Password-Based Encrypted Group Key Agreement

Ratna Dutta and Rana Barua
(Corresponding author: Ratna Dutta)

Stat-Math Unit, Indian Statistical Institute
203, B.T. Road, Kolkata, India 700108 (Email: {ratna_r,rana}@isical.ac.in)

(Received Aug. 18, 2005; accepted Sept. 19, 2005)

Abstract

This paper presents an efficient password-based authenticated encrypted group key agreement protocol immune to dictionary attack under the computation Diffie-Hellman (CDH) assumption. In a password-based key agreement protocol, the users only share a human-memorable low entropy password; and using this low-entropy password, the users can agree upon a high-entropy session key which they may use to build a secure communication channel among themselves. While designing such protocols, one should limit the number of on-line password guessing and achieve the security against dictionary attack. Our protocol is obtained from the multi-party key agreement protocol of Kim *et al.* We analyze the security in the security model formalized by Bellare *et al.* following their proof techniques. Our proposed scheme achieves efficiency in both communication and computation aspects and is proven to be secure in both the ideal cipher model and the random oracle model.

Keywords: CDH problem, dictionary attack, encrypted group key agreement, password-based protocol

1 Introduction

A password-based group key agreement protocol allows a group of users, who share only a low entropy human-memorable password and communicating over a public network, to agree upon a high entropy session key among themselves. This session key can later be used to implement secure multi-cast sessions.

In common scenario, a user cannot remember long random numbers and he has to store it somewhere in his system. This requires an extra security burden and introduces a new point of weakness. Password-based protocols enable us to remove this weakness. These protocols require the users simply to remember a low quality human-memorable weak secret shared password (say, a 4 digit PIN number or a 6 character string) chosen uniformly from a relatively small dictionary rather than high quality symmetric encryption key. A password-based key agreement enables to establish a strong session key from a weak shared secret.

Password-based key agreement protocols can be useful for highly mobile environments, such as emergency rescue and military operations [44, 46, 56], conference/meeting [3, 46], personal networking [8, 26] etc. These applications require collaboration among a small group of users where the system lacks a fixed security infrastructure. The users share only a human-memorable low entropy password. Using this low entropy password, users can agree upon a high entropy session key by invoking a secure password-based group key agreement protocol and use this strong symmetric session key to build a secure communication channel among themselves.

Password-based protocols are associated with many security concerns, mainly because most user's passwords are drawn from a relatively small and easily generated dictionary. An adversary can try all possible combinations of secret keys in a given small set of values (i.e. the dictionary) using a brute-force method. This attack is not effective in case of high-entropy key, but can be very damaging when the secret key is a low-entropy password, because the attacker has a non-negligible chance of winning. This off-line exhaustive search is called dictionary attack. To achieve security against dictionary attack is a fundamental security goal in designing password-based authenticated key agreement protocols.

Another security attribute is on-line guessing attack or impersonation attack in which the adversary attempts to impersonate a user by guessing the password. If the attack fails, the adversary can eliminate this value from the set of possible passwords. For a real world adversary, such on-line attacks are hardest to prevent and easiest to detect. We make a realistic assumption that the number of on-line attacks an adversary can mount is severely limited; the adversary will not be able to eliminate (guess) more than one password after one active interaction with some user. We do not impose such restrictions on other attacks like off-line password guessing, eavesdropping etc.

Survey on Previous Work: The problem of password-based protocols has an extensive history. The first work to deal with off-line dictionary attacks is by Bellovin and Merritt [7]. They introduced a family of Encrypted Key Exchange (EKE) protocols to withstand dictionary attack. This work is very influential and become the basis for much future work in the area. Following it, a number of protocols for password-based key exchange have been proposed [25, 50], security of which are based on heuristic arguments. For a survey of works and techniques related to password-authentication, we refer to [24, 30, 42].

Halevi and Krawczyk [24] first considered a rigorous treatment of the security model for password-based authenticated protocol and provided a scheme formally proven to be secure under standard assumptions. Boyarsky [9] enhanced this protocol to make it secure in the multi-user scenario. The Halevi-Krawczyk model considered an asymmetric hybrid model in which one party (the server) may hold a high-quality key and the other party (the human) may hold a password. This model is inapplicable to settings in which communication has to be established among humans sharing a common low-entropy password. Recently, two formal models for password-based key exchange have been proposed. One by Bellare, Pointcheval and Rogaway [6] which is based on [4, 5] and a second by Boyko, MacKenzie and Patel [10] following [48].

Bellare et al. proposed a 2-party password-based protocol in [6]. An extension of this work to multi-party setting is presented by Bresson et al. [13]. Bresson et al. [12] examine the security of 2-party AuthA password-authenticated key exchange protocol standardized by IEEE P1363 Study Group and prove its security following the proof technique of [13]. The security of all these schemes are both in the random oracle and the ideal cipher model.

In the ideal cipher model, a keyed cipher is viewed as a family of random permutations that are queried via oracle to encrypt and decrypt. If the same query is asked twice, identical answers are provided and for each new query, a truly random value is prodeced by the oracle. In practice, any deterministic symmetric encryption function can instantiate the ideal cipher (see [6] for concrete constructions). For example, AES [43] can be used for this purpose. Ideal cipher model does not provide the same security guarantees as those in the random oracle and the standard models, but it is certainly superior to those provided by *ad-hoc* protocol designs. Reducing ideal-cipher model assumption is an interesting research problem.

Using the multi-party simulatibility technique [41], Boyko et al. [10] and MacKenzie et al. [38] presented protocols (2-party) which are secure in the random oracle model. Goldrich and Lindell [21] provided a protocol based on general assumptions and prove its security in the standard security model of [10]. Unfortunately, their protocol is much inefficient for practical use and does not allow concurrent executions. Independent of this work, Katz, Ostrovsky and Yung [28] suggested a highly efficient password-authenticated key exchange (2 party) protocol under standard DDH assumption in the security model of [6].

There are several works extending 2-party Diffie-Hellman key exchange protocols to multi-party setting. To design group key agreement protocols in password-based setting is another flavor of group key agreement. The only work in this area is, to the best of our knowledge, by Bresson *et al.* [13][1]. As already mentioned, their proposed scheme is secure in both the random oracle model and the ideal cipher model.

Our Contribution: In this work, we present a very efficient encrypted group key agreement protocol in password-based setting where the members of the group share only a human-memorable password and the system may not have any secure public key infrastructure.

In our construction, users communicate over an insecure public network and share only a low quality password among themselves. No secure private communication channel is assumed to be held by the users. The goal of our work is to agree upon a high quality common secret key among the users bootstrapped from this low quality password. Once the users agree upon a high quality secret session key, they can use this key to build secure multi cast channel among themselves and can communicate in a private and authenticated manner. The emphasis of this work is in protecting passwords against dictionary attacks that take advantage of low entropy of the chosen password.

Recently, Kim, Lee, Lee [31] presented an efficient constant round authenticated group key agreement protocol. Our password-based group key agreement is based on their protocol. It is not a trivial task to convert a provably authenticated group key agreement into a password-based group key agreement. The straight-forward conversion of the protocol of Kim et al. [31] by replacing the signature scheme by a symmetric encryption scheme using the password as secret key does not enable the protocol to regist off-line dictionary attacks due to presence of redundancy in the second round communication. To remove this redundancy, we make several modifications in the unauthenticated version of the protocol of [31] and then apply encryption-based authentication mechanism using the password as secret key to transform it into a secure password-based protocol that withstand dictionary attack. This yields our password-based scheme. We provide a concrete security analysis of our protocol in the security model formalized by Bellare et al. [6] under CDH assumption and security against dictionary attack is achieved in both the random oracle and the ideal cipher model.

Our proposed scheme is efficient from communication

[1] Very recently, we came to know about another password-based group key agreement by Lee et al. [34]. However, the protocol is not authenticated because there is no way to convince a user that the message that he receives is indeed comming from the intended participant.

point of view as it requires only 2 rounds and uses symmetric key encryption instead of signature for message authentication. This reduces the bandwidth of the messages sent and makes the protocol faster as compared to signature-based key agreement protocols. Thus the communication efficiency is increased.

On a more positive note, each group member performs at most 3 modular exponentiations, 4 one-way hash function evaluations, $n-1$ XOR operations, 2 encryptions and $n+1$ decryptions. The operations dependent on the number of group members are the XOR operation and symmetric key decryption operation. These highly reduce the total cost of computation as compared to other multi-party (password-based [13]) key agreement protocols [15, 31]. In contrast to the password-based group key agreement protocol of Bresson et al. [13], our protocol is applicable for large number of participants. The number of participants in the protocol of [13] is restricted to be small ($n \leq 100$); otherwise the protocol becomes impractical and inefficient as the number of rounds is linearly dependent on n. Our protocol being of constant round and highly efficient, does not require to impose such a restriction on n.

