Group Key Exchange over Combined Wired and Wireless Networks

Junghyun Nam and Dongho Won

Abstract: **A group key exchange protocol is a cryptographic primitive that describes how a group of parties communicating over a public network can come up with a common secret key. Due to its significance both in network security and cryptography, the design of secure and efficient group key exchange protocols has attracted many researchers' attention over the years. However, despite all the efforts undertaken, there seems to have been no previous systematic look at the growing problem of key exchange over combined wired and wireless networks which consist of both stationary computers with sufficient computational capabilities and mobile devices with relatively restricted computing resources. In this paper, we present the first group key exchange protocol that is specifically designed to be well suited for this rapidly expanding network environment. Our construction meets simplicity, efficiency, and strong notions of security.**

Index Terms: **Combined wired and wireless networks, decisional Diffie-Hellman (DDH) assumption, group key exchange, mobile devices.**

I. INTRODUCTION

The primary goal of cryptography is to provide a means for communicating confidentially and with integrity over a public channel. Roughly speaking, confidentiality means that the data transferred is not disclosed to unauthorized parties, and integrity means that the transferred data cannot be modified by an unauthorized party without being detected. It is well accepted that the most effective way to achieve the goal is by establishing a common secret key called session key and then using this key together with standard cryptographic algorithms for message encryption and authentication. Thus, the problem of establishing confidential and integrity-preserving communication is commonly reduced to the problem of designing a key exchange protocol that allows the parties communicating over a public network to establish a session key. Needless to say, a tremendous amount of research effort has been devoted to the design and analysis of key exchange protocols in a variety of settings (e.g., [2]–[6] and their follow-ups).

The first priority in designing a key exchange protocol is placed on ensuring the security of session keys to be established in the protocol. Even if it is computationally infeasible to break the cryptographic algorithms used, the whole system becomes vulnerable to all manner of attacks if the keys are not securely established. However, the experience has shown that the design of secure key exchange protocols is notoriously difficult; there is a long history of protocols for this domain being proposed and later found to be flawed (see [7] for a comprehensive list of examples). Thus, key exchange protocols must be subjected to a thorough and systematic scrutiny before they can be deployed into a public network, which might be controlled by an adversary. This concern has prompted active research on formal models of security for key exchange [8]–[17], and highlighted the importance of rigorous security proofs for protocols in a well-defined model. Although rigorously proving a protocol secure can often be a lengthy and complicated task, proofs are advocated as invaluable tools for obtaining a high level of assurance in the security of key exchange protocols [12], [14], [18]–[22].

Efficiency is another important consideration in designing key exchange protocols. In particular, it may become a critically practical issue for mobile applications where users may be resource constrained. Although mobile computing technology has become more powerful and accessible than ever before, there still remains a marked difference in computation resources between small mobile devices and general-purpose stationary computers. Mobile devices are typically characterized by low processing capability and limited power supply [23], which are inherent in the mobility nature. It is thus necessary that the cost due to security-related operations should be minimized for mobile devices in such a way that the required security goals are not compromised. This significantly adds to the challenge of securing communications in wireless networks. In fact despite all the work conducted over many decades, security is still a major limiting factor for the full adoption of mobile devices [22], [24]–[27].

In this paper, we are interested in key exchange in the group setting where a session key is to be established among a group of parties. Protocols for group key exchange are essential in building secure multicast channels for applications where quite a large number of users are likely to be involved (e.g., video conferencing and massive online gaming). As these group-oriented applications proliferate in modern computing environments, research on group key exchange has received much attention in recent years (a very partial list of recent work includes [12], [15], [19], [20], [28]–[44]). The efficiency of group key exchange protocols is measured with respect to computation overhead, as well as communication overhead. Computation overhead is mostly concerned with the number of public-key cryptographic operations that users have to perform, while communication overhead is usually quantified as both the number of rounds of communication among users and the number of messages sent/received by users (see [45] for some results on communication complexity of group key exchange). In particular for a group key exchange protocol to be scalable, it is of prime importance in many real-life applications that the protocol should be able to run in a constant number of communication rounds.

The goal of this work is to present a solution to the problem of group key exchange over combined wired and wireless

Manuscript received April 26, 2006.

