Journal of Information Hiding and Multimedia Signal Processing ©2014 ISSN 2073-4212
Ubiquitous International Volume **5**, Number **4**, October **2014**

Structured and Efficient Password-Based Group Key Agreement Protocol

Hongfeng Zhu

Software College
Shenyang Normal University
No.253, HuangHe Bei Street, HuangGu District, Shenyang, P.C 110034 - China
zhuhongfeng1978@163.com

Received November, 2013; revised April, 2014

Abstract. *Password-based group key agreement protocol (PGKA) is a kind of user experience-centered scheme which can provide many advantages, such as, memorable secret, no hardware involved, strong security and high efficiency etc. To accomplish above mentioned PGKA scheme, we should put emphasis on how to design a mechanism to apply cryptographic secret instead of password to capture the secure attributes, and at the same time make the calculated consumption and communicating round restrict to an acceptable bound. In this paper, we design a new structured and efficient password-based group key agreement protocol, which has a general structure to realize N-party PGKA scheme with members revocation or join easily. Then, we give a concrete PGKA protocol and an example for N=4 case. Furthermore, we prove the security of the new PGKA protocol which based on the DDH assumption under the random-oracle and ideal-cipher models. Finally, in comparison with recent related literatures, our PGKA protocol is much more efficient in both sides of communication and computation under the same secure setting.*

Keywords: Authentication, Password, Group key, Ideal cipher model, Random oracle model

1. **Introduction.** In the network information age, it is important to structure password-based group key agreement schemes which are designed to provide a set of players, communicating over a public network, and holding a shared human-memorable password with a session key to be used to implement secure multicast sessions, e.g., video conferencing, collaborative computation, file sharing via internet, secure group chat, group purchase of encrypted content and so on. Consequently for the user experience, its very convenient for users only remembering low-entropy passwords to surf the Internet. Therefore we must use cryptographic (i.e., high-entropy) secret instead of password to confirm the security during the communicating process. Nowadays, there are mainly three directions for password-based secure protocols.

(1) Authentication Password-based using smart card scheme[1-8]: It is the most common scheme which checks the validity of the login message to authenticate remote users. The first password-based authentication scheme was proposed by Lamport in 1981[1]. Thereafter, many modified schemes were proposed to improve the security and efficiency [2-7]. Recently, Authentication password-based using smart card scheme turns towards multi-server realm and simple computing step by step[8]. So it is a mixing hardware technology to strengthen security and convenience, however, it must have a smart card device.

(2) Password-based authenticated key exchange (PAKE): This kind of password-based scheme is usually including two-party(a client and a server) or three-party(two clients and a server) which is a hybrid technology of software secure components, e.g., PKI, secret sharing, symmetric cryptography and so on. Bellovin and Merritt [9] were the first to present protocols for password-only authenticated key exchange, where users are required to remember only a short password. These initial works (and others [10-13]) were relatively informal and did not provide definitions or proofs of security. Then, formal definitions and provably secure protocols about the "hybrid" model were proposed [14][15], followed by models for the password-only setting [16-18] and associated protocols with proofs of security in the random oracle/ideal cipher models [16, 17, 19, 35]or in the standard model [18, 20-22]. To further improve the security of the PAKE scheme, recently, some authors use two-server [28-30] method or secret sharing technology [31-34] to construct redundant or threshold password-based authentication protocols.

(3) Password-based group key agreement: To design group key agreement protocols in password-based setting is difficult but is very useful in many application environments . The first work in this area is by Bresson et al [24]. As already mentioned, their proposed scheme is secure in both the random oracle model and the ideal cipher model. Next Lee presents a password-based group key protocol [23] which is not authenticated because there is no way to convince a user that the message that he receives is indeed coming from the intended participant. Recently there are three literatures about password-based group key scheme [25-27, 36] and Abdalla M, et al [36] points out the literature [25] which is subjected to an off-line dictionary attack, however their efficiency is unsatisfactory. Consequently, the article proposes a scheme which aims at designing a structured, efficient and secure password-based group key agreement which is more efficient than the literatures [25-27, 36,40, 41] (in section 6).

Our Contributions: In this paper, we put forward a new simple and efficient PGKA. We present our contributions below: Our proposed scheme is efficient from communication point of view as it requires only 2 rounds and uses symmetric key encryption instead of signature for message authentication in the round 1. And in the round 2, we mainly use hash function and $\oplus$ operations to authenticate each other for all the participants and compute the group session key. These methods reduce the bandwidth of the messages sent and make the protocol faster as compared to the recent password-based group key agreement protocols. Thus the communication efficiency is increased a lot. About security, our PGKA protocol is also provable security which is based on ideal cipher model and random oracle model that are more practical than standard secure model in the real surroundings. The rest of the paper is organized as follows: We outline preliminaries in Section 2. Next, a primary generic construction idea of a PGKA protocol from password is presented in Section 3, followed by a concrete protocol from password scheme in Section 4. Then, the security model and the security analysis are given in Section 5 and efficiency comparison is given in Section 6. This paper is finally concluded in Section 7.

2. **Preliminaries.**

2.1. **Decisional Diffie-Hellman (DDH) assumption.** Let $G =< g >$ be any finite cyclic group of prime order q. The DDH assumption is that it is difficult to distinguish the following real Diffie-Hellman distribution Γ_{real} and and random Diffie-Hellman distribution Γ_{real}.

$$\Gamma_{real} = \{g^x, g^y, g^{xy} | x, y \in_R Z_q\},$$
$$\Gamma_{rand} = \{g^x, g^y, g^z | x, y, z \in_R Z_q\}$$

More formally, if we define the advantage function $Adv_G^{ddh}(A)$ as:

$$Adv_G^{ddh}(A) = |\Pr[A(X) = 1 | X \subset \Gamma_{real}] - \Pr[A(Y) = 1 | Y \subset \Gamma_{rand}]|$$

we say that the DDH assumption holds in group G if $Adv_G^{ddh}(A)$ is negligible for any probabilistic polynomial time adversary A. We denote $Adv_G^{ddh}(t)$ the maximum value of $Adv_G^{ddh}(A)$ overall adversary A running in time at most t.

2.2. **Multi-Decisional Diffie-Hellman (MDDH) assumption.** Define real *Multi Diffie-Hellman distribution* Π_{real} and random *Multi Diffie-Hellman distribution* Π_{rand} of size n as follows [26]:

$$\Pi_{real} = \{\{g^{x_i}\}_{1 \le i \le n}, \{g^{x_j x_{j+1}}\}_{1 \le j \le n} | x_1, ..., x_n \in_R Z_q\}$$

$$\Pi_{rand} = \{\{g^{x_i}\}_{1 \le i \le n}, \{g^{y_j}\}_{1 \le j \le n} | x_1, ..., x_n, y_1, ..., y_{n-1} \in_R Z_q\}$$

Define the advantage function $\mathrm{Adv}_G^{mddh}(A)$ as

$$\mathrm{Adv}_G^{mddh}(A) = |\Pr[A(X) = 1 | X \in \Pi_{real}] - \Pr[A(Y) = 1 | Y \in \Pi_{rand}]|$$

Lemma 1.: For any group G and any integern, the MDDH problem can be reduced to the DDH problem [24]:

$$\mathrm{Adv}_G^{mddh}(t) \le n\mathrm{Adv}_G^{ddh}(t)$$

2.3. **Symmetric encryption.** A symmetric encryption scheme $E_k(Kgen, E, D)$ consists of three algorithms as follows:

- Randomized Key Generation Algorithm $Kgen$: it returns a key k drawn from the key space $Keys(E_k)$ at random.
- Encryption Algorithm E: it takes key $k \in Keys(E_k)$ and a plaintext $M \in \{0,1\}^*$ as the inputs and outputs a ciphertext $C \in \{0,1\}^*$. We write $C = E_k(M)$
- Decryption Algorithm D: it takes key $k \in Keys(E_k)$ and a ciphertext $C \in \{0,1\}^*$ as the inputs and outputs a plaintext $M \in \{0,1\}^*$. We write $M = E_k(C)$

3. **Structured Group Key Agreement from Password.** We now consider the generic construction for a two-round group key agreement from password. All group participants $U_1, U_2, ..., U_n$ are organized in an ordered chain and U_{i+1} is the successor of U_i.Based on shared low entropy secret password $\boldsymbol{pw}$,the temporary two-party symmetric session key can be computed in a parallel algorithm and which is used as the shared secret between the participant U_i and its successor $U_{i+1}, i = 1, ..., n$.