The rest of the paper is organized as follows. In Section 2, we recall some definitions and the security model for password-based group key agreement. Our protocol is presented in Section 3. The security analysis is focused in Section 4. In Section 5, we discuss how to incorporate mutual authentication to our protocol. We analyze the efficiency of our protocol in Section 6 and finally, conclude in Section 7.

2 Preliminaries

In this section, we define the Computation Diffie-Hellman (CDH) problem and describe the security notion that a password-based group key agreement protocol should achieve. We use the notation $a \in_R S$ to denote that a is chosen uniformly from the set S.

2.1 Computation Diffie-Hellman (CDH) Problem

Let $G = \langle g \rangle$ be a multiplicative group of some large prime order q. Then Computation Diffie-Hellman (CDH) problem in G is defined as follows:

Instance: (g, g^a, g^b) for some $a, b \in Z_q^*$.
Output: g^{ab}.

The success probability of any probabilistic, polynomial-time algorithm $\mathcal{A}$ in solving CDH problem in G is defined to be:

$$\mathsf{Succ}^{\mathsf{CDH}}_{G,\mathcal{A}} = \mathsf{Prob}[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \in_R Z_q^*].$$

CDH assumption: There exists no probabilistic, polynomial-time algorithm that (t, ϵ)-breaks CDH problem in G. In other words, for every probabilistic, polynomial-time algorithm $\mathcal{A}$, $\mathsf{Succ}^{\mathsf{CDH}}_{G,\mathcal{A}} \leq \epsilon$ for sufficiently small $\epsilon > 0$.

2.2 Security Model

We now briefly describe the formal security model of Bellare et al. [6] as standardized by Bresson et al. [12, 13] and refer the reader to [6, 12, 13] for more details.

A protocol P for password-based group key agreement assumes that there is a set $\mathcal{P} = \{U_1, U_2, \ldots, U_n\}$ of n users (n is fixed), who share a low entropy secret password pw drawn uniformly from a small dictionary of size N. The adversary is given control over all communication in the external network.

We assume that users do not deviate from the protocol and adversary never participates as a user in the protocol. This adversarial model allows concurrent execution of the protocol among n users. The interaction between the adversary $\mathcal{A}$ and the protocol participants occur only via oracle queries, which model the adversary's capabilities in a real attack. These queries are as follows (Π^i_U denotes the i-th instance of user U and sk^i_U denotes the session key after execution of the protocol by Π^i_U):

- $\mathsf{Send}(U, i, m)$: The adversary can carry out an active attack by this query. The adversary may intercept a message and then either modify it, create a new one or simply forward it to the intended participant. The output of the query is the reply (if any) generated by the instance Π^i_U upon receipt of message m. The adversary is allowed to prompt the unused instance Π^i_U to initiate the protocol by invoking $\mathsf{Send}(U, i, \text{“}\mathsf{Start}\text{”})$.
- $\mathsf{Execute}(\{(U_1, i_1), \ldots, (U_n, i_n)\})$: This query reflects the adversary's ability to passively eavesdrop on honest execution of password-based group key agreement protocol among unused instances $\Pi^{i_1}_{U_1}, \ldots, \Pi^{i_n}_{U_n}$ and outputs the transcript of the execution. A transcript consists of the messages that were exchanged during the honest execution of the protocol.
- $\mathsf{Reveal}(U, i)$: If a group key sk^i_U has previously been accepted by Π^i_U, then it is retured to the adversary unconditionally by Π^i_U. Otherwise a value NULL is returned. This query allows the adversary to learn session keys from previous and concurrent executions, modeling improper exposure of past session keys and insuring independence of different session keys in different execution.
- $\mathsf{Corrupt}(U)$: This query models the attacks resulting in the password pw to be revealed. $\mathcal{A}$ gets back from this query pw, but does not get any internal data on U. This is weak corruption model. There is another notion, called strong corruption model where the internal data of U also gets revealed on $\mathsf{Corrupt}(U)$ query. This query models the security level of forward secrecy which means the adversary does not

learn any information about previous established session key when making a Corrupt query.

– Test(U, i): This query is allowed only once, at any time during the adversary's execution. A bit $b \in \{0, 1\}$ is chosen uniformly at random. The adversary is given sk_U^i if $b = 1$, and a random session key if $b = 0$. This oracle measures the adversary's ability to distinguish a real session key from a random one.

Note that one cannot prevent the adversary to guess the password on-line because the passwords have low entropy. The Execute query does not carry out any on-line guessing attack since the adversary is passive there. So only the Send queries count such on-line password guess. Now if q_S is the maximum number of Send queries that an adversary may ask, then q_S represents the number of flows the adversary may have built by himself and thus the number of passwords he would have tried.

One can define the session identity sid_U^i for instance Π_U^i to be

$$\mathsf{sid}_U^i := S = \{(U_1, i_1), \ldots, (U_n, i_n)\}$$

such that $(U, i) \in S$ and $\Pi_{U_1}^{i_1}, \ldots, \Pi_{U_n}^{i_n}$ are involved in the session to agree upon a common key. The session identity uniquely identifies a session. Note that all the instances involved in a session have same session identity. We also assume that an instance of a user participates in at most one session. This means that the session identities of an instance for different sessions are mutually disjoint. We also define a boolean function acc_U^i which is set to be 1 by Π_U^i upon normal termination of the session and 0 otherwise.

The *correctness* of the protocol P means that if the instances $\Pi_{U_1}^{i_1}, \ldots, \Pi_{U_n}^{i_n}$ having same session identity accept, then the instances must conclude with the same session key. In other words, if $\mathsf{sid}_{U_j}^{i_j} = \mathsf{sid}_{U_k}^{i_k}$ for $1 \leq j, k \leq n$ such that $\mathsf{acc}_{U_j}^{i_j} = 1$ for $1 \leq j \leq n$, then $\mathsf{sk}_{U_j}^{i_j} = \mathsf{sk}_{U_k}^{i_k}$ for $1 \leq j, k \leq n$.

The Send, Execute, Reveal and Corrupt queries are asked several times, but Test query is asked only once and on a *fresh* instance. We say that an instance Π_U^i is *fresh* if (1) no Corrupt query has been made by the adversary since the begining of the game; and (2) Π_U^i has computed a session key and neither Π_U^i nor its partners have been asked a Reveal query (*i.e.* the adversary queried neither $\mathsf{Reveal}(U, i)$ nor $\mathsf{Reveal}(U', j)$ with $(U', j) \in \mathsf{sid}_U^i, U' \in \mathcal{P}$). A fresh instance should have computed a session key that is accepted by the instance.

Finally adversary outputs a guess bit b'. Such an adversary is said to win the game if $b = b'$ where b is the hidden bit used by the Test oracle.

Let Succ denote the event that the adversary $\mathcal{A}$ wins the game for a protocol P. We define

$$\mathsf{Adv}_{\mathcal{A},P} := |2\ \mathsf{Prob}[\mathsf{Succ}] - 1|$$

to be the advantage of the adversary $\mathcal{A}$ in attacking the protocol P where the probability space is over all the random coins of the adversary and all the oracles.

We denote by $\mathsf{Adv}_P^{\mathsf{AKA}}(t, q_E, q_S)$ the maximum advantage of any adversary attacking protocol P, running in time t and making q_E calls to the Execute oracle and q_S calls to the Send oracle. Next we define the security against dictionary attack for a password-based protocol as follows [20]:

Definition 1. *A password based group key agreement protocol P is secure against dictionary attack if for every dictionary $\mathcal{D}$,*

$$\mathsf{Adv}_P^{\mathsf{AKA}}(t, q_E, q_S) \leq \frac{q_S}{|\mathcal{D}|} + \epsilon(l)$$

where ϵ is a negligible function in the security parameter l of the system.

This means that the advantage of an adversary essentially grows with the ratio of number of interactions to number of passwords. A polynomially-many calls to the Execute and the Reveal oracles are of no help to an adversary. So the best that an adversary can do is just trying its luck by guessing the password in active on-line (impersonation) attacks.

Remark 2. *Session identity is required to identify a session uniquely and all participants executing a session should hold the same session identity. Conventionally, session idetity sid_U^i for an instance Π_U^i is set to be the concatenation of all (broadcasted) messages sent and received by Π_U^i during its course of execution. This essentially assumes that all the partners of Π_U^i hold the same concatenation value of sent and received messages which may not be the case in general. Our definition of session identity is different and can be applied for more general protocols.*

Remark 3. *We will make the assumption that in each session at most one instance of each user participates. Further, an instance of a particular user participates in exactly one session. This is not a very restrictive assumption, since a user can spawn an instance for each session it participates in. On the other hand, there is an important consequence of this assumption. Suppose there are several sessions which are being concurrently executed. Let the session ID's be $\mathsf{sid}_1, \ldots, \mathsf{sid}_k$. Then for any instance Π_U^i, there is exactly one j such that $(U, i) \in \mathsf{sid}_j$ and for any $j_1 \neq j_2$, we have $\mathsf{sid}_{j_1} \cap \mathsf{sid}_{j_2} = \emptyset$. Thus at any particular point of time, if we consider the collection of all instances of all users, then the relation of being in the same session is an equivalence relation whose equivalence classes are the session IDs. Moreover, an instance Π_U^i not only knows U, but also the instance number i – this being achieved by maintaining a counter.*

Remark 4. *In a password-based group key agreement protocol, a fixed set of participants share a low quality password among themselves drawn uniformly from a relatively small dictionary. When a new user wants to join the group, he has to know the common password shared by the group. Similarly, when a member wants to leave,*

the rest of the participants should choose a new password. Handling dynamic operations (cf. join/leave) in password based setting are thus messy and complicated. A password setup algorithm is required to be executed by the set of participants each time a membership change occurs. This can essentially make the protocol vulnerable to dictionary attacks since the size of the dictionary is small.