The authors are with the School of Information and Communication Engineering, Sungkyunkwan University, Korea, email: {jhnam, dhwon}@security.re.kr.

A preliminary version of this work was presented in part at TrustBus 2005 [1].

networks, which consist of both low performance mobile devices with some form of battery power and high performance stationary computers with no power constraint. When one considers the broad range of wirelessly connected mobile devices used today, it is clear that integrating such network-enabled devices into secure group communication systems is timely and will be increasingly important. While a number of problems related to group key exchange have been tackled and solved over the past years, there seems to have been no previous systematic look at the growing problem of group key exchange over combined wired and wireless networks. Indeed, most previous group key exchange protocols are not well suited for network environments similar to our setting. Even though some constant-round protocols have been proposed [20], [36], [40], [41], [46], they are still too costly to be practical for applications involving mobile devices with limited computing resources. The reason for this is that these protocols are fully symmetric and therefore, as group size grows, the workload of every user also increases substantially, imposing an unfair, excessive burden on small mobile devices. On the other hand, other constant-round protocols [34],[1] [39], [42], [47] are extremely asymmetric and thus also are not efficiently applicable in our network setting; in these protocols one or two special users must perform $O(n)$ public-key cryptographic operations in a group of size n, being a significant performance bottleneck in a large group setting. It is these observations that prompted the present work aimed at designing a group key exchange protocol well suited for combined wired and wireless networks.

In this work, we concentrate on *contributory* key exchange protocols in which the session key is derived as a function of contributions provided by all parties. In contributory key exchange protocols, a correctly behaving party is assured that as long as his contribution is chosen at random, even a coalition of all other parties will not be able to have any means of controlling the final value of the session key. Thus, the use of contributory key exchange protocols is often recommended to prevent some parties having any kind of advantage over the others [28], [48]. Note that the well-known group key distribution protocols from [49]–[51] are not contributory. Furthermore, these non-contributory protocols, while they focus on minimizing the cost of the rekeying operations associated with group updates, lack at least one of the important security properties: Perfect forward secrecy [52] or known key security [53] (see below for descriptions of these properties).

Our contribution is to present the first constant-round group key exchange protocol that is well suited for the combined wired and wireless networks. The key feature of our approach is that it avoids any potential performance bottleneck of the system while keeping the burden on low power users at minimal, by evenly spreading most of workload across high power users. This feature enables to distinguish our protocol from all other constant-round protocols [20], [34], [36], [39]–[42], [46], [47] in that the maximum amount of computation to be done by any single user is bounded by $O(\sqrt{n})$ in our protocol whereas this amount per user rises up to $O(n)$ in the other protocols. Thus, if we define the computational complexity of a protocol as the maximum computation rate per user in the protocol,[2] our result may be regarded as the first constant-round protocol that has computational complexity lower than $O(n)$ (see Section II-D for more details on the efficiency of our protocol). In addition, the number of major cryptographic operations to be performed by each mobile user in our protocol remains low and constant without respect to the number of participants. This represents a key difference between our protocol and the well-known fully-symmetric protocols [20], [36], [40], [41], [46] where the computational load on each user scales linearly as the group size grows.

As stated already, proofs are invaluable for getting secure protocols. Our protocol is provably secure against a passive adversary under the decisional Diffie-Hellman (DDH) assumption (see Section III-C for definition of the DDH assumption). We provide a rigorous proof of security for the proposed protocol in the framework of a well-defined formal model. Our security proof of course captures important security notions of (1) perfect forward secrecy and (2) known key security. It is thus guaranteed in our protocol that: (1) Disclosure of long-term secret information does not compromise the security of previously established session keys and (2) exposure of some session keys does not affect the security of other session keys.

The remainder of this paper is organized as follows. In Section II, we present our construction of a group key exchange protocol well suited for combined wired and wireless networks. Next in Section III, we give some preliminaries required for the security proof of the proposed protocol, including a communication and adversary model with an associated definition of security. Then, in Section IV, we rigorously prove the security of the protocol against a passive adversary under the DDH assumption. Finally, Section V concludes this work.