Specifically, the outgoing message of the symmetric encryption including shared secret password $\boldsymbol{pw}$ becomes the key encapsulation, and the session key computation process at the receiver can be used as the decapsulation algorithm to retrieve and compute the symmetric key locally. The message with symmetric encryption including password $\boldsymbol{pw}$gives confidentiality protection and origin authentication to the group key agreement protocol. The structure of the kind of group key agreement from password is illustrated in Fig. 1 which includes the following two rounds.

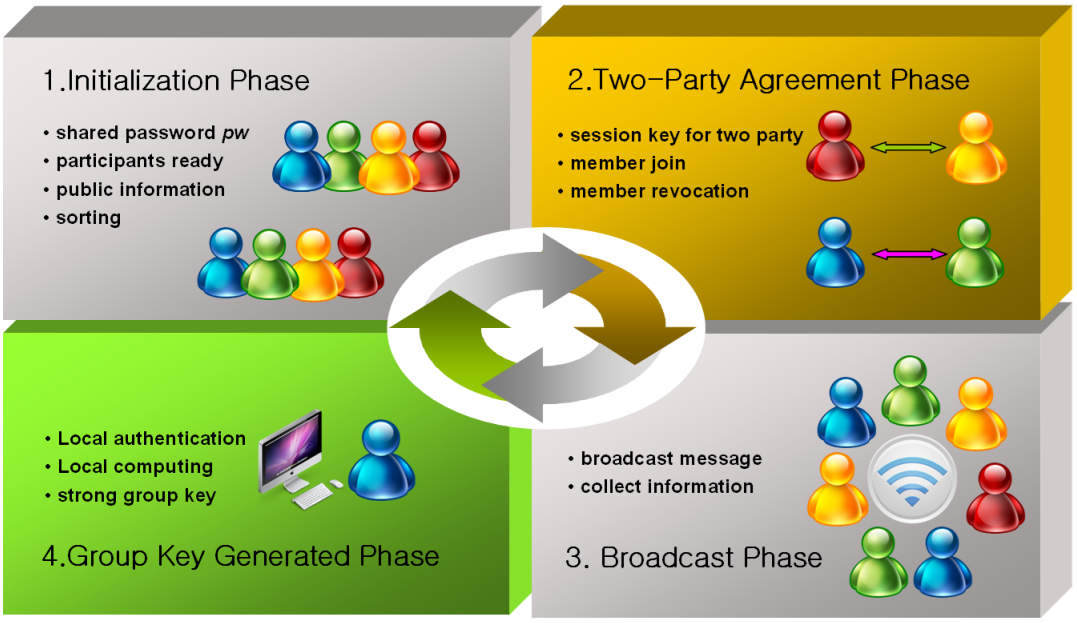

FIGURE 1. Structure of the PGKA phases (The phases are presented clockwise)

Round 1.

(1) Let $U = \{U_1, U_2, ..., U_n\}$ be the set of protocol participants. All the participants $U_1, U_2, ..., U_n$ run the following process. The participant U_i sends message ψ_i to its successor $U_{i+1}, i = 1, ..., (n-1)$. At the same time, the participant U_{i+1} sends message $\Psi_i^{'}$ to its predecessor $U_i, i = 1, ..., (n-1)$. This process is presented in the following Fig. 2.

$$U_1 \xrightarrow{\psi_1} U_2 \xrightarrow{\psi_2} U_3 \xrightarrow{\psi_3} U_4 \cdots\cdots \xrightarrow{\psi_{n-2}} U_{n-1} \xrightarrow{\psi_{n-1}} U_n \xrightarrow{\psi_n} U_1,$$
$$U_1 \xleftarrow{\psi_1^{'}} U_2 \xleftarrow{\psi_2^{'}} U_3 \xleftarrow{\psi_3^{'}} U_4 \cdots\cdots \xleftarrow{\psi_{n-2}^{'}} U_{n-1} \xleftarrow{\psi_{n-1}^{'}} U_n \xleftarrow{\psi_n^{'}} U_1,$$

FIGURE 2. Exchange temporary information

(2) First, the successor $U_{i+1}, i = 1, ..., (n-1)$, verify the validity of the messages ψ_i and $\psi_{i+1}^{'}$ from U_i and U_{i+2}.Then participants $U_i, i = 2, ..., n$, compute the secret $SK_{i-1,i}$ and $SK_{i,i+1}$ (U_1 computes $SK_{1,2}$ and $SK_{n,1}$), which is the session secret computed by a sender U_i and its successor U_{i+1} and its predecessor U_{i-1}. We can present this via the following Fig. 3.

$$U_1 \underset{\text{Local Computed}}{\overset{SK_{1,2}}{\longleftrightarrow}} U_2 \underset{\text{Local Computed}}{\overset{SK_{2,3}}{\longleftrightarrow}} U_3 \cdots\cdots U_{n-1} \underset{\text{Local Computed}}{\overset{SK_{n-1,n}}{\longleftrightarrow}} U_n \underset{\text{Local Computed}}{\overset{SK_{n,1}}{\longleftrightarrow}} U_1$$

FIGURE 3. Two shared session key between a participant and its successor, its predecessor

The above process can be simultaneous and parallel.

Round 2.

(1) The participants $U_i, i = 2, ..., n$, compute and broadcast X_i, where $X_i = B_{i-1} \oplus B_i = H(SK_{i-1,i}, ID_{session}) \oplus H(SK_{i,i+1}, ID_{session})$. Note that the first participant U_1 computes and broadcasts $X_1 = B_n \oplus B_1 = H(SK_{n,1}, ID_{session}) \oplus$

$H(SK_{1,2}, ID_{session})$. Here,$H(\bullet)$ is a one-way hash function and $ID_{session}$ is the public ephemeral information that consists of participants' identities and a nonce, aiming to make the protocol secure against known-key attacks.

(2) Finally, with the two secrets session keys $SK_{i-1,i}$ and $SK_{i,i+1}$, the participants $U_i\,(i = 1, ..., n)$ computes $B_i = H(SK_{i,i+1}, ID_{session})$ and $B_{i-1} = H(SK_{i-1,i}, ID_{session})$ for extracting from X_i to get all $B_j\,(j = 1, ..., n)$ as the following Fig. 4.

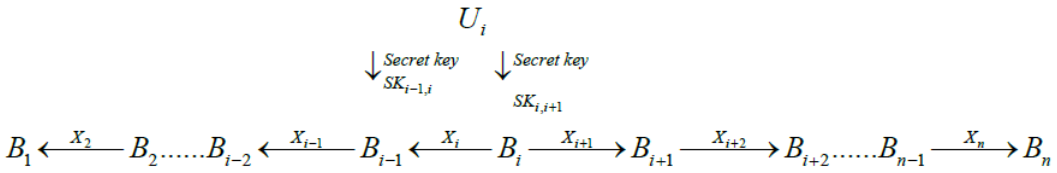

FIGURE 4. Participant U_i obtains all secrets by using its two secret session keys

(3) Firstly, the participant $U_i, (i = 1, ..., n)$ compares B_{i-1} and $H(SK_{i-1,i}, ID_{session})$ locally. After colletcing all the X_i, the $U_i, (i = 1, ..., n)$ will verify if $X_1 \oplus X_2 \oplus X_3 \oplus ... \oplus X_{n-1} \oplus X_n = 0$ holds. Otherwise, output an error symbol $\perp$ and abort. Each participant computes the group session key$GSK_i = H(B_1 \parallel B_2 \parallel ... \parallel B_n)$. Because each participant can locally computes $B_i (1 \leq i \leq n)$ by his own two secret session keys, $GSK_1 = GSK_2 = ... = GSK_n$ holds obviously. What this means is that all the participants agree on a common strong group session key.