3 Protocol

Our protocol is based on the group key agreement protocol of Kim, Lee, Lee [31]. We modify their protocol to password-based group key agreement and provide a concrete security proof in both the random oracle model and the ideal cipher model under CDH assumption. We adopt the formal security model of Bellare et al. [6] (as standardized by Bresson et al. [13]) to prove the security against dictionary attacks.

Suppose a set of n users $\mathcal{P} = \{U_1, U_2, \ldots, U_n\}$ share a low entropy secret password pw drawn uniformly from a small dictionary of size N and wish to establish a high entropy common session key among themselves. We identify a user U_i with it's instance $\Pi_{U_i}^{d_i}$ (for some integer d_i) during a protocol execution and U_{n+i} is taken to be U_i.

Let $G = \langle g \rangle$ be a multiplicative group of some large prime order q and $\overline{G} = G \setminus \{1\}$. Then $\overline{G} = \{g^x | x \in Z_q^*\}$. We take a cryptographically secure hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$ where l is a security parameter, $l \leq |q|$ ($|q|$ is the bit length of q). We also consider three block ciphers $(\mathcal{E}_k, \mathcal{D}_k)$, $(\mathcal{E}'_k, \mathcal{D}'_k)$ and $(\mathcal{E}''_k, \mathcal{D}''_k)$ where k is a password uniformly drawn from a small dictionary of size N. Here $\mathcal{E}_k$, $\mathcal{E}'_k$ and $\mathcal{E}''_k$ are keyed permutations over $\overline{G}$, $\{0,1\}^{2l}$ and $\{0,1\}^l$ respectively whilst $\mathcal{D}_k$, $\mathcal{D}'_k$ and $\mathcal{D}''_k$ are the respective inverses of $\mathcal{E}_k$, $\mathcal{E}'_k$ and $\mathcal{E}''_k$. We denote the concatenation of $A, B \in \{0,1\}^l$ by $A|B$. Figure 1 illustrates our password-based protocol for $n = 5$. The formal description of the protocol is given below.

Procedure PwdKeyAgree $(U[1, \ldots, n])$
(Round 1):
$U_0 = U_n, U_{n+1} = U_1$;
1. **for** $i = 1$ to n **do in parallel**
2. $U_i (= \Pi_{U_i}^{d_i})$ chooses $x_i \in_R Z_q^*$ and nonce $k_i \in_R \{0,1\}^l$;
3. U_i computes $X_i = g^{x_i}$ and $Y_i = \mathcal{E}_{\mathsf{pw}}(X_i)$;
3. U_i sends Y_i to U_{i-1} and U_{i+1};
4. **end for**
(Round 2):
$Y_0 = Y_n, Y_{n+1} = Y_1$;
5. **for** $i = 1$ to $n-1$ **do in parallel**
6. U_i on receiving Y_{i-1} from U_{i-1} and Y_{i+1} from U_{i+1} computes $X_{i-1} = \mathcal{D}_{\mathsf{pw}}(Y_{i-1})$, $X_{i+1} = \mathcal{D}_{\mathsf{pw}}(Y_{i+1})$;
7. U_i computes $K_i^L = \mathcal{H}(X_{i-1}^{x_i})$, $K_i^R = \mathcal{H}(X_{i+1}^{x_i})$, $\overline{X}_i = K_i^R \oplus K_i^L$, $\overline{Y}_i = \mathcal{E}'_{\mathsf{pw}}(k_i | \overline{X}_i)$;
8. U_i sends $\overline{Y}_i$ to the rest of the users;
9. **end for**
10. U_n on receiving Y_{n-1} from U_{n-1} and Y_{n+1} from U_{n+1} computes $X_{n-1} = \mathcal{D}_{\mathsf{pw}}(Y_{n-1})$, $X_{n+1} = \mathcal{D}_{\mathsf{pw}}(Y_{n+1})$;
11. U_n computes $K_n^L = \mathcal{H}(X_{n-1}^{x_n})$, $K_n^R = \mathcal{H}(X_{n+1}^{x_n})$, $\overline{X}_n = k_n \oplus K_n^R$, $\overline{Y}_n = \mathcal{E}''_{\mathsf{pw}}(\overline{X}_n)$;
12. U_n sends $\overline{Y}_n$ to the rest of the users;
Note that $K_i^R = K_{i+1}^L$ for $1 \leq i \leq n-1$ and $K_n^R = K_1^L$;
(Key Computation):
13. **for** $i = 1$ to n **do in parallel**
14. **for** $j = 1$ to $n-1$, $j \neq i$ **do**
15. U_i computes $\mathcal{D}'_{\mathsf{pw}}(\overline{Y}_j)$ and extracts $\overline{X}_j$, k_j;
16. **end for**
17. U_i computes $\overline{X}_n = \mathcal{D}''_{\mathsf{pw}}(\overline{Y}_n)$;
18. **end for**
19. **for** $i = 1$ to n **do in parallel**
20. **for** $j = 1$ to $i-1$ **do**
21. U_i computes $K_{i-j}^L = K_{i-j+1}^L \oplus \overline{X}_{i-j}$;
Note that $K_{i-j}^L = K_{i-j-1}^R$ and $K_1^L = K_n^R$.
22. **end for**
Thus U_i has recovered K_n^R.
23. **end for**
24. **for** $i = 1$ to n **do in parallel**
25. U_i computes $k_n = \overline{X}_n \oplus K_n^R$ and the session key $\mathsf{sk}_{U_i}^{d_i} = \mathcal{H}(k_1 | k_2 | \ldots | k_n)$;
26. **end for**
end PwdKeyAgree

We consider the users $U_1, \ldots, U_n$ participating in the protocol, are on a ring and U_{i-1}, U_{i+1} are respectively the left and right neighbors of U_i for $1 \leq i \leq n$ ($U_0 = U_n, U_{n+1} = U_1$). In the first round, each user U_i chooses randomly a private key $x_i \in_R Z_q^*$ and a nonce $k_i \in_R \{0,1\}^l$, computes $X_i = g^{x_i}$, encrypts it using the password and sends the encrypted value to his neighbors U_{i-1}, U_{i+1}. In the second round, each user U_i recovers X_{i-1}, X_{i+1} by decrypting the encrypted values that he receives from his neighbors, computes his left key $K_i^L = \mathcal{H}(X_{i-1}^{x_i})$ and right key $K_i^R = \mathcal{H}(X_{i+1}^{x_i})$. User U_i, for $1 \leq i \leq n-1$ computes $\overline{X}_i = K_i^R \oplus K_i^L$ and sends $\overline{Y}_i = \mathcal{E}'_{\mathsf{pw}}(k_i | \overline{X}_i)$ to the rest of the users in the second round. In contrast, user U_n computes $\overline{X}_n = k_n \oplus K_n^R$ and sends $\overline{Y}_n = \mathcal{E}''_{\mathsf{pw}}(\overline{X}_n)$ in this round. We note that right key of U_i is same as the left key of U_{i+1}. Finally each user U_i on receiving the encrypted messages $\overline{Y}_j$ from all the users, decrypts those and extracts $\overline{X}_j$, for $1 \leq j \leq n$ and $n-1$ nonces k_j, $1 \leq j \leq n-1$. User U_i then recovers the nonce k_n by computing K_n^R as follows making use of his own left key K_i^L and right key K_i^R.

U_i computes $K_{i-j}^L = K_{i-j+1}^L \oplus \overline{X}_{i-j}$ for $1 \leq j \leq i-1$. Note that $K_{i-j}^L = K_{i-j-1}^R$ and $K_1^L = K_n^R$. Thus U_i recovers the right key K_n^R of U_n. Then U_i computes the nonce $k_n = \overline{X}_n \oplus K_n^R$ and computes the session key $\mathsf{sk}_{U_i}^{d_i} = \mathcal{H}(k_1 | k_2 | \ldots | k_n)$.

