Fake_n and Real. Moreover, in the full version of their paper, they provide an even tighter security reduction between these two problems.

In our security analysis, we will need a challenger that outputs a new tuple either from $\mathsf{PDH}_n^{\star}$ or $\mathsf{PDH}_n^{\$}$, according to an input bit. That is, we have a fixed bit β, and for any new query S, $\mathsf{Chall}^{\beta}(S)$ outputs a new tuple from $\mathsf{PDH}_n^{\star}$ if $\beta = 0$, or from $\mathsf{PDH}_n^{\$}$ if $\beta = 1$. If the same S is queried again, then the same output tuple is returned. It is a well-known result that after q queries to the challenger, any adversary in time t cannot guess the bit β with advantage larger than $q \times \mathsf{Adv}_{\mathbb{G}}^{\mathsf{pddh}_n}(t) \leq qn \times \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)$.

5 Proof of Theorem 3

We proceed by defining several experiments (or *games*), the first one being the real-world experiment (in which the success of the adversary in outputting $b' = b$ — denoted by event S — is larger than $(1+\mathsf{Adv}^{\mathsf{ake}}(\mathcal{A}))/2$ by definition), the last one being a *trivially secure* experiment in which the success of the adversary is straightforwardly 1/2.

Game $\mathbf{G}_0$: This is the real attack game, in the random-oracle and ideal-cipher models.

Game $\mathbf{G}_1$: We simulate the random oracles $\mathcal{G}$, $\mathcal{H}$ and $\mathcal{A}uth$ in a classical way using the lists $\Lambda_{\mathcal{G}}$, $\Lambda_{\mathcal{H}}$ and $\Lambda_{\mathcal{A}uth}$, with a random value for any new query, and we cancel executions (by halting the simulation and declaring the adversary successful) in which a collision occurs in the output of hash functions. The probability of such bad event is upper-bounded by the birthday paradox.

$$\left|\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_0]\right| \leq \frac{q_{\mathcal{G}}^2}{2^{\ell_{\mathcal{G}}}} + \frac{q_{\mathcal{H}}^2}{2^{\ell_{\mathcal{H}}}} + \frac{q_{\mathcal{A}uth}^2}{2^{\ell_{\mathcal{A}uth}}}.$$

Game $\mathbf{G}_2$: In this game, we start to control the simulation of the ideal cipher by maintaining a list Λ that keeps track of the previous queries-answers and that links each query to a specific user. Members of the list Λ are of the form $(type, S, i, \alpha, k, z, z^\star)$, where $type \in \{\mathsf{enc}, \mathsf{dec}\}$. Such record means that $\mathcal{E}_k(z) = z^\star$, and $type$ indicates which kind of queries generated the record. The index i indicates which player is associated with the key k, while S indicates the session with which we are dealing. These values are both set to $\perp$ if k does not come from a $\mathcal{H}$ query of the form $(S, i, *)$ with $i \in \{1, \dots, n\}$, and S of any form. The element α will be explained later.

- On encryption query $\mathcal{E}_k(z)$, we look for a record $(\cdot, \cdot, \cdot, \cdot, k, z, *)$ in Λ. If such a record exists, we return its last component. Otherwise, we choose uniformly at random $z^\star \in \mathbb{G}$, add $(\mathsf{enc}, \perp, \perp, \perp, k, z, z^\star)$ to Λ, and return $z^\star$.
- On decryption query $\mathcal{D}_k(z^\star)$, we look for a record $(\cdot, \cdot, \cdot, \cdot, k, *, z^\star)$ in Λ. If such a record exists, we return its sixth component. Otherwise, we distinguish two sub-cases, by looking up in Λ_H if k has been returned to a hash query of the form $(S, i, *)$: if it the case, we choose z at random in $\mathbb{G}^\star = \mathbb{G}\backslash\{0\}$ and update the list Λ with $(\mathsf{dec}, S, i, \perp, k, z, z^\star)$; otherwise, we choose z at random in $\mathbb{G}^\star$ and update the list Λ with $(\mathsf{dec}, \perp, \perp, \perp, k, z, z^\star)$. In both cases, the decryption query on $z^\star$ is answered with z.