A member revocation: Assume that a participant $U_x\,(1 < x < n)$ leaves the group. Then group members change the group size into $(n-1)$.The participants U_{x-1} and U_{x+1} respectively remove the shared values $k_{x-1,x}$ and $k_{x,x+1}$ with U_x. The participant U_{x+1} becomes the new successor of participan U_{x-1}.Aiming to update group key, the participant U_{x-1} needs to send new message $\tilde{\psi}_{x-1}$ to its new successor U_{x+1} and U_{x+1} needs to send new message $\tilde{\psi}'_x$ to its new predecessor U_{x-1}. Then, the participant U_{x+1} verifies the validity of the message $\tilde{\psi}_{x-1}$ and computes the secret $SK_{x-1,x+1}$, and U_{x-1} verifies the validity of the message $\tilde{\psi}'_x$ and computes the secret $SK_{x-1,x+1}$ which is a new shared secret between U_{x-1} and U_{x+1}. Each party U_j that follows U_x changes their index to $(j-1)$. Then, from Step (1) of Round 2, all the $(n-1)$ participants implement the above protocol to get a new group session key.

A new member join: Assume that a new entity joins the group whose size is n. Then, the new participant U_{n+1}, becomes the successor of participant U_n and the participant U_1 becomes the successor of participant U_{n+1}.

The participant U_n sends message $\tilde{\psi}_n$ to its new successor U_{n+1} while U_{n+1} sends message $\tilde{\psi}'_n$ to U_n. U_{n+1} sends $\tilde{\psi}_{n+1}$ to U_1 and U_1 send $\tilde{\psi}'_{n+1}$ to U_{n+1} as the following Fig. 5(a).

From the message $\tilde{\psi}_n$ and $\tilde{\psi}'_{n+1}$, the new participant U_{n+1} verifies the validity of the message and computes the secret $SK_{n,n+1}$ which is the new shared secret between U_n and its new successor U_{n+1}. At the same time, the first participant U_1 updates its secret with $SK_{n+1,1}$ as the following Fig. 5(b). Then, from step (1) of round 2, the participants in the group implement the above protocol to get a new group session key.

Remark. 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. A password setup algorithm is another problem and we need not handle it in this paper.

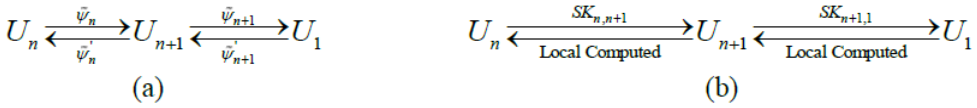

FIGURE 5. A new member join case

4. **A Concrete Password-based Group Key Agreement Protocol.** The notation used hereafter is shown in Table 1.

TABLE 1. Notations

Symbol	Definition
Ui , IDi	**The Participant i and its identity information**
U	**set of protocol participants**
p,q	**Two large and different safe primes, where p = 2q+1**
G_q	**a subgroup of quadratic residues in** Z_p^* **, that is** $G_q = \{i^2 \mid i \in Z_p^*\}$
Ti	**Timestamp for Ui**
E_k	**A secure symmetric encryption with the key** k
x_i	$x_i \in_R Z_q$ **chosen by each Ui**
$\oplus$	**A bitwise Xor operator**
$\|$	**Means that two adjacent messages are concatenated**
H	**A secure one-way hash that** $0,1\}^* \to \{0,1\}^{l_H}$**,** **where** l_H **denotes outputted bit-length of H**

Round 1.
Let $U = \{U_1, U_2, ..., U_n\}$ be the set of protocol participants. All the participants $U_1, U_2, ..., U_n$ run the following process. We assume that $x_i \in [1, ..., q-1]$ is the secret key of the participant U_i with identity $ID_i (i = 1, ..., n)$, and $y_i \equiv g^{x_i} \bmod p$ is the corresponding temporary public key. The participants $U_i (i = 1, ..., n)$ perform the following steps:

Step1 U_i chooses $x_i \in_R Z_q$, and computes $y_i^{'} = E_{k_i}(y_i, ID_i, T_i)$ where $y_i \equiv g^{x_i} \bmod p$, $k_i = H(U, i, pw)$, U is the set of $U_1, U_2, ..., U_n$, T_i is the timestamp. Then U_i sends $< ID_i$, $y_i^{'} >$ to U_{i+1} and U_{i-1} (U_1 sends $< ID_1$, $y_i^{'} >$ to U_2 and U_n.

Step2 After receiving the $y_{i-1}^{'}$ and $y_{i+1}^{'}$ from U_{i-1} and U_{i+1}, U_i computes the k_{i-1} and k_{i+1} using shared password $\boldsymbol{pw}$. Then U_i extracts $y_{i-1}^{'}$ and $y_{i+1}^{'}$ to get the $< y_{i-1}, ID_{i-1}, T_{i-1} >$ and $<y_{i+1}, ID_{i+1}, T_{i+1} >$, and verifies the ID and the two timestamps. If messages are fresh, U_i computes the two two-party session keys $SK_{i-1,i} = H(ID_i, ID_{i-1}, T_i, T_{i-1}, (g^{x_{i-1}})^{x_i} \bmod p)$ and $SK_{i,i+1} = H(ID_i, ID_{i+1}, T_i, T_{i+1}, (g^{x_{i+1}})^{x_i} \bmod p)$. The first participant U_1 computes $SK_{1,n} = H(ID_1, ID_n, T_1, T_n, (g^{x_n})^{x_1} \bmod p)$ and $SK_{1,2} = H(ID_1, ID_2, T_1, T_2, (g^{x_2})^{x_1} \bmod p)$

Round 2.
Each participant computes and broadcasts $X_i = H\left(SK_{i-1,i}, ID_{session}\right) \oplus H\left(SK_{i,i+1}, ID_{session}\right)$. The U_1 computes and broadcasts $X_1 = H\left(SK_{n,1}, ID_{session}\right) \oplus H\left(SK_{1,2}, ID_{session}\right)$.

Finally, each participant $U_i (i = 1, ..., n)$ uses the shared secret two-party session keys $SK_{i-1,i}$ and $SK_{i,i+1}$ to get all $B_j (j = 1, ..., n)$ as the following algorithms:

(1) When $j \geq i$,

$$\begin{aligned} B_i &= H(SK_{i,i+1}, ID_{session}); \\ B_{i+1} = X_{i+1} \oplus B_i &= H(SK_{i,i+1}, ID_{session}) \oplus H(SK_{i+1,i+2}, ID_{session}) \oplus H(SK_{i,i+1}, ID_{session}) \\ &= H(SK_{i+1,i+2}, ID_{session}); \\ B_{i+2} = X_{i+2} \oplus B_{i+1} &= H(SK_{i+1,i+2}, ID_{session}) \oplus H(SK_{i+2,i+3}, ID_{session}) \oplus H(SK_{i+1,i+2}, ID_{session}) \\ &= H(SK_{i+2,i+3}, ID_{session}); \\ &\ldots\ldots; \end{aligned}$$

$$\begin{aligned} B_n = X_n \oplus B_{n-1} &= H(SK_{n-1,n}, ID_{session}) \oplus H(SK_{n,1}, ID_{session}) \oplus H(SK_{n-1,n}, ID_{session}) \\ &= H(SK_{n,1}, ID_{session}). \end{aligned}$$

(2) when j < i,

$$\begin{aligned} B_{i-1} &= H(SK_{i-1,i}, ID_{session}); \\ B_{i-2} = X_{i-1} \oplus B_{i-1} &= H(SK_{i-2,i-1}, ID_{session}) \oplus H(SK_{i-1,i}, ID_{session}) \oplus H(SK_{i-1,i}, ID_{session}) \\ &= H(SK_{i-2,i-1}, ID_{session}); \\ B_{i-3} = X_{i-2} \oplus B_{i-2} &= H(SK_{i-3,i-2}, ID_{session}) \oplus H(SK_{i-2,i-1}, ID_{session}) \oplus H(SK_{i-2,i-1}, ID_{session}) \\ &= H(SK_{i-3,i-2}, ID_{session}) \\ &\ldots\ldots; \\ B_1 = X_2 \oplus B_2 &= H(SK_{1,2}, ID_{session}) \oplus H(SK_{2,3}, ID_{session}) \oplus H(SK_{2,3}, ID_{session}) \\ &= H(SK_{1,2}, ID_{session}). \end{aligned}$$

To sum it up, we can see the Table2.