Remark 5. *The procedure* PwdKeyAgree *is obtained by modifying the unauthenticated protocol of Kim* et al. *[31] by introducing encryption-based authentication mechanism. The direct replacement of the signature scheme used in [31] by a symmetric encryption scheme using the password as secret key does not yield a secure password-based protocol and one can mount an off-line dictionary*

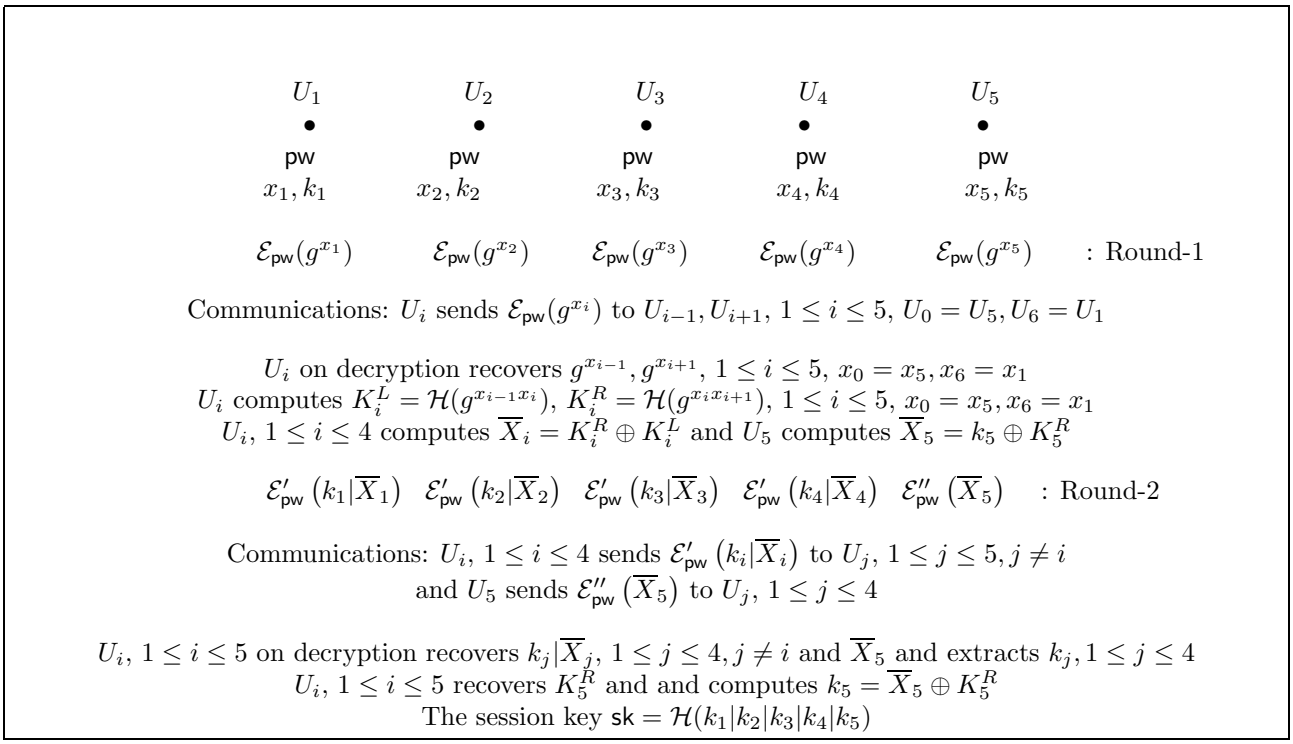

Figure 1: Password-based group key agreement among $n = 5$ users

attack as follows: Observe that in the unauthenticated version of the protocol of Kim et al. [31] with n users, each user U_i for $1 \le i \le n-1$ sends $k_i|T_i$ whilst user U_n sends $k_n \oplus K_n^R|T_n$ in the second round, where $T_i = K_i^L \oplus K_i^R$ for $1 \le i \le n$. We thus obtain the relation $T_1 \oplus T_2 \oplus \ldots \oplus T_n = 0$. Now when we introduce encryption-based authentication mechanism using the password as the secret key, the ciphertexts in the second round communication are simply the encryption of $k_i|T_i$ for $1 \le i \le n-1$ and the encryption of $k_n \oplus K_n^R|T_n$. Thus the plaintexts are co-related instead of being random. This redundancy enables an adversary to make the protocol vulnerable to dictionary attacks by guessing the password off-line and verifying whether the decrypted values ($k_1|T_i$ for $1 \le i \le n-1$, $k_n \oplus K_n^R|T_n$) in the second round communication leads to $T_1 \oplus T_2 \oplus \ldots \oplus T_n = 0$. If so, the adversary's guess for password is correct. To prevent such attacks, we remove the redundancy by restricting U_n to send the encryption of only $k_n \oplus K_n^R$ instead of $k_n \oplus K_n^R|T_n$ in this round. As a result, the key computation is appropriately modified.

Remark 6. *Generally, while executing this protocol, we identify a user U_i with its instance $\Pi_{U_i}^{d_i}$, where d_i is the instance number of the user in the session being executed. At the start of the protocol, the session identity $\mathsf{sid}_{U_i}^{d_i}$ may not be known. This set may be build up (by each participant) as the protocol proceeds. Moreover, when an instance Π_U^i aborts the protocol, it sets $\mathsf{acc}_U^i = 0$ and $\mathsf{sk}_U^i = \mathsf{NULL}$. The procedure* PwdKeyAgree *may be appropriately modified to include these. The details are omitted. Note that the algorithms are correct provided the users are honest, i.e. they do not deviate from the protocol (we additionally assume that the adversary never participates as a user). Then after the execution of the protocol, the group of users agree upon a common session key.*

4 Security Analysis

We will show that our password based authenticated group key agreement protocol is secure in the model as described in Subsection 2.2. Our protocol is based on the CDH assumption and security is achieved in both the random oracle model and the ideal cipher model. However, this proof does not deal with forward secrecy. We leave it for full version.

Theorem 1. *The password based encrypted key agreement protocol P described above satisfies the following:*

$$\begin{aligned} \mathsf{Adv}_P^{\mathsf{AKA}}(t, q_{\mathcal{E}}, q_H, q_E, q_S) &\le \frac{q_{\mathcal{E}}^2}{\min\{(q-1), 2^l\}} + \frac{2q_S}{N} + B \\ B &= 4q_H q_S^2 \mathsf{Succ}_G^{\mathsf{CDH}}(t) \end{aligned}$$

where t is the time bound of the protocol execution P, N is the size of the dictionary of all possible passwords and $q_{\mathcal{E}}, q_H, q_E, q_S$ are respectively the maximum number of encryption/decryption, hash, Execute *and* Send *queries an adversary may make.*

Proof. Suppose $\mathcal{A}$ is an adversary attacking the password based encrypted protocol P. We incrementally define a sequence of games starting from the real game G_0 and ending up at G_4. We define the following two events for game G_i, $0 \le i \le 4$:

– S_i := the event that $b = b'$ where b is the hidden bit involved in the Test query and b' is it's guess output by the adversary $\mathcal{A}$ in game G_i.

– $\mathsf{Encrypt}_i$:= the event that $\mathcal{A}$ makes a Send query on a message that is encrypted by $\mathcal{A}$ itself by guessing the password (*i.e.* $\mathcal{A}$ gets the encryption capability by guessing the password) in game G_i.

The probability of the event $\mathsf{Encrypt}_i$ measures the security against dictionary attack.

Game $\mathbf{G_0}$: This is the real attack and is initialized by drawing a password pw from the set of all possible passwords. The adversary $\mathcal{A}$ is given all the instances of users in order to cover concurrent execution of the protocol P. The adversary also has access of several oracles: hash oracle, encryption/decryption oracles, Execute, Send, Reveal and Test oracles and can submit a polynomial number of queries to these oracles. Finally, $\mathcal{A}$ outputs its guess b' for the bit b involved in the Test query. In this case, $\mathcal{A}$'s advantage is equal to the advantage in the real protocol P. Hence by definition,

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathcal{A},\mathcal{P}} = 2\ \mathsf{Prob}[S_0] - 1.$$

Game $\mathbf{G_1}$: In this game, the Send, Execute, Reveal and Test queries are simulated as in the real attack. We simulate the hash oracle and the encryption/decryption oracles as in the game G_0 by maintaining a hash list Hlist of size q_H, an encryption list Elist of size $q_{\mathcal{E}}$ and a decryption list Dlist as follows (the lists are initially empty).

– *Hash query:* For a hash query $\mathcal{H}(m)$ such that a record $(*, m, r)$ appears in Hlist, the answer is r. Otherwise, choose a random $r \in \{0,1\}^l$ and set $\mathcal{H}(m) = r$. The record (δ, m, r) is added to Hlist where $\delta \in \{0,1\}$ is a bit indicating the originator of the query: $\delta = 0$ if the query comes from the simulator and $\delta = 1$ if the query comes from the adversary.

– *Encryption query:* For an encryption query $E_k(X)$ (E is either $\mathcal{E}$ or $\mathcal{E}'$ or $\mathcal{E}''$) such that a record $(\delta, k, X, \Lambda, Y)$ appears in Elist (δ is either 0 or 1 and Λ is either E or D), the answer is Y. Otherwise, a random ciphertext Y of length $|X|$ is chosen and the record (δ, k, X, E, Y) is added to Elist where $\delta \in \{0,1\}$. If the query comes from the simulator, δ is set to be 0; else the query is directly asked by the adversary and δ is set to be 1.