Such a simulation is perfect, except for the following three points. First, collisions may appear that contradict the permutation property of the ideal-cipher: the probability can be upper-bounded by $(q_{\mathcal{E}}+q_{\mathcal{D}})^2/2|\mathbb{G}|$. Second, we avoided z being equal to 1 in the decryption queries. Finally, in the case of the decryption query simulation, one will cancel executions (by halting the simulation and declaring the the adversary successful) if the value k (involved in a decryption query) is output later by $\mathcal{H}$. Fortunately, this happens with probability at most $q_H/2^{\ell_{\mathcal{H}}}$ for each decryption query. Intuitively, as it will become clear in the next games, we indeed want to make sure that, for any k involved in a decryption query, if k comes from a $\mathcal{H}$ query, we know the corresponding pair (S, i). All being considered, such bad events are unlikely:

$$\left|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]\right| \leq \frac{(q_{\mathcal{E}} + q_{\mathcal{D}})^2}{2|\mathbb{G}|} + \frac{q_{\mathcal{D}}}{|\mathbb{G}|} + \frac{q_{\mathcal{H}} q_{\mathcal{D}}}{2^{\ell_{\mathcal{H}}}}.$$

Game $\mathbf{G}_3$: In this game, we change the simulation of the decryption queries, and make use of our challenger to embed an instance of the PDH problem in the protocol simulation. In this game, we set $\beta = 0$, so that our challenger $\mathsf{Chall}^{\beta}(\cdot)$ output tuples $(\zeta_1, \ldots, \zeta_n, \gamma_1, \ldots, \gamma_n)$ according to the $\mathsf{PDH}_n^{\star}$ distribution. We use these $(2n)$-tuples to properly simulate the decryption queries.

More precisely, we issue a new tuple each time a new session S appears in a decryption query. But if several queries are asked with the same S, the challenger outputs the same tuple, so we will derive many related instances, granted the random self-reducibility. The latter tells us that, given one tuple outputted by the challenger, then for any randomly chosen $(\alpha_1, \ldots, \alpha_n)$, the tuple $(\zeta_1^{\alpha_1}, \ldots, \zeta_n^{\alpha_n}, \gamma_1^{\alpha_1\alpha_2}, \ldots, \gamma_n^{\alpha_n\alpha_1})$ has the same distribution as the original one.

We make use of this property as follows, by modifying the first sub-case previously considered for *new* decryption queries.

- On a new decryption query $\mathcal{D}_k(z^{\star})$, such that $k = \mathcal{H}(S, i, *)$ was previously obtained from $\mathcal{H}$ for some valid index i, we query $\mathsf{Chall}^{\beta}(S)$ in order to get a tuple $(\zeta_1, \ldots, \zeta_n, \gamma_1, \ldots, \gamma_n)$. We then randomly choose $\alpha \in \mathbb{Z}_q^{\star}$, add $(\mathsf{dec}, S, i, \alpha, k, z = \zeta_i^{\alpha}, z^{\star})$ to Λ, and return z.

Above, we have defined the list Λ whose elements are of the form $(type, S, i, \alpha, k, z, z^{\star})$. The component '$\alpha$' now comes into play. This element is an exponent indicating how we applied the random self-reducibility of the PDDH problem, to the instance generated by the challenger upon the request S: $X = \zeta_i^{\alpha}$. Here, the element α can only be defined if S and i are known (in order to know which tuple, and which ζ_i, we are working with.) If α is unknown to the simulator, we set $\alpha = \perp$.

This change does not modify the view of the adversary, so: $\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_2]$.