TABLE 2. The value of B_i

Notation	B_1	B_2	... ...	B_i	... ...
Value	$H(SK_{1,2}, ID_{session})$	$H(SK_{2,3}, ID_{session})$	... ...	$H(SK_{i,i+1}, ID_{session})$	... ...

Here, $ID_{session}$ is the public ephemeral information that consists of participants' identities and a nonce, aiming to make the protocol secure against known-key attacks. Furthermore, each participant $U_i (i = 1, ..., n)$ verifies if $X_1 \oplus X_2 \oplus X_3 \oplus ... \oplus X_{n-1} \oplus X_n = 0$ holds and all participants will continue to compute the group key. If not, output an error symbol $\perp$ and abort. After all participants accomplish the verifying, they compute the group session key $GSK_i = H(B_1||B_2||...||B_n)$. This will be the common group session key agreed by all participants. The example for N=4 is shown in Fig. 6.

5. **Security Consideration.**

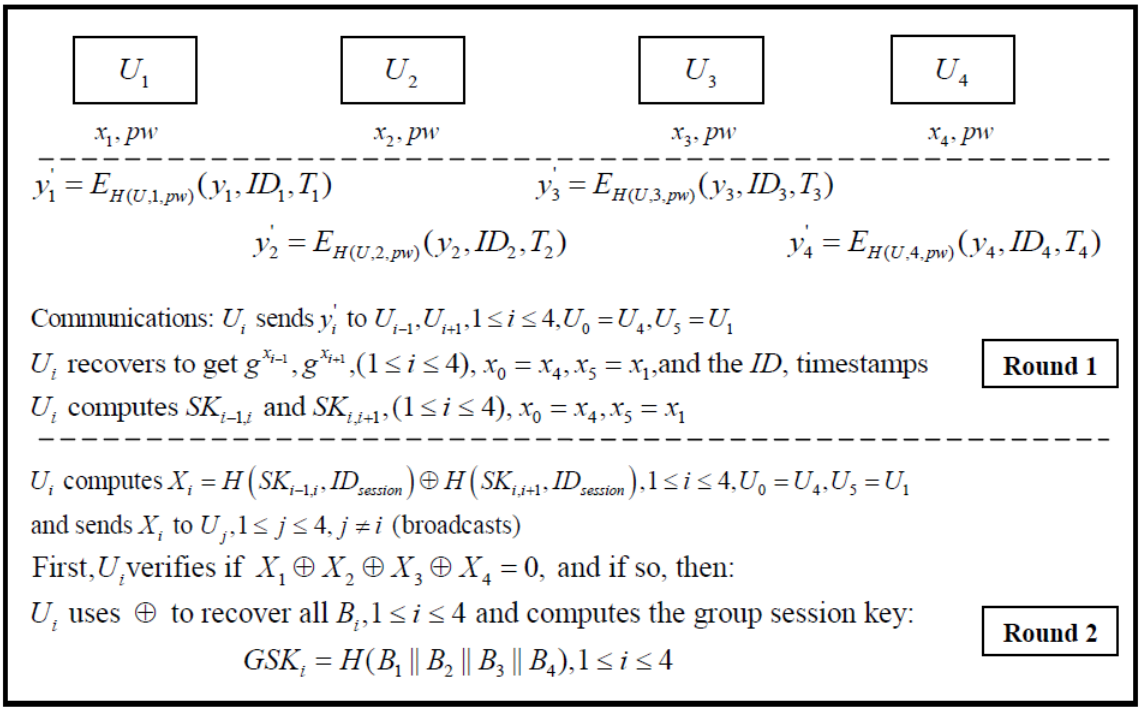

FIGURE 6. Password-based group key agreement among $n = 4$ users

5.1. **Security Model.** We describe below our security model following closely the Real-or-Random (ROR) model of Abdalla et al.[36, 38], instead of the Find-then-Guess (FTG) model of Bellare and Rogaway [16] as standardized by Bresson et al. [24, 39].

A protocol P for password-based group key agreement assumes that there is a set $U = \{U_1, U_2, ..., U_n\}$ of n fixed participants, who share a low entropy secret password $\boldsymbol{pw}$ drawn uniformly from a small dictionary of size N. This security model allows concurrent execution of the protocol among n participants, so each of participants may have several instances (or called oracles) involved in distinct ones. We assume that participants will execute the protocol faithfully and adversary never becomes a participant in the protocol. This adversarial model allows concurrent execution of the protocol among n participants. The interaction between the adversary A and the protocol participants occur only via oracle queries, which model the adversarys capabilities in a real attack.

We use the j-th to denote the instance of U_i by U_i^j. During the process of protocol, the adversary can control over all communication in the external network. The interaction between the adversary and participants occur only via oracle queries, which model the adversary's capabilities in a real attack. These queries are as follows [26, 36]:

Send(U_i^j, m): The adversary can carry out an active attack by this query. The output of the query is the response generated by the instance U_i^j upon receipt of message m according to the execution of the protocol P. The adversary A is allowed to prompt the unused instance U_i^j to initiate the protocol by invoking Send (U_i^j, "Start").

Test(U_i^j): This query models the misuse of the session key by instance (U_i^j). Once the instance U_i^j has accepted a session key, the adversary can attempt to distinguish it from a random key as the basis of determining security of the protocol. A random

bit b is chosen: if $b = 0$ then session key is returned while if $b = 1$ then a random key is returned.The random key must however be consistent among the n 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 participants have the same view, they thus have the same random key but independent of the real view.

Finally adversary outputs a guess bit $b^{'}$. Such an adversary A is deemed to win the game if $b = b^{'}$ where b is the hidden bit used by the Test oracle. Let **Suc** denote the event that the adversary wins the game. We define $\mathrm{Adv}_P^{aka}(A,t) = |2\Pr[\mathrm{Suc}] - 1|$ to be the advantage of the active adversary A in attacking the protocol P [21]. Protocol P is a secure password-based authenticated group key agreement protocol resistant to the dictionary attacks if A's advantage is a negligible function for any adversary A running in time at most t.

In the ROR model, Execute, Send and Test queries can be asked by the adversary. Execute queries were described to model passive attack. However, they can easily be simulated using the Send queries. In security analysis of our protocol, we refine the way only using Send and Test queries. Let q_{active} denotes the number of messages made by the adversary, without including those he has just forwarded. This number upper-bounds the number of attempts that the adversary guesses the password. In addition, let $q_{session}$ denotes the number of sessions the adversary has initiated, thus $nq_{session}$ upper-bounds the number of messages the adversary has sent in the protocol. We emphasize that this is stronger than considering Execute and Send queries: while being polynomial 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.

5.2. **Security Proof.** Our protocol is based on the DDH assumption and security is achieved in both the random oracle model[1] and the ideal cipher model[2].

Lemma 2. Let E, and F be events defined on a probability space:
If $\Pr[\mathrm{E}| \neg \mathrm{F}] = \Pr[E^{'} \mid \neg \mathrm{F}]$, then *mid* $\Pr[\mathrm{E}]$-$\Pr[E^{'}] \mid \leq \Pr[\mathrm{F}]$ [37].