– *Decryption query:* For a decryption query $D_k(Y)$ (D is either $\mathcal{D}$ or $\mathcal{D}'$ or $\mathcal{D}''$) such that a record $(\delta, k, X, \Lambda, Y)$ appears in Elist (δ is either 0 or 1 and Λ is either E or D), the answer is X. Otherwise, the following rule is applied to obtain the answer X.
Case 1: if $D = \mathcal{D}$, then $X = g^r$ where r is chosen randomly from Z_q^*.
Case 2: if $D = \mathcal{D}'$, then $X = r_1|r_2$ where $r_1, r_2 \in \{0,1\}^l$ are chosen uniformly at random.
Case 3: if $D = \mathcal{D}''$, then $X = r$ where $r \in \{0,1\}^l$ is chosen uniformly at random.
The record (k, Y, D, X) is added to Dlist and the record $(0, k, X, D, Y)$ is added to Elist. Note that a record (k, Y, D, X) appears in Dlist if and only if $D_k(Y)$ is asked first before the corresponding encryption query. For game G_i, the event $\mathsf{Encrypt}_i$ occurs when there exists a record $(1, \mathsf{pw}, X, *, Y)$ in Elist, such that Y has been submitted to a Send query. Hence the corresponding record (pw, Y, D, X) does not appear in Dlist.

From this simulation, we easily see that the games G_1 and G_0 are perfectly indistinguishable, unless the permutation property (bijection) of block cipher does not hold. This means that there exists a single ciphertext (respectively plaintext) for two different plaintexts (respectively ciphertexts). Now since the encryption and decryption lists are at most of size $q_{\mathcal{E}}$, the probability that the permutation property of block cipher does not hold is at most $\frac{q_{\mathcal{E}}^2}{2\min\{(q-1), 2^{2l}, 2^l\}}$. We abort the game G_1 when such a collision occurs (and b' is set to be a random bit). Hence

$$|\mathsf{Prob}[S_1] - \mathsf{Prob}[S_0]| \leq \frac{q_{\mathcal{E}}^2}{2\min\{(q-1), 2^l\}}$$

Game $\mathbf{G_2}$: This game is same as the game G_1 except that we abort when $\mathsf{Encrypt}_1$ occurs in game G_1. In other words, we delete the protocol executions whenever a $\mathsf{Send}(U, i, Y)$ query is asked and a record of the form $(1, \mathsf{pw}, X, *, Y)$ appears in Elist. Then

$$|\mathsf{Prob}[S_2] - \mathsf{Prob}[S_1]| \leq \mathsf{Prob}[\mathsf{Encrypt}_1].$$

Note that in games G_0, G_1, G_2, the following distribution holds for transcript, session key pair:

$$\mathsf{Dist} := \left\{ \begin{array}{l} x_1, \ldots, x_n \longleftarrow Z_q^*; k_1, \ldots, k_n \longleftarrow \{0,1\}^l \\ X_1 = g^{x_1}, X_2 = g^{x_2}, \ldots, X_n = g^{x_n}; \\ Y_1 = \mathcal{E}_{\mathsf{pw}}(X_1), \ldots, Y_n = \mathcal{E}_{\mathsf{pw}}(X_n); \\ K_1^R = K_2^L = \mathcal{H}(g^{x_1 x_2}), K_2^R = K_3^L = \mathcal{H}(g^{x_2 x_3}), \\ \quad \ldots, K_n^R = K_1^L = \mathcal{H}(g^{x_n x_1}); \ : (T, \mathsf{sk}) \\ \overline{X}_1 = k_1 | K_1^L \oplus K_1^R, \ldots, \overline{X}_{n-1} \\ \quad = k_{n-1} | K_{n-1}^L \oplus K_{n-1}^R, \overline{X}_n = k_n \oplus K_n^R; \\ \overline{Y}_1 = \mathcal{E}'_{\mathsf{pw}}(\overline{X}_1), \ldots, \overline{Y}_{n-1} = \mathcal{E}'_{\mathsf{pw}}(\overline{X}_{n-1}), \overline{Y}_n \\ \quad = \mathcal{E}''_{\mathsf{pw}}(\overline{X}_n); \\ T = (Y_1, \ldots, Y_n; \overline{Y}_1, \ldots, \overline{Y}_n); \\ \quad \mathsf{sk} = \mathcal{H}(k_1 | k_2 | \ldots | k_n) \end{array} \right\}$$

Game $\mathbf{G_3}$: This game is exactly same as the game G_2 except that we use an instance ($A = g^a, B = g^b$) of CDH problem with its solution $C = g^{ab}$ and the given values a, b in the simulation according to the following distribution of transcript, session key pair.

$$\mathsf{Dist}' := \left\{ \begin{array}{l} c_1, c_2, x_3, \ldots, x_n \longleftarrow Z_q^*; k_1, \ldots, k_n \longleftarrow \{0,1\}^l \\ X_1 = A^{c_1}, X_2 = B^{c_2}, X_3 = g^{x_3}, \ldots, X_n = g^{x_n}; \\ Y_1 = \mathcal{E}_{\mathsf{pw}}(X_1), \ldots, Y_n = \mathcal{E}_{\mathsf{pw}}(X_n); \\ K_1^R = K_2^L = \mathcal{H}(C^{c_1 c_2}), K_2^R = K_3^L = \mathcal{H}(B^{c_2 x_3}), \\ \quad K_3^R = K_4^L = \mathcal{H}(g^{x_3 x_4}) \ldots, \ : (T, \mathsf{sk}) \\ K_{n-1}^R = K_n^L = \mathcal{H}(g^{x_{n-1} x_n}), \\ \quad K_n^R = K_1^L = \mathcal{H}(A^{x_n c_1}); \\ \overline{X}_1 = k_1 | K_1^L \oplus K_1^R, \ldots, \overline{X}_{n-1} \\ \quad = k_{n-1} | K_{n-1}^L \oplus K_{n-1}^R, \overline{X}_n = k_n \oplus K_n^R; \\ \overline{Y}_1 = \mathcal{E}'_{\mathsf{pw}}(\overline{X}_1), \ldots, \\ \quad \overline{Y}_{n-1} = \mathcal{E}'_{\mathsf{pw}}(\overline{X}_{n-1}), \overline{Y}_n = \mathcal{E}''_{\mathsf{pw}}(\overline{X}_n); \\ T = (Y_1, \ldots, Y_n; \overline{Y}_1, \ldots, \overline{Y}_n); \\ \quad \mathsf{sk} = \mathcal{H}(k_1 | k_2 | \ldots | k_n) \end{array} \right\}$$

The encryption, decryption, hash, Execute, Reveal and Test queries are simulated as in game G_2. Let us now explicitely describe the send queries. All the Send queries are simulated as in the real game execept the send queries of users U_n, U_1, U_2, U_3. Let us denote the initial send query by Send_0 and the send queries of first and second round by Send_1 and Send_2 respectively. Send_0 query is invoked simply to prompt an unused instance to initialize the protocol. Send_0 and Send_2 queries have no output. For any query of the form $\mathsf{Send}_2(U, d, \overline{Y})$ for an instance Π_U^d, one invokes the decryption oracle to compute $\overline{X} = \mathcal{D}'_{\mathsf{pw}}(\overline{Y})$ (or $\mathcal{D}''_{\mathsf{pw}}(\overline{Y})$). The oracle goes into an expecting state. Note that we do not allow the event Encrypt to occur, i.e. adversary is not given the power of encrypting a message by itself guessing the password. So any subsequent send query (i.e. after $\mathsf{Send}_0(*, *, \text{“Start”})$ query) is on a properly encrypted message which is encrypted by the simulator (and not by the adversary) on querying the encryption oracle.

Consider send queries for instance $\Pi_{U_1}^{d_1}$ which are limitted to any of the followings:
$\mathsf{Send}_0(U_1, d_1, \text{“Start”})$, $\mathsf{Send}_1(U_1, d_1, Y_2)$, $\mathsf{Send}_1(U_1, d_1, Y_n)$, and $\mathsf{Send}_2(U_1, d_1, \overline{Y}_i)$ for $1 \leq i \leq n$, $i \neq 1$. The Send_0 and Send_2 queries are simulated as usual. In response to $\mathsf{Send}_1(U_1, d_1, Y_1)$ and $\mathsf{Send}_1(U_1, d_1, Y_n)$ queries, one invokes the decryption and hash oracles as in game G_2 to compute $X_2 = \mathcal{D}_{\mathsf{pw}}(Y_2), K_1^R = K_2^L = \mathcal{H}(C^{c_1 c_2})$ and $X_n = \mathcal{D}_{\mathsf{pw}}(Y_n), K_1^L = K_n^R = \mathcal{H}(A^{c_1 x_n})$ respectively. If one of these two send queries are yet not asked, then the oracle goes into an expecting state; otherwise encryption oracle is invoked to output $\overline{Y}_1 = \mathcal{E}'_{\mathsf{pw}}(k_1 | K_1^L \oplus K_1^R)$.