Game $\mathbf{G}_4$: We are now ready to simulate the Send queries in a different way, but only in the second and third rounds: when the session S is defined, user i computes the symmetric keys as before $k_j = \mathcal{H}(S, j, pw)$, for all j. We thus know we are working with the tuple $(\zeta_1, \ldots, \zeta_n, \gamma_1, \ldots, \gamma_n)$.

In the second round, U_i randomly chooses a value $z_i^{\star} \in \mathbb{G}$ to be broadcasted, and asks $z_i = \mathcal{D}_{k_i}(z_i^{\star})$, using the above simulation (which leads to add α_i to the list Λ, unless $z_i^{\star}$ already appeared as an encryption result. But the latter event cannot happen with probability greater than $q_{\mathcal{E}}/|\mathbb{G}|$.)

In the third round, U_i recovers $z_{i-1} = \mathcal{D}_{k_{i-1}}(z_{i-1}^{\star})$ and $z_{i+1} = \mathcal{D}_{k_{i+1}}(z_{i+1}^{\star})$. But then, two situations may appear:

- $z_{i-1}^{\star}$ and $z_{i+1}^{\star}$ have been simulated according to the above simulation of the second round, and then one gets α_{i-1} and α_{i+1} in the list Λ such that $z_{i-1} = \zeta_{i-1}^{\alpha_{i-1}}$ and $z_{i+1} = \zeta_{i+1}^{\alpha_{i+1}}$;

– one of the $z_j^\star$ has been previously answered by the encryption oracle in response to an attacker query $\mathcal{E}_k(z^\star)$, where $k = \mathcal{H}(S, j, pw)$ is the correct key for player U_j in session S. We denote such an event by $\mathsf{Encrypt}$. In such a case, we stop the simulation, letting the adversary win.

If everything runs smoothly, one gets

$$z_i = \zeta_i^{\alpha_i} \quad z_{i-1} = \zeta_{i-1}^{\alpha_{i-1}} \quad z_{i+1} = \zeta_{i+1}^{\alpha_{i+1}}.$$

One can then correctly compute

$$Z_i = \mathsf{CDH}(z_{i-1}, z_i) = \gamma_{i-1}^{\alpha_{i-1}\alpha_i} \quad Z_{i+1} = \mathsf{CDH}(z_i, z_{i+1}) = \gamma_i^{\alpha_i \alpha_{i+1}}.$$

One then broadcasts $X_i = Z_{i+1}/Z_i$. After this final round, everybody can compute the session key as before. The simulation is still perfect, unless the above bad events happen:

$$\left| \Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_3] \right| \leq \frac{q_{\mathcal{E}} q_{\mathsf{passive}}}{|\mathbb{G}|} + \Pr[\mathsf{Encrypt}_4] \leq \frac{n q_{\mathcal{E}} q_{\mathsf{session}}}{|\mathbb{G}|} + \Pr[\mathsf{Encrypt}_4].$$

Game $\mathbf{G}_5$: Since it is clear that the security of the above scheme still relies on the DDH assumption, we now flip the bit β to 1, in order to receive tuples $(\zeta_1, \ldots, \zeta_n, \gamma_1, \ldots, \gamma_n)$ according to the $\mathsf{PDH}_n^\$$ distribution (in which the y_i's denote the values $\log_g \gamma_i$).

$$\left| \Pr[\mathsf{S}_5] - \Pr[\mathsf{S}_4] \right| \leq q_{\mathsf{session}} \mathsf{Adv}_{\mathbb{G}}^{\mathsf{pddh}_n}(t)$$
$$\left| \Pr[\mathsf{Encrypt}_5] - \Pr[\mathsf{Encrypt}_4] \right| \leq q_{\mathsf{session}} \mathsf{Adv}_{\mathbb{G}}^{\mathsf{pddh}_n}(t).$$