Theorem 1. Under the DDH assumption, the password-based group key agreement protocol P described in section 4 satisfies the following:

$$\mathrm{Adv}_P^{aka}(t) \leq \frac{2q_{active}}{N} + 4nq_{session}\mathrm{Adv}_G^{ddh}(t) + 2q_H^4/(2^{l_H})^3 + 2q_H(2q_H + q_D)/2^{l_H}$$

$$+\frac{4q_H^2}{2^{l_H}|G|^2} + \frac{q_D + 2q_E + 8q_H + (q_E + q_D)^2}{|G|}$$

Where $\mathrm{Adv}_P^{aka}(t)$ denote the advantage of any active adversary A breaking the semantic security of the session key, running in time t and making at most q_{active} active attempts within at most $q_{session}$ sessions. q_H, q_E, q_D denote the number of oracle queries of this adversary making to the random oracles H, and to the ideal-cipher oracles E and D, respectively. N is the size of the dictionary.
Please see appendix for proof for **Theorem 1**.

5.3. **Efficiency comparison.** 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 3 compares our protocol and Bresson et al.'s password-based group key agreement protocol (BCP) [24] Dutta et al.'s protocol

[25] and recent literatures [26][27] 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, DDH stands for Decisional Diffie-Hellman assumption, PDDH means Parallel Decisional Diffie-Hellman assumption, IC denotes ideal cipher model and RO stands for random oracle model).

TABLE 3. Protocol comparison

Protocol	Communication					Computation									Hardness Assumption	Security Model
	R	BL	PTP	B	C	Exp	*H*	XOR	Enc	Dec	PK	*S*	*V*	M		
[24]	n	$n\|e\|$	$2n$ -2	-	No	$2n$	-	-	1	2	No	-	-	-	TG-CDH,M-DDH	IC, RO
[25]	2	$2\|e\|$	n +1	2	Yes	3	4	n -1	2	n+1	No	-	-	-	CDH	IC, RO
[26]	2	$2\|e\|$	-	4	Yes	5	n+4	-	1	2	No	-	-	n-1	DDH, M-DDH	IC, RO
[27]	4	$4\|e\|$	-	6	No	/	3	-	1	n-1	Yes	n	$2n$-2	/	/	IC, RO
[36]	2	$2\|e\|$	-	4	Yes	3	3	-	1	2	No	-	-	n-1	DDH, PDDH	IC, RO
Ours	2	$2\|e\|$	2	1	Yes	3	6	n	1	2	No	-	-	-	DDH	IC, RO

Notes:
n: total number of users in a group
R: total number of rounds
BL: maximum bit length of messages sent per participant
PTP: maximum number of point-to-point communication per participant
B: broadcasts per user
C: parallelism in the first round
Exp: maximum number of modular exponentiations computed per participant
H: maximum number of hash function evaluation per participant
XOR: maximum number of XOR operations computed per participant
Enc: maximum number of symmetric key encryptions per participant
Dec: maximum number of symmetric key decryptions per participant
PK: using asymmetric encryption per participant
S: Signature per participant
V: verify Signature per participant
M: multiplication modular per participant
-: zero or nothing
/: no concrete algorithm or Hardness Assumption
$|e|$: maximum size of an encrypted plaintext

From the Table 3, we can see easily that our protocol is much more efficient (no matter how communication or computation under the same security model) than recent related literatures [24-27, 36]. For literature [24], it used serial method to deliver the messages which consumes mass communication and computation so that as the size of the participants increases linearly, the cost and difficulty to make it increases linearly too. Contrasting with literature [24], ours protocol only need constant round and constant modular exponentiations(see "R", "BL", "PTP" and "Exp" in the table 3). Then the main different between the literature [25] and our protocol is the computation aspect. The literature [25] uses the symmetric key encryptions to transfer messages in the round 2 which leading to computation increasing linearly with the size of the participants increasing linearly. However ours protocol is still constant round for computation(see "Dec"and "PTP" in the table 3). About The literature [26], it has too much multiplication modular operations. The literature [27] uses asymmetric encryption and signature to get the group session key which consume mass computing because each participant must sign n signatures and verify 2n-2 signatures. Moreover, The literature [27] is not given the concrete algorithm so that we can't analyze some computing efficiency. Finally, the main difference between ours protocol and literature [36] is that we use the XOR operations to substitute for the multiplication modula operations so that can improve computing efficiency in the round 2.

6. **Conclusion.** We described an efficient password-based group key exchange protocol, mainly based on symmetric key encryption, hash function and $\oplus$ operations. The protocol is proven secure against dictionary attacks under the DDH assumption,

in the ideal-cipher and random oracle models. In the future, we will study the secure password-based efficient group key agreement protocol under the standard model instead of random oracle model, and give the PGKA protocol more secure properties.

Acknowledgement. This research was supported by Liaoning Provincial Natural Science Foundation of China (Grant No. 20102202, 201102201) and Liaoning Baiqianwan Talents Program (Grant No.2011921046).

REFERENCES

[1] L. Lamport, Password authentication with insecure communication.*Communications of the ACM*, vol. 24, no. 11, pp. 770-772, 1981.
[2] R. Lennon, S. Matyas, C. Meyer, Cryptographic authentication of time invariant quantities, *IEEE Trans. Communications*, vol. 29, no. 6, pp. 773-777, 1981.
[3] S.M. Yen, K.H. Liao, Shared authentication token secure against replay and weak key attack, *Information Processing Letters*, vol. 62, no. 2, pp. 77-80, 1997.
[4] E.J. Yoon, E.K. Ryu, K.Y. Yoo, Further improvement of an efficient password based remote user authentication scheme using smart cards, *IEEE Trans. Consumer Electronics* , vol.50, no. 2, pp. 612-614, 2004.
[5] H.C Hsiang, W.K. Shih, Weaknesses and improvements of the Yoon-Ryu-Yoo remote user authentication scheme using smart cards, *Computer Communications*, vol. 32, no. 4, pp. 649-652, 2009.
[6] D.B. He, J.H. Chen, J. Hu, Weaknesses of a remote user password authentication scheme using smart card, *International Journal of Network Security*, vol. 13, no. 1, pp. 58-60, 2011.
[7] W.B. Hsieh, J.S. Leu, Exploiting hash functions to intensify the remote user authentication scheme, *Computers & Security* , vol. 31, no. 6, pp. 791-798, 2012.
[8] J.L. Tsai, Efficient multi-server authentication scheme based on one-way hash function without verification table, *Computers & Security* , vol. 27, no. 3-4, pp. 115-121, 2008.
[9] S.M. Bellovin, M. Merritt, Encrypted key exchange: password-based protocols secure against dictionary attacks, *IEEE Symposium on Research in Security and Privacy*, pp. 72-84, 1992.
[10] S.M. Bellovin, M. Merritt., Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise, *Proc. of the 1st ACM conference on Computer and communications security*, pp. 244V250, 1993.
[11] D. Jablon, Strong password-only authenticated key exchange, *ACM Comput. Commun*, vol. 26. no. 5, pp. 5-20, 1996.
[12] S. Lucks, Open key exchange: how to defeat dictionary attacks without encrypting public keys, *Proc. of the Security Protocols Workshop*, LNCS 1361, Springer, pp. 79-90, 1997.
[13] T. Wu, The secure remote password protocol, *Proc. of Internet Society Symp, Network and Distributed System Security*, pp. 97-111, 1998.
[14] S. Halevi, H. Krawczyk, Public-key cryptography and password protocols, *ACM Transactions on Information and System Security*, vol. 2, no. 3, pp. 230-268, 1999.
[15] M. Boyarsky, Public-key cryptography and password protocols: The multi-user case, *Proc. of the 7st ACM conference on Computer and communications security*, pp. 63-72, 1999.
[16] M. Bellare, D. Pointcheval, P. Rogaway, Authenticated key exchange secure against dictionary attacks, *Proc. of the Security Protocols Workshop*, LNCS 1807, Springer, pp. 139-155, 2000.
[17] V. Boyko, P. MacKenzie, S. Patel, Provably secure password-authenticated key exchange using DiffieVHellman, *Proc. of the Security Protocols Workshop*, LNCS 1807, Springer, pp. 156-171, 2000.
[18] O. Goldreich, Y. Lindell, Session-key generation using human passwords only, *Journal of Cryptology*, vol. 19, no. 3, pp. 241-340, 2006.
[19] P. MacKenzie, S. Patel, R. Swaminathan, Password-authenticated key exchange based on RSA, *Proc. of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, LNCS 1976, Springer, pp. 599-613, 2000.
[20] J. Katz, R. Ostrovsky, M. Yung, Efficient password-authenticated key exchange using human-memorable passwords, *EUROCRYPT 2001*, LNCS 2045, Springer, pp. 475-494, 2001.
[21] R. Gennaro, Y. Lindell, A framework for password-based authenticated key exchange, *EUROCRYPT 2003*, LNCS 2656, Springer, pp. 524-543, 2003.