For instance $\Pi_{U_2}^{d_2}$, $\mathsf{Send}_1(U_2, d_2, Y_1)$ and $\mathsf{Send}_1(U_2, d_2, Y_3)$ are answered as follows: One invokes the decryption oracle $\mathcal{D}_{\mathsf{pw}}$ on Y_1, Y_3 as usual to get X_1, X_3 respectively and hash oracle to obtain $K_2^L = K_1^R = \mathcal{H}C^{c_1 c_2}), K_2^R = K_3^L = \mathcal{H}(B^{c_2 x_3})$.If both these send queries are taken place, then the encryption oracle is invoked to output $\overline{Y}_2 = \mathcal{E}'_{\mathsf{pw}}(k_2 | K_2^L \oplus K_2^R)$; else the oracle goes into an expecting state.

On $\mathsf{Send}_1(U_3, d_3, Y_2)$ and $\mathsf{Send}_1(U_3, d_3, Y_4)$ queries for instance $\Pi_{U_3}^{d_3}$, X_2, X_4 are obtained by querying $\mathcal{D}_{\mathsf{pw}}$ on Y_2, Y_4 respectively. The output is $\overline{Y}_3 = \mathcal{E}'_{\mathsf{pw}}(k_3 | K_3^L \oplus K_3^R)$ where $K_3^L = K_2^R = \mathcal{H}(B^{c_2 x_3})$ and $K_3^R = K_4^L = \mathcal{H}(g^{x_3 x_4})$, provided both the send queries are asked. Otherwise the oracle goes into an expecting state.

Similarly on $\mathsf{Send}_1(U_n, d_n, Y_1)$ and $\mathsf{Send}_1(U_n, d_n, Y_{n-1})$ queries for instance $\Pi_{U_n}^{d_n}$, X_1 and X_{n-1} are recovered by invoking decryption oracle $\mathcal{D}_{\mathsf{pw}}$ on Y_1 and Y_{n-1} respectively and hash oracle is invoked to compute $K_n^L = K_{n-1}^R = \mathcal{H}(g^{x_n x_{n-1}}), K_n^R = K_1^L = \mathcal{H}(A^{c_1 x_n})$. The output is $\overline{Y}_n = \mathcal{E}''_{\mathsf{pw}}(k_n \oplus K_n^R)$, provided both the send queries are asked. Otherwise the oracle goes into an expecting state.

These simulations are still perfect as soon as new random c_1, c_2 are drawn and x_i for $3 \leq i \leq n$ are random. Thus as long as c_1, c_2, x_i for $3 \leq i \leq n$ are random, the simulation is indistinguishable from that in real attack scenario, i.e. the distributions Dist and Dist′ are equivalent. Hence

$$\mathsf{Prob}[S_3] = \mathsf{Prob}[S_2]$$

and also

$$\mathsf{Prob}[\mathsf{Encrypt}_3] = \mathsf{Prob}[\mathsf{Encrypt}_2] = \mathsf{Prob}[\mathsf{Encrypt}_1].$$

Game $\mathbf{G_4}$: In this game, we do exactly as above except that any hash value involving $K_1^R (= K_2^L)$, asked by the users are answered independently from the random oracles. More explicitely, we are given an instance $(A = g^a, B = g^b)$ of CDH problem without its solution $C = g^{ab}$ and without the values a, b. Now, whenever two successive users U_1 and U_2 compute $X_1 = g^{x_1}$, $X_2 = g^{x_2}$ respectively choosing x_1, x_2 randomly from Z_q^*, we simulate as in G_3, i.e. with $X_1 = A^{c_1}$ and $X_2 = B^{c_2}$ where c_1, c_2 are random values in Z_q^*. The exponents x_j for other users U_j are chosen at random from Z_q^*. The only difference of this game with G_3 is that the hash value involving $K_1^R (= K_2^L)$, asked by the users (simulator), is answered with a random value r from $\{0,1\}^l$ instead of quering the hash oracle. Here the adversary $\mathcal{A}$ may ask the same hash query which is still answered by quering the random oracle. Thus we have the folowing distribution of transcript, session key pair.

$$\mathsf{Dist}'' := \left\{ \begin{array}{l} c_1, c_2, x_3, \ldots, x_n \longleftarrow Z_q^*; k_1, \ldots, k_n \longleftarrow \{0,1\}^l \\ X_1 = A^{c_1}, X_2 = B^{c_2}, X_3 = g^{x_3}, \ldots, X_n = g^{x_n}; \\ Y_1 = \mathcal{E}_{\mathsf{pw}}(X_1), \ldots, Y_n = \mathcal{E}_{\mathsf{pw}}(X_n); \\ K_1^R = K_2^L = r \in \{0,1\}^l, K_2^R = K_3^L = \mathcal{H}(B^{c_2 x_3}), \\ \quad K_3^R = K_4^L = \mathcal{H}(g^{x_3 x_4}) \ldots, \ : (T, \mathsf{sk}) \\ K_{n-1}^R = K_n^L = \mathcal{H}(g^{x_{n-1} x_n}), K_n^R = K_1^L \\ \quad = \mathcal{H}(A^{x_n c_1}); \\ \overline{X}_1 = k_1 | K_1^L \oplus K_1^R, \ldots, \overline{X}_{n-1} \\ \quad = k_{n-1} | K_{n-1}^L \oplus K_{n-1}^R, \overline{X}_n = k_n \oplus K_n^R; \\ \overline{Y}_1 = \mathcal{E}'_{\mathsf{pw}}(\overline{X}_1), \ldots, \overline{Y}_{n-1} = \mathcal{E}'_{\mathsf{pw}}(\overline{X}_{n-1}), \\ \quad \overline{Y}_n = \mathcal{E}''_{\mathsf{pw}}(\overline{X}_n); \\ T = (Y_1, \ldots, Y_n; \overline{Y}_1, \ldots, \overline{Y}_n); \\ \quad \mathsf{sk} = \mathcal{H}(k_1 | k_2 | \ldots | k_n) \end{array} \right\}$$

We define an event AskH as follows:

– AskH := the event that the adversary $\mathcal{A}$ has discovered that the broadcasted message involving hash value K_1^R is incorrect by using its hash oracle queries.

The adversary $\mathcal{A}$ can find an inconsistency in the hash value involving K_1^R if $\mathcal{A}$ can submit a correct guess for $C^{c_1c_2}$ to the hash oracle. This is because of the fact that the same hash query asked by the adversary $\mathcal{A}$, is still answered by quering the random hash oracle whereas that asked by the simulator is answered with the random value $r \in \{0,1\}^l$. Thus if AskH occurs, then Hlist will contain a record of the form $(1, C^{c_1c_2}, r_1)$ for some $r_1 \neq r$. We abort the game if AskH occurs and $\mathcal{A}$'s output b' is chosen randomly. Thus, as long as the event AskH does not occur, the distributions Dist′ and Dist″ are equivalent.

Hence

$$|\mathsf{Prob}[S_4] - \mathsf{Prob}[S_3]| \leq \mathsf{Prob}[\mathsf{AskH}]$$

and

$$|\mathsf{Prob}[\mathsf{Encrypt}_4] - \mathsf{Prob}[\mathsf{Encrypt}_3]| \leq \mathsf{Prob}[\mathsf{AskH}].$$

Now given $(A = g^a, B = g^b)$, one can output a correct Diffie-Hellman value $C = g^{ab}$ if both the following two events occur: (1) two successive users U_1, U_2 compute $X_1 = A^{c_1}$ and $X_2 = B^{c_2}$ and use random value as a hash value involving $K_1^R (= K_2^L)$ instead of querying the hash oracle and (2) the adversary $\mathcal{A}$ makes a hash query (among at most q_H) on a correctly guessed value $C^{c_1c_2}$. By guessing $C^{c_1c_2}$ that has been asked by the adversary, and the corresponding hash query, simulator extracts $C = (C^{c_1c_2})^{(c_1c_2)^{-1}}$. Hence

$$\mathsf{Succ}^{\mathsf{CDH}}_{G,\mathcal{A}}(t) \geq \frac{1}{q_H q_S^2}\mathsf{Prob}[\mathsf{AskH}].$$

In this game, $\mathcal{A}$'s output bit b' is totally random. So

$$\mathsf{Prob}[S_4] = \frac{1}{2}.$$

Combining all the probability bounds obtained from all the games, we get

$$\begin{aligned} |\mathsf{Prob}[S_0] - \frac{1}{2}| &\leq \frac{q_\mathcal{E}^2}{2\min\{(q-1), 2^l\}} + \mathsf{Prob}[\mathsf{Encrypt}_4] + B \\ B &= 2q_H q_S^2 \mathsf{Succ}^{\mathsf{CDH}}_{G,\mathcal{A}}(t). \end{aligned}$$