Game $\mathbf{G}_6$: In order to stop active attacks, where the adversary forges flows, we modify the computation of the key confirmations: we replace the function $\mathcal{A}uth$ by a private one $\mathcal{A}uth'$: $\mathcal{A}uth_i = \mathcal{A}uth'(S, \{z_j^\star, X_j\}_j, K_i, i)$, where

$$\begin{aligned} K_i &= Z_i^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2} = \gamma_{i-1}^{n\alpha_{i-1}\alpha_i} X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2} \\ &= g^{n(\alpha_{i-1}\alpha_i y_{i-1})} X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2}. \end{aligned}$$

Let us list all the information a (powerful) adversary may have, from all the X_j sent by U_j in the S-th session:

$$\log X_j = y_j(\alpha_j \alpha_{j+1}) - y_{j-1}(\alpha_{j-1}\alpha_j) = A_j y_j - A_{j-1} y_{j-1}.$$

As explained in [15], this does not leak any information about y_{i-1}, since the above system contains only $n-1$ independent equations with n unknowns. Any value for y_{n-1} is thus possible and would determine all the other values.

Therefore, after this modification, the probability for the adversary to see the difference between the current and the previous experiments is to query $\mathcal{A}uth(S, \{z_j^\star, X_j\}_j, K_i, i)$, which is upper-bounded by $q_{\mathcal{A}uth}/|\mathbb{G}|$.

$$\big|\Pr[\mathsf{S}_6] - \Pr[\mathsf{S}_5]\big| \leq \frac{q_{\mathcal{A}uth}}{|\mathbb{G}|} \qquad \big|\Pr[\mathsf{Encrypt}_6] - \Pr[\mathsf{Encrypt}_5]\big| \leq \frac{q_{\mathcal{A}uth}}{|\mathbb{G}|}.$$

Game $\mathbf{G}_7$: Finally, we now derive the session keys using a private random oracle $\mathcal{G}'$: $sk_i = \mathcal{G}'(S, \{z_j^\star, X_j, \mathcal{A}uth_j\}_j)$. As above, after the modification of the derivation of the session key, the probability for the adversary to see the difference between the current and the previous experiments is to query $\mathcal{G}(S, \{z_j^\star, X_j, \mathcal{A}uth_j\}_j, K_i)$. Since the previous game, we know that inside each session, all the honest users have the same view, and thus theses queries are identical: the probability of such an event can also be upper-bounded by $q_{\mathcal{G}}/|\mathbb{G}|$, since no information has been leaked about K_i (except it does not correspond to the $\mathcal{A}uth$ queries asked above.)

$$\big|\Pr[\mathsf{S}_7] - \Pr[\mathsf{S}_6]\big| \leq \frac{q_{\mathcal{G}}}{|\mathbb{G}| - q_{\mathcal{A}uth}} \leq \frac{2q_{\mathcal{G}}}{|\mathbb{G}|}$$

$$\big|\Pr[\mathsf{Encrypt}_7] - \Pr[\mathsf{Encrypt}_6]\big| \leq \frac{q_{\mathcal{G}}}{|\mathbb{G}| - q_{\mathcal{A}uth}} \leq \frac{2q_{\mathcal{G}}}{|\mathbb{G}|}.$$

Furthermore, because the private oracle $\mathcal{G}'$ is private to the simulator, it is clear that

$$\Pr[\mathsf{S}_7] = \frac{1}{2}.$$

Game $\mathbf{G}_8$: In order to conclude the proof, we need to upper-bound the event $\mathsf{Encrypt}_7$. One can note that the password pw is only used in the simulation of the second and third rounds, to compute z_i, z_{i-1} and z_{i+1} (using the elements ζ_i, ζ_{i-1} and ζ_{i+1}), but eventually, we output X_i only, which are computed from the γ_{i-1} and γ_i. The latter is totally independent of the former.