[22] S. Jiang, G. Gong, Password based key exchange with mutual authentication, *EUROCRYPT 2005*, LNCS 3357, Springer, pp. 267-279, 2005.
[23] S. M. Lee, J. Y. Hwang and D. H. Lee, Efficient password-based group key exchange, *the proc. of TrustBus 2004*, LNCS 3184, Springer, pp. 191V199, 2004.
[24] E. Bresson, O. Chevassut and D. Pointcheval, Group Diffie-Hellman key exchange secure against dictionary attack, *ASIACRYPT 2002*, LNCS 2501, Springer, pp. 497V514, 2002.
[25] R. Dutta, R. Barua, Password-Based Encrypted Group Key Agreement.*International Journal of Network Security*, vol.3, no.1, PP.23V34, 2006.
[26] M.H Zheng, H.H Zhen, J Li, G.H. Cui, Efficient and provably secure password-based group key agreement protocol.*Computer Standards & Interfaces*, vol.31, no.5, PP.948V953, 2009.
[27] H. Li, C.K Wu, J. Sun. A general compiler for password-authenticated group key exchange protocol.*Information Processing Letters*, vol. 110, no.4, 160V167,2010.
[28] J. Brainard, A. Juels, B. Kaliski, M. Szydlo, Nightingale: A new two-server approach for authentication with short secrets, *Proc. 12th USENIX Security Symp* pp. 201V213, 2003.
[29] M. Szydlo, B. Kaliski, Proofs for two-server password authentication, *RSA Cryptographers Track 2005*, LNCS 3376, Springer, pp. 227V244, 2005.
[30] K. Jonathan, M. Philip , T. Gelareh, G. Virgil, Two-server password-only authenticated key exchange,*Applied Cryptography and Network Security*, LNCS 3531, Springer, pp. 1V16, 2005.
[31] G.R. Blakley, Safeguarding Cryptographic Keys, *Proc. AFIPS 1979 National Computer Conf*, pp. 313-317, 1979.
[32] A. Shamir, How to Share a Secret, *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.
[33] P. MacKenzie, T. Shrimpton, M. Jakobsson, Threshold password-authenticated key exchange, *CRYPTO 2002*, LNCS 2442, Springer, pp. 385V400, 2002.
[34] M. Di Raimondo, R. Gennaro, Provably secure threshold password-authenticated key exchange, *EUROCRYPT 2003*, LNCS 2656, Springer, pp. 507V523, 2003.
[35] M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, *Proc. of the 1st ACM conference on Computer and communications security*, pp. 62V73, 1993.
[36] M. Abdalla, E. Bresson , O. Chevassut, D. Pointcheval, Password-based group key exchange in a constant number of rounds, *Public Key Cryptography - PKC 2006*, LNCS 3958, Springer, pp. 427V442, 2006.
[37] J.W. Byun, D.H. Lee, J.I. Lim, EC2C-PAKA: an efficient client-to-client password-authenticated key agreement, *Information Sciences*, vol. 177, no. 19, pp. 3995-4013, 2007.
[38] M. Abdalla, P.A. Fouque, D. Pointcheval, Password-based authenticated key exchange in the three-party setting, *Public Key Cryptography - PKC 2005*, LNCS 3386, Springer, pp. 65-84, 2005.
[39] E. Bresson, O. Chevassut, D. Pointcheval, J. Quisquater, Provably authenticated group Diffie-Hellman key exchange, *Proc. of the 8th ACM conference on Computer and Communications Security*, pp255-264, 2001.
[40] T.Y. Wu, Y.M. Tseng, T.T. Tsai, A revocable ID-based authenticated group key exchange protocol with resistant to malicious participants, em Computer Networks, vol. 56, no. 12, pp. 2994-3006, 2007.
[41] E. Bresson, O. Chevassut, D. Pointcheval, J. Quisquater, Provably authenticated group Diffie-Hellman key exchange, *Proc. of the 8th ACM conference on Computer and Communications Security*, pp. 255-264, 2001.
[42] T.Y. Wu, Y.M. Tseng, T.T. Tsai, A revocable ID-based authenticated group key exchange protocol with resistant to malicious participants, *Computer Networks*, vol.56, no. 12, pp. 2994-3006, 2012.
[43] T.Y. Wu, Y.M. Tseng, Towards ID-based authenticated group key exchange protocol with identifying malicious participants, *Computer Science*, vol.23, no. 2, pp. 315-334, 2012.

Appendix: Proof of Theorem 1. Proof. We define a sequence of experiments from the experiment $\mathbf{Exp}_0$ to $\mathbf{Exp}_8$ incrementally. In each of them, we simulate a sequence of adversary behaviors and environments, and measure the advantage of an adversary A about the agreed session key. That means,in each experiment $\mathbf{Exp}_i$, when the adversary A asks a Test query, a coin for bit b is flipped to specify the answer of

a rea incrementally. In each of them, we simulate a sequence of adversary behaviors and environments, and measure the advantage of an adversary A about the agreed session key. That means, in each experiment l session key or a random nonce. Then A guesses bit b and outputs the guessed bit $b^{'}$, Let Suc_ibe the success event that $b^{'} = b$, in the $\mathbf{Exp}_i$ and $\Pr[\mathrm{Suc}_i]$ is its probability. At the end of the experiments, we measure the probability $|\Pr[\mathrm{Suc}_i]\text{-}\Pr[\mathrm{Suc}_{i-1}]|$ between $\mathbf{Exp}_i$ and $\mathbf{Exp}_{i-1}$, for $1 \leq i \leq 8$. By using each difference of probability, we finally get the result of **Theorem 1**.

Experiment $\mathbf{Exp}_0$. It is a real world protocol in the random-oracle and ideal-cipher models. An adversary A can query the hash oracles H, the ideal-cipher oracles E, D, and all instances of users. The advantage of A in this real protocol is defined as $\mathrm{Adv}_P^{aka} = 2\Pr[\mathrm{Suc}_0] - 1$

Experiment $\mathbf{Exp}_1$. In this experiment, We simulate the random oracles H in a classical way by maintaining the lists L_H(initially empty) by oracles query as follows Fig. 7.

Hash query

When A asks a H hash query of the form (S,i,m,r) in L_H :

(1) If a record (S,i,m,r) has existed, then the answer is r ;

(2) ElseIf a record (S,i,m,r) has not existed, then choose a random $r \in \{0,1\}^l$ and update record (S,i,m,r) to the L_H .

where i indicates user i and while S indicates the session with which we are dealing.