Now to find $\mathsf{Prob}[\mathsf{Encrypt}_4]$, we argue as follows. The event $\mathsf{Encrypt}_4$ occurs whenever a $\mathsf{Send}(U, i, Y)$ query is asked by the adversary $\mathcal{A}$ where a record of the form $(1, \mathsf{pw}, X, *, Y)$ appears in Elist. We note that $\mathcal{A}$ may ask at most q_S Send queries and so is able to build at most q_S data (encryptions) by itself. Hence at most q_S passwords it might have tried in the on-line guessing. In other words, at most q_S impersonation attempts can be done by $\mathcal{A}$ by guessing the password. Moreover, from the point of view of the adversary $\mathcal{A}$, the plaintexts to be encrypted are completely indistinguishable from random plaintexts drawn from the domain of corresponding encryption functions as long as the exponents from Z_q^* and the nonces from $\{0,1\}^l$ chosen by the simulator in game G_3 or G_4 are random. Hence an off-line exhaustive search on passwords will not get any bias on the actual password pw and a polynomially-many calls to the Execute and the Reveal oracles will be of no help to $\mathcal{A}$. Thus the simulation is completely independent from the password pw from information theoretic sense. This implies that the probability that a message, submitted to a Send query, is encrypted by the adversary itself guessing the password is at most $\frac{q_S}{N}$. So we have

$$\mathsf{Prob}[\mathsf{Encrypt}_4] \leq \frac{q_S}{N}$$

where q_S is the maximum number of Send queries the adversary may ask, i.e. maximum number of impersonation attempts that adversary can make. This yields the statement of the theorem. □

5 Mutual Authentication

Mutual authentication is desirable to confirm each other's knowledge of the agreed common key for each pair of parties in the group before preceeding to use the common key as a session key. Mutual authentication thus enables to convince to each of two parties that the other knows the password. We note that our protocol as presented in Section 3 achieves key agreement only, not mutual authentication. However, one can trivially add mutual authentication by using the shared agreed key to construct a simple "authenticator" for the other parties. There are various well-known classical approaches for constructing such "authenticators" [12, 15] that converts any key agreement protocol into a protocol that provides mutual authentication. For instance, we can adopt the following methodology for verification of the common session key sk.

Each user U_i chooses a random challange C_i; computes $\sigma = H_1(\mathsf{sk})$ where H_1 is a cryptographically secure one way hash function; a tag $\tau_i = \mathsf{Mac}_\sigma(\mathsf{ID}|C_i)$ where $\mathsf{ID} = U_1|U_2|\cdots|U_n$ and Mac is a secure Message Authentication Code generation function; and sends $C_i|\tau_i$ to the rest of the users. Each user U_j, on receiving $C_i|\tau_i$, verifies τ_i on C_i for all i, $1 \leq i \leq n, i \neq j$ (making use of his own common secret sk). If verification succeeds, then U_j computes the session key $\mathsf{sk}_0 = H_0(\mathsf{sk})$ where H_0 is another cryptographically secure one way hash function; otherwise U_j aborts the protocol. These modifications add one more round to our protocol. The details of the security is omitted.

6 Efficiency

Efficiency of a protocol is related to the costs of communication and computation. Communication cost involves counting total number of rounds and total messages transmitted through the network during a protocol execution. Number of rounds is a critical concern in practical environments where number of group members is large. Table

Table 1: Protocol comparison

Protocol	Communication			Computation					Hardness Assumption	Security Model
	R	BL	PTP	Exp	H	XOR	Enc	Dec		
BCP [13]	n	$n\|e\|$	$2n-2$	$2n$	-	-	1	2	TG-CDH,M-DDH	i.c.m,r.o.m
Our protocol	2	$2\|e\|$	$n+1$	3	4	$n-1$	2	$n+1$	CDH	i.c.m, r.o.m

Notes:

n: total number of users in a group
R: total number of rounds
BL: maximum bit length of messages sent per user
PTP: maximum number of point-to-point communication per user
Exp: maximum number of modular exponentiations computed per user
H: maximum number of hash function evaluation per user
XOR: maximum number of XOR operations computed per user
Enc: maximum number of symmetric key encryptions per user
Dec: maximum number of symmetric key decryptions per user
$|e|$: maximum size of an encrypted plaintext

1 compares our protocol and Bresson et al.'s password-based group key agreement protocol (BCP) [13] where the following notations are used (TG-CDH stands for Trigon Group Computational Diffie-Hellman Problem, M-DDH stands for Multi Decision Diffie Hellman Problem, i.c.m denotes ideal cipher model and r.o.m stands for random oracle model).

Our protocol requires only 2 rounds which makes our protocol efficient from communication point of view. Each user sends one message in each round. The maximum bits that a member sends during the execution of the protocol is $2|e|$ where $|e|$ is the maximum size of an encrypted plaintext (either by applying $\mathcal{E}$ or $\mathcal{E}'$ or $\mathcal{E}''$). As mentioned earlier, we consider the users $U_1, \ldots, U_n$ participating in the protocol are on a ring and U_{i-1}, U_{i+1} are respectively the left and right neighbors of U_i for $1 \leq i \leq n$ ($U_0 = U_n, U_{n+1} = U_1$). User U_i, $1 \leq i \leq n-1$, sends a message in round 1 only to the users U_{i-1}, U_{i+1} and a message in round 2 to the rest of the $n-1$ users whilst the last user U_n sends one message in each round to all the $n-1$ users. Each group member performs at most 3 modular exponentiations, 4 one-way hash function evaluations, $n-1$ XOR operations, 2 encryptions and $n+1$ decryptions. The operations dependent on the number of group members are the XOR operation and symmetric key decryption operation. The total cost of computation is highly reduced due to the use of XOR operation in our protocol as compared to other multi-party password-based authenticated key agreement protocol [13]. We use symmetric key encryption and decryption, cost of which are low compared to signature generation and verification. Bandwidth of messages communicated are also shorter as signatures are not appended to them. Hence our protocol achieves efficiency in both communication and computation aspects. Our constant round protocol can be applied for a large group of participants as compared to multi-party password-based protocol of Bresson et al. [13] (which becomes impractical if $n > 100$).

7 Conclusion

We have presented an efficient and secure password-based authenticated group key agreement protocol. The security is achieved in the formal security model of Bellare et al. [6]. We use the proof technique of Bresson et al. [13]. The protocol is proven to be secure under CDH assumption in both the random oracle model and the ideal cipher model. To obtain secure password-based efficient group key agreement protocol under standard assumption without using random oracle is an interesting research topic and this area requires to be studied for further improvement.

References

[1] M. Abdalla, M. Bellare and P. Rogaway, "DHIES: An encryption scheme based on the Diffie-Hellman problem," in *CT-RSA 2001*, pp. 143-158, 2001.

[2] M. Abdalla, P. A. Fouque and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *PKC 2005*, LNCS 3386, pp. 65-84, Springer-Verlag, 2005.

[3] N. Asokan and P. Ginzboorg, "Key agreement in ad-hoc networks," *Computer Communications*, vol. 23, no. 18, pp. 1627–1637, 2000.

[4] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Crypto'93*, LNCS 773, pp. 231-249, Springer-Verlag, 1994.

[5] M. Bellare and P. Rogaway, "Provably secure session key distribution: the three party case, in *STOC'95*, pp. 57-66, ACM Press, 1995.

[6] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *B. Preneel, editor, Eurocrypt 2000*, LNCS 1807, pp. 139–155, Springer-Verlag, 2000.

[7] S. M. Bellovin and M. Merritt, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file

compromise," in *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pp. 244–250, 1993.

[8] Bluetooth, *Specification of Bluetooth System*, Dec. 1999, available at http://www.bluetooth.com/ developer/specification /specification.asp.

[9] M. Boyarsky, "Public-key cryptography and password protocols: The multi-user case," in *ACM Security (CCS'99)*, pp. 63–72, 1999.

[10] V. Boyko, P. MacKenzie and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman," in *Eurocrypt 2000*, LNCS 1807, pp. 156–171, Springer-Verlag, May 2000.

[11] E. Bresson, O. Chevassut and D. Pointcheval, "New security results on encrypted key exchange," in *PKC 2004*, LNCS 2947, pp. 145–158, Springer-Verlag, Mar. 2004.

[12] E. Bresson, O. Chevassut and D. Pointcheval, "Proof of security for password-based key exchange (IEEE P1363 AuthA protocol and extensions)," in *ACM-CCS'03*, pp. 241-250, 2003.

[13] E. Bresson, O. Chevassut and D. Pointcheval, "Group Diffie-Hellman key exchange secure against dictionary attack, in *Asiacrypt'02*, LNCS 2501, pp. 497–514, Springer-Verlag, 2002.

[14] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably authenticated group Diffie-Hellman key exchange - the dynamic case," in *Asiacrypt 2001*, LNCS 2248, pp. 290–309, Springer-Verlag, 2001.