We can thus simplify the simulation of the second and third rounds: In the second round, U_i randomly chooses $z_i^\star \in \mathbb{G}$, and sends it (this is exactly as before.) However no decryption is needed. In the third round, U_i simply computes and sends $X_i = \gamma_i/\gamma_{i-1}$ (this is just to make sure that the product of the X_i is equal to 1, but we just need random elements satisfying this relation, since they do not appear anywhere else.) This is a perfect simulation, since one does not need anymore to compute K_i.

At this point, the password is never used, and can thus be chosen at the very end only, which makes clear that probability of the $\mathsf{Encrypt}$ event is less than the number of first flows manufactured by the adversary, divided by N. The latter part is upper-bounded by q_{active}:

$$\Pr[\mathsf{Encrypt}_7] = \Pr[\mathsf{Encrypt}_8] \leq q_{\mathsf{active}}/N.$$

In the above, we used the fact that collisions in the output of $\mathcal{H}$ have been eliminated in previous games.

Putting all equations together, one easily gets the announced bound.

6 Conclusion

We described a constant-round password-based key exchange protocol for group, derived from the Burmester-Desmedt scheme. The protocol is proven secure against dictionary attacks under the DDH assumption, in the ideal-cipher and random oracle models. It remains an open problem to find a scheme whose security depends on the number of active sessions rather than on the number of manufactured flows.

Acknowledgements

The first and fourth authors were supported in part by France Telecom R&D as part of the contract CIDRE, between France Telecom R&D and École normale supérieure. The third author was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Information and Computing Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. This document is report LBNL-59542. See `http://www-library.lbl.gov/disclaimer`.

References

1. Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84, Les Diablerets, Switzerland, January 23–26, 2005. Springer-Verlag, Berlin, Germany.
2. Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 191–208, San Francisco, CA, USA, February 14–18, 2005. Springer-Verlag, Berlin, Germany.
3. 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, Bruges, Belgium, May 14–18, 2000. Springer-Verlag, Berlin, Germany.
4. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption: How to encrypt with RSA. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Perugia, Italy, May 9–12, 1994. Springer-Verlag, Berlin, Germany. `http://www-cse.ucsd.edu/users/mihir`.
5. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably authenticated group Diffie-Hellman key exchange – the dynamic case. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 290–309, Gold Coast, Australia, December 9–13, 2001. Springer-Verlag, Berlin, Germany.

6. 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, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer-Verlag, Berlin, Germany.
7. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Group Diffie-Hellman key exchange secure against dictionary attacks. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 497–514, Queenstown, New Zealand, December 1–5, 2002. Springer-Verlag, Berlin, Germany.
8. Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 255–264, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.
9. Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system (extended abstract). In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286, Perugia, Italy, May 9–12, 1994. Springer-Verlag, Berlin, Germany.
10. Mike Burmester and Yvo Desmedt. A secure and scalable group key exchange system. *Information Processing Letters*, 94(3):137–143, May 2005.
11. Ratna Dutta and Rana Barua. Password-based encrypted group key agreement. *International Journal of Network Security*, 3(1):30–41, July 2006. `http://isrc.nchu.edu.tw/ijns`.
12. Ian T. Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.
13. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
14. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
15. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125, Santa Barbara, CA, USA, August 17–21, 2003. Springer-Verlag, Berlin, Germany.
16. Hyun-Jeong Kim, Su-Mi Lee, and Dong Hoon Lee. Constant-round authenticated group key exchange for dynamic groups. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 245–259, Jeju Island, Korea, December 5–9, 2004. Springer-Verlag, Berlin, Germany.
17. Su-Mi Lee, Jung Yeon Hwang, and Dong Hoon Lee. Efficient password-based group key exchange. In Sokratis K. Katsikas, Javier Lopez, and Günther Pernul, editors, *TrustBus 2004: Trust and Privacy in Digital Business, 1st International Conference*, volume 3184 of *Lecture Notes in Computer Science*, pages 191–199, Zaragoza, Spain, August 30 – September 1, 2004. Springer-Verlag, Berlin, Germany.