FIGURE 7. Oracles query for hash

Define the collision event in the output of the H function by Col_H, The probability of such bad event upper-bounder is $q_H^2/2^{l_H}$ by the birthday paradox. It is clear that $\mathbf{Exp}_1$ and $\mathbf{Exp}_0$ are perfectly indistinguishable unless that the bad event Bad1($=\mathrm{Col}_H$(step1 round1)$\vee$ Col_H(round2 broadcast X_i) $\vee$ Col_H(round2 compute session key)) does occur. Therefore, we get:

$$\Pr[\mathrm{Suc}_1|\neg\mathrm{Bad}_1] = \Pr[\mathrm{Suc}_0|\neg\mathrm{Bad}_1]$$

By **lemma 2**, we have:

$$|\Pr[\mathrm{Suc}_1] - \Pr[\mathrm{Suc}_0]| \leq \Pr[\mathrm{Bad}_1] \leq q_H^2/2^{l_H} + q_H^4/(2^{l_H})^3 + q_H^2/2^{l_H} = 2q_H^2/2^{l_H} + q_H^4/(2^{l_H})^3$$

Where $q_H^4/(2^{l_H})^3$ means when broadcast the form $X_i = H(\cdot) \oplus H(\cdot)$ in round2 that bad event Col_H (round2 broadcast X_i)=$(q_H^2/2^{l_H} \wedge 1/2^{l_H} \wedge q_H^2/2^{l_H})$=$q_H^4/(2^{l_H})^3$.

Experiment $\mathbf{Exp}_2$. This experiment simulates the ideal-cipher oracles $E$and D through maintaining a list $L_{E,D}$ which keeps track of the previous queries-answers and that links each query to a specific user. Records of $L_{E,D}$ are of the form $(S,i,e,k,y,\Lambda,y^{'})$, where $\Lambda \in \{\mathrm{enc},\mathrm{dec}\}$. Such record means that $E_k(y) = y^{'}$, and $\wedge$ indicates which kind of queries generated the record. The index i indicates which user is associated with the key k, while S indicates the dealing session. These values are both set to $\perp$ if k does not come from a H query of the form $(S,i,*)$ for $1 \leq i \leq n$. The e will be explained in **Experiment $\mathbf{Exp}_3$**. We simulate E and D oracles as follows Fig. 8.

The above simulation is perfect, as long as the following two bad events don't occur:

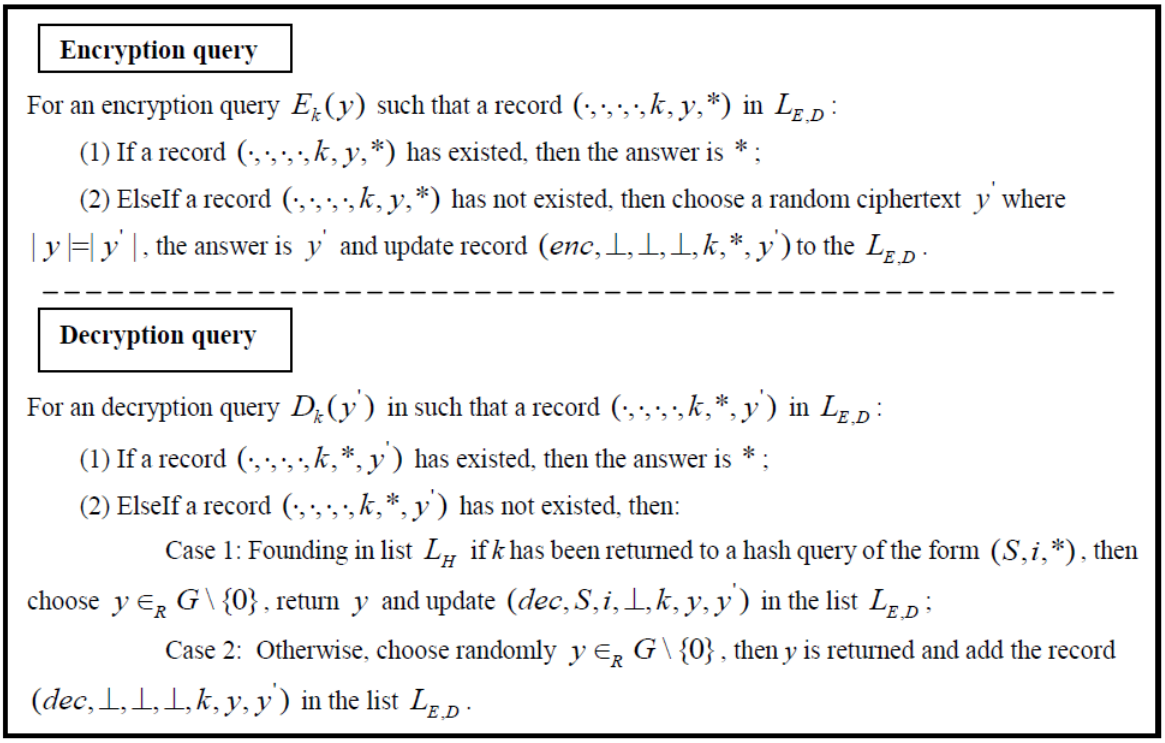

FIGURE 8. Oracles query for encryption/decryption

(1) Collisions may appear that contradict the permutation property of the ideal-cipher. Define this collision even by $\mathrm{Col}_{E,D}$, the probability of such bad even can be upper-bounded by $\frac{(q_E+q_D)^2}{2|G|}$.
(2) In the case of the decryption query simulation, one will abort executions if the value k involved in a decryption query is output later by H. Fortunately, this even denoted by Col_k happens with probability at most $q_H/2^{l_H}$ for each decryption query. Intuitively, as it will become clear in the next experiments, we indeed want to make sure that, for any k involved in a decryption query, if k comes from a H query, we know the corresponding tuple (S,i).

Above simulation is perfect unless the bad event $\mathrm{Bad}_2(=\mathrm{Col}_{E,D} \vee \mathrm{Col}_k)$does occur. Therefore, we get:

$$\Pr[\mathrm{Suc}_2|\neg\mathrm{Bad}_2] = \Pr[\mathrm{Suc}_1|\neg\mathrm{Bad}_2]$$

By **lemma 2**, we have:

$$|\Pr[\mathrm{Suc}_2] - \Pr[\mathrm{Suc}_1]| \leq \Pr[\mathrm{Bad}_2] \leq \frac{(q_E+q_D)^2}{2|G|} + \frac{q_D}{2|G|} + q_H q_D/2^{l_H}$$

Experiment Exp$_3$. This experiment change the simulation of the decryption queries, and make our challenger to embed an instance of the MDDH problem in the protocol simulation. Let the challenger output tuples $(\zeta_1,...,\zeta_n,\lambda_1,...,\lambda_{n-1})$ according to the Π_{real} distribution. We use these tuples to properly simulate the decryption queries. More precisely, we make 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 therandom $self-reducibility$[38]. The latter tells us that, given a

tuple outputted by the challenger, and then for any randomly chosen $(e_1, e_2, ..., e_n)$, the tuple $(\zeta_1^{e_1}, ..., \zeta_n^{e_n}, \lambda_1^{e_1 e_2}, ..., \lambda_{n-1}^{e_{n-1} e_n})$ has the same distribution as the original tuple. We make this property as follows, by modifying the first sub-case previously considered for new decryption queries in experiment $\mathbf{Exp}_2$ as follows Fig. 9.

Decryption query

For a new decryption query $D_k(y')$ such that $k = H(S, i, *)$ was previously obtained from H with valid index i, we query challenger for getting a tuple $(\zeta_1, ..., \zeta_n, \lambda_1, ..., \lambda_{n-1})$. Then we choose $e \in_R Z_q^*$ randomly, add the record $(dec, S, i, e, k, y = \zeta_i^e, y')$ in list $L_{E,D}$, and return y.

FIGURE 9. New decryption queries in experiment $\mathbf{Exp}_2$

Now, the list $L_{E,D}$ whose records are of the form $(S, i, e, k, y, \Lambda, y')$ has been defined. The element e is an exponent indicating how we applied the random $self-reducibility$ of the MDDH problem to the instance generated by the challenger upon the request $S : z = \zeta_i^e$. Here, e can only be defined if S and i are known. If e is unknown to the simulator, we set $e{=}\perp$. Above change of simulation on the decryption queries does not modify the view of the adversary, so we can get $\Pr[\text{Suc}_3] = \Pr[\text{Suc}_2]$.