[15] E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," in *Proceedings of 8th Annual ACM Conference on Computer and Communications Security*, pp. 255–264, 2001.

[16] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *EUROCRYPT'94*, LNCS 950, pp. 275–286, Springer-Verlag, 1995.

[17] J. W. Byun, I. R. Jeong, D. H. Lee and C. S. Park, "Password-authenticated key exchange between clients with different passwords," in *ICICS'02*, LNCS 2513, pp. 134–146, Springer-Verlag, Dec. 2002.

[18] R. Dutta and R. Barua, "Constant round dynamic group key agreement," in *ISC 2005*, LNCS, Springer-Verlag, to appear on Sept. 2005, Singapore.

[19] Y. Ding and P. Horster, "Undetectable on-line password guessing attacks," *ACM SIGOPS Operating Systems Review*, vol. 29, no. 4, pp. 77–86, Oct. 1995.

[20] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange," in *Eurocrypt 2003*, LNCS 2656, pp. 524–543, Springer-Verlag, May 2003.

[21] O. Goldreich and Y. Lindell, "Session-key generation using human memorable passwords only," in *Crypto 2001*, LNCS 2139, pp. 408–432, Springer-Verlag, Aug. 2001.

[22] L. Gong, "Optimal authentication protocols resistant to password guessing attacks," in *CSFW'95*, IEEE Computer Society, pp. 24–29, Kenmare, County Kerry, Ireland, Mar. 1995.

[23] L. Gong, T. M. A. Lomas, R. M. Needham and J. H. Saltzer, "Protecting poorly chosen secrets from guessing attacks," *IEEE Journal of Selected on Communications*, vol. 11, no. 5, pp. 648–656, June 1993.

[24] S. Halevi and H. Krawczyk, "Public key cryptography and password protocols," *ACM Transactions on Information and System Security*, pp. 524-543, 1999.

[25] D. P. Jablon, "Strong password-only authenticated key exchange," *SIGCOMM Computer Communication Review*, vol. 26, no. 5, pp. 5–26, 1996.

[26] M. Jakobsson and S. Wetzel, "Security weaknesses in bluetooth," in *Proceedings of the RSA Cryptographer's Track (RSA CT'01)*, RSA Data Security, LNCS 2020, pp. 176–191, Springer-Verlag, 2001.

[27] S. Jiang and G. Gong, "Password-based Key exchange With mutual authentication," in *SAC 2004*, LNCS 3006, pp. 291-306, Springer-Verlag, 2004.

[28] J. Katz, R. Ostrovsky and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords," in *Eurocrypt 2001*, LNCS 2045, pp. 475–494, Springer-Verlag, May 2001.

[29] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," in *CRYPTO 2003*, LNCS 2729, pp. 110-125, Springer-Verlag, 2003.

[30] C. Kaufman, R. Perlman and M. Speciner, *Network security*, Prentice Hall, 1997.

[31] H. J. Kim, S. M. Lee and D. H. Lee, "Constant-round authenticated group key exchange for dynamic groups," in *Asiacrypt'04*, LNCS 3329, pp. 245-259, Sringer-Verlag, 2004.

[32] J. Kim, S. Kim, J. Kwak and D. Won, "Cryptanalysis and improvement of password authenticated key exchange scheme between clients with different passwords," in *ICCSA'04*, LNCS 3043, pp. 895–902, Springer-Verlag, May 2004.

[33] H. Krawczyk, "SIGMA: The "SIGn-and-MAc" approach to authenticate Diffie-Hellman and its use in the ike protocols," in *Crypto'03*, LNCS 2729, pp. 400–425, Springer-Verlag, Aug. 2003.

[34] S. M. Lee, J. Y. Hwang and D. H.L'ee, "Efficient password-based group key exchange," in *proceedings of TrustBus 2004*, LNCS 3184, pp. 191–199, Springer-Verlag, 2004.

[35] C. L. Lin, H. M. Sun and T. Hwang, "Three-party encrypted key exchange: attacks and solution," *ACM SIGOPS Operating Systems Review*, vol. 34, no. 4, pp. 12–20, Oct. 2000.

[36] C. L. Lin, H. M. Sun, M. Steiner and T. Hwang, "Three-party encrypted key exchange without server public keys," *IEE Communications Letters*, vol. 5, no. 12, pp. 497–499, Dec. 2001.

[37] S. Lucks, "Open key exchange: How to defeat dictionary attacks without encrypting public keys," in *Proceedings of the Workshop of Security Protocols*, LNCS 1361, pp. 79-90, Springer-Verlag, 1997.

[38] P. MacKenzie, S. Patel and R. Swaminathan, "Password-authenticated key exchange based on RSA," in *Asiacrypt 2000*, LNCS 1976, pp. 599–613, Springer-Verlag, Dec. 2000.

[39] P. D. Mackenzie, *The PAK suit: Protocols for password-authenticated key exchange*, Contributions to IEEE P1363.2, 2002.

[40] P. D. Mackenzie, T. Shrimpton and M. Jakobsson, "Threshold password-authenticated key exchange," in *Crypto 2002*, LNCS 2442, pp. 385–400, Springer-Verlag, Aug. 2002.

[41] P. MacKenzie and R. Swaminathan, *Secure Network Authentication with Password Identification*, Submission to IEEE P1363a, Aug. 1999, Available from http://grouper.ieee.org/groups/1363/.

[42] A. Menezes, P. V. Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRG Press, 1997.

[43] NIST, *AES*, Dec. 2000, Available at http://www.nist.gov/aes.

[44] K. Obraczka, G. Tsudik and K. Viswanath, "Pushing the limits of multicast in ad hoc networks," in *Intenational Conference on Distributed Computing System*, pp. 719-722, Apr. 2001.

[45] S. Patel, "Number theoritic attacks on secure password schemes," in *Proceedings of the 1997 IEEE Sympossium on Security and Privacy*, pp. 236–247, 1997.

[46] C. E. Perkins, *Ad hoc networking* Addition Wesley, 2001.

[47] M. D. Raimondo and R. Gennaro, "Provably secure threshold password-authenticated key exchange," in *Eurocrypt 2003*, LNCS 2656, pp. 507–523, Springer-Verlag, May 2003.

[48] V. Shoup, *On formal models for secure key exchange*, Technical Report RZ 3120, IBM, 1999.

[49] J. G. Steiner, B. C. Neuman and J. L. Schiller, "Kerberos: An authentication service for open networks," in *Proceedings of the USENIX Winter Conference*, pp. 191–202, Dallas, TX, USA, 1998.

[50] M. Steiner, G. Tsudik and M. Waidner, "Refinement and extension of encrypted key exchange," *ACM SIGOPS Operating Systems Review*, vol. 29, no. 3, pp. 22–30, July 1995.

[51] G. Tsudik and E. V. Herreweghen, "Some remarks on protecting weak keys and poorly-chosen secrets from guessing attacks," in *SRDS'93: The 12th Symposium on Reliable Distributed Systems*, IEE Computer Society, Princeton, New Jersey, USA, pp. 136–142, Oct. 1993

[52] S. Wang, J. Wang and M. Xu, "Weakness of a password-authenticated key exchange protocol between clients with different passwords," in *ACNS'04*, LNCS 3089, pp. 414–425, Springer-Verlag, June 2004.

[53] T. Wu, "The secure remote password protocol," in *1998 Internet Society Symposium on Network and Distributed System Security*, pp. 97–111, 1998.

[54] H. T. Yeh, H. M. Sun and T. Hwang, "Efficient three-party authentication and key agreement protocols resistant to password guessing attacks," *Journals of Information Science and Engineering*, vol. 19, no. 6, pp. 1059–1070, Nov. 2003.

[55] M. Zhang, "Password authenticated key exchange using quadretic residues," in *Proceedings of ACNS 2004*, LNCS 3089, pp. 233–247, Springer-Verlag, June 2004.

[56] L. Zhou and Z. J. Hass, "Securing ad hoc networks," *IEEE Network Magazine*, vol. 13, no. 6, pp. 24-30, 1999.

Ratna Dutta was born in Calcutta, India. She has completed her B.Sc (Honours in Mathematics) in 1996 and M.Sc. (in Applied Mathematics) in 1998 from University of Calcutta, India. She has joined in 2000 for her Ph.D in Discrete Mathematics and Theoretical Computer Science in Stat-Math Unit of Indian Statistical Institute, Calcutta, India and submitted her thesis on Cryptology in August, 2005. Currently she is a Senior Research Fellow at Indian Statistical Institute.

Rana Barua received his B.Sc and M.Sc in Mathematics from University of Calcutta, India, in 1973 and 1975 respectively. He received his Ph.D degree in Descriptive Set Theory from Indian Statistical Institute in 1987. He is currently a professor at Indian Statistical Institute. He has worked in Descriptive Set Theory and Logic, Automata Theory, Simple Voting Games and his current research interests include Cryptography.