Experiment $\mathbf{Exp}_4$. Starting from this experiment, we will simulate the Send queries in a different way, but only in the Step2 and Step3 in proposed protocol. When the session S is defined, user I computes the symmetric keys as $k_j = H(S, i, pw)$, for all j. Thus we know that we are working with the tuple $(\zeta_1, ..., \zeta_n, \lambda_1, ..., \lambda_{n-1})$.

In the Step2 of the round1, U_i chooses $y_i' \in_R G$ to be broadcasted, and queries $y_i = D_{k_i}(y_i')$ using the simulation in experiment $\mathbf{Exp}_3$. The simulation leads to add e_i to the list $L_{E,D}$, unless y_i' already existed as an encryption result. Defining the event that y_i' already existed as an encryption result by $\mathbf{Eve}_4$, but the probability $\Pr[\mathbf{Eve}_4]$ is at most $q_E/|G|$.

In the Step 2 of the Round 1, $U_i$computes $y_{i-1} = D_{k_{i-1}}(y_{i-1}')$ and $y_{i+1} = D_{k_{i+1}}(y_{i+1}')$. We omit the the ID and timestamp, then there are two cases appear:

Case 1: y_{i-1}' and y_{i+1}' have been simulated according to the above simulation of the Step2, and then one gets e_{i-1} and e_{i+1} in the list $L_{E,D}$ such that $y_{i-1} \equiv \zeta_{i-1}^{e_{i-1}} \bmod p$ and $y_{i+1} \equiv \zeta_{i+1}^{e_{i+1}} \bmod p$.

Case 2: one of the y_j' has been previously answered by the encryption oracle in response to an adversary query $E_k(y')$, where $k = H(S, j, pw)$ is the correct key for player U_j in session S. Now, we denote such an event by Enc_i in experiment $\text{Exp}_i (4 \leq i \leq 8)$. In this case, the simulation is terminated and let the adversary win.

Then user U_i can get $y_{i-1} \equiv \zeta_{i-1}^{e_{i-1}} \bmod p$ and $y_{i+1} \equiv \zeta_{i+1}^{e_{i+1}} \bmod p$, and correctly compute $SK_{i-1,i} = H(\cdot, \cdot, \cdot, \cdot, \lambda_{i-1}^{e_{i-1} e_i} \bmod p)$ and $SK_{i+1,i} = H(\cdot, \cdot, \cdot, \cdot, \lambda_{i+1}^{e_{i+1} e_i} \bmod p)$. Then U_i computes and broadcasts $X_i = H(SK_{i-1,i}, ID_{session}) \oplus H(SK_{i,i+1}, ID_{session})$. After this final step, each user can compute the session key as before. Because we have already simulated the form $X_i = H(\cdot) \oplus H(\cdot)$ which bad event is $q_H^4/(2^{l_H})^3$ in $\mathbf{Exp}_2$, we just only simulate $\lambda_{i-1}^{e_{i-1} e_i} \bmod p$ and $\lambda_{i+1}^{e_{i+1} e_i} \bmod p$.The simulation is still

perfect, unless the bad event Bad4(=$\text{Eve}_4 \vee \text{Enc}_4$) does occur. Therefore, we get $\Pr[\text{Suc}_4|\neg\text{Bad}_4] = \Pr[\text{Suc}_3|\neg\text{Bad}_4]$.

Based on lemma 2, we can defer:

$$\Pr[\text{Suc}_4] - \Pr[\text{Suc}_3] \leq \Pr[\text{Bad}_4]$$

$$\Pr[\text{Suc}_4] - \Pr[\text{Suc}_3] \leq q_E/|G| + \Pr[\text{Enc}_4]$$

Experiment Exp$_5$.Since it is clear that the security of the above protocol still relies on the DDH assumption, we let the challenger output tuples $(\zeta_1, ..., \zeta_n, \lambda_1, ..., \lambda_{n-1})$ according to the Π_{rand} distribution. By Lemma 1 and Lemma 2, we then can get:

$$\Pr[\text{Suc}_5] - \Pr[\text{Suc}_4]| \leq q_{session}\text{Adv}_G^{mddh}(t) \leq nq_{session}\text{Adv}_G^{ddh}(t)$$

$$\Pr[\text{Enc}_5] - \Pr[\text{Enc}_4]| \leq q_{session}\text{Adv}_G^{mddh}(t) \leq nq_{session}\text{Adv}_G^{ddh}(t)$$

Experiment Exp$_6$. For the authenticator, each participant $U_i(i = 1, ..., n)$ verifies if $X_1 \oplus X_2 \oplus X_3 \oplus ... \oplus X_{n-1} \oplus X_n = 0$ holds and all participants will continue to compute the group key. The probability for the adversary to distinguish the current and the previous experiments is to query $H\left(SK_{i,i+1}, ID_{session}\right)$, which is upper-bounded by $q_H/|G|$. And for The probability for the adversary to distinguish the current and the previous experiments is to query $X_i = H\left(SK_{i-1,i}, ID_{session}\right) \oplus H\left(SK_{i,i+1}, ID_{session}\right)$ which is upper-bounded by $q_H^2/2^{l_H}|G|^2$.So we have

$$\Pr[\text{Suc}_6] - \Pr[\text{Suc}_5]| \leq q_H^2/2^{l_H}|G|^2$$

$$\Pr[\text{Enc}_6] - \Pr[\text{Enc}_5]| \leq q_H^2/2^{l_H}|G|^2$$

Experiment Exp$_7$. Like the experiment **Experiment Exp$_6$**, after the modification of the computation of the session key, the probability for the adversary to distinguish the current and theprevious experiment is to query $H(B_1||B_2||...||B_n)$. From the previous experiment, we know that all honest users have the same view inside each session, thus these queries are same: the probability of such event is upper-bounded by $q_H/|G|$, since no information has been leaked about $B_i$except it does not correspond to the H queries which asked above. So we have

$$\Pr[\text{Suc}_7] - \Pr[\text{Suc}_6]| \leq q_H/(|G| - q_H) \leq 2q_H/|G|$$

$$\Pr[\text{Enc}_7] - \Pr[\text{Enc}_6]| \leq 2q_H/|G|$$

Experiment Exp$_8$. Note that the password $\boldsymbol{pw}$ is only used in the simulation of the Step1 and Step2 in round1, to compute $y_{i-1}^{'}$, $y_i^{'}$, $y_{i+1}^{'}$, but eventually, U_i computed using the $\lambda_{i-1}^{e_{i-1}e_i}$ and $\lambda_{i+1}^{e_{i+1}e_i}$ is outputted only. They are totally independent each other. Thus we can simplify the simulation of the Step1 and Step2: In the Step1, user U_i chooses $y_i^{'} \in G$ randomly, and sends it but no decryption is needed. In the Step2, U_i simply computes and sends X_i. The simulation is perfect, since one does not need to compute $SK_{i-1,i}$ and $SK_{i,i+1}$ anymore.

In the above simulation, the password $\boldsymbol{pw}$ is never used, and can thus be chosen at the end only, which makes clear that probability of the even Enc_8 is no more than q_{active}/N where q_{active} denotes the number of first flows manufactured by the adversary. Based on the fact that collisions in the output of H have been eliminated in previous experiments, we have

$$\Pr[\mathrm{Enc}_8] - \Pr[\mathrm{Enc}_7]| \leq q_{active}/N$$

Consequently from equations in all experiments, we get the **Theorem 1**.

$$\mathrm{Adv}_P^{aka}(t) \leq \frac{2q_{active}}{N} + 4nq_{session}\mathrm{Adv}_G^{ddh}(t) + 2q_H^4/(2^{l_H})^3 + 2q_H(2q_H + q_D)/2^{l_H} + \frac{4q_H^2}{2^{l_H}|G|^2} + \frac{q_D + 2q_E + 8q_H + (q_E + q_D)^2}{|G|}$$