II. THE PROPOSED PROTOCOL

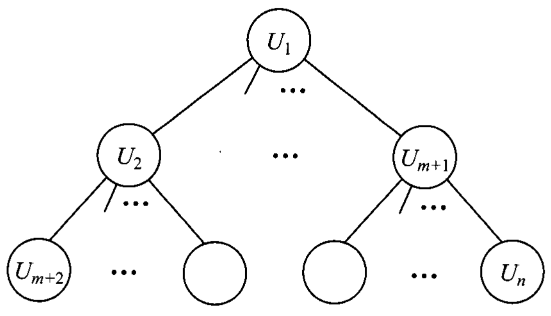
In this section, we introduce a group key exchange protocol P-CWWN appropriate for use in combined wired and wireless networks, where users with resource-constrained mobile devices and users with high-performance stationary computers coexist as potential protocol participants. The protocol P-CWWN consists of two subprotocols: The basic protocol BP and the generalized protocol GP. The basic protocol BP is designed for use in the extreme case where the number of high power users participating in P-CWWN is only one or two. The generalized protocol GP is intended for use in the more general and interesting case where there are more than two high power users. We first construct the basic protocol BP for the extreme case and then extend it to construct the generalized protocol GP. As we shall see later in Section IV, both of the protocols BP and GP are provably secure against a passive adversary under the DDH assumption.

A. Setup

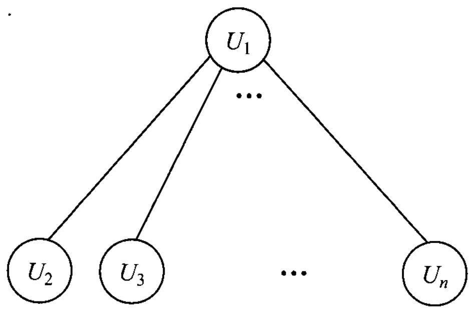
Let $\mathcal{G}$ be a nonempty set of n users who wish to establish a common session key by participating in the group key exchange protocol P-CWWN. We divide this set $\mathcal{G}$ of protocol participants into two disjoint subsets $\mathcal{S}$ and $\mathcal{R}$, i.e., $\mathcal{G} = \mathcal{S} \cup \mathcal{R}$, where $\mathcal{S} = \{U_1, \cdots, U_{n_s}\}$ is the nonempty set of users with

[1] We refer to [22] for a security enhancement to the protocol of [34].

[2] In fact, this definition is also used in the full version of [20] which is available at `http://eprint.iacr.org/2003/171`.



(a) $n_s > 2$



(b) $n_s = 1$ or $n_s = 2$

Fig. 1. $\mathcal{G} = \mathcal{S} \cup \mathcal{R}$, $\mathcal{S} = \{U_1, \cdots, U_{n_s}\}$, and $\mathcal{R} = \{U_{n_s+1}, \cdots, U_n\}$

stationary computers that have sufficient computational capabilities and $\mathcal{R} = \{U_{n_s+1}, \cdots, U_n\}$ is the set of users with mobile devices that have relatively restricted computing resources. As stated in the Introduction, the design goal of our protocol is to evenly spread most of workload across the high power users while keeping the computational burden on low power users at minimal. Towards the goal, we arrange the users in a tree structure with height 2 according to their computing power as depicted in Fig. 1(a). All users in $\mathcal{R}$ are at leaves in the tree while the users in $\mathcal{S}$ could be at any level in the hierarchy from 0 to 2. In our protocol, the amount of computation to be done by a user at an internal node scales linearly with the number of children of the user, while the users at leaf nodes perform only a constant amount of computation (except for multiplication). Accordingly, minimizing the maximum number of children of a user corresponds to minimizing the maximum amount of computation to be done by any single user, and therefore avoiding any potential performance bottleneck in the protocol.

The maximum number of children of a user depends on the number of users at level 1, which we denote by m. To see this, assume a fixed number n of users to be arranged in the tree of Fig. 1(a). If we increase the value of m, the number of U_1's children increases while the average number of children of each user at level 1 decreases. If instead we decrease the value of m, the average number of children per user at level 1 increases while the number of U_1's children decreases. So to minimize the maximum amount of computation to be performed by any single user, it is important to choose the value of m carefully.

Let n_r denote the cardinality of $\mathcal{R}$ (i.e., $n = n_s + n_r$). Given n_s and n_r, we can derive an optimal value of m as follows:

$$m = \begin{cases} 0, & \text{if } n_s = 1 \text{ or } n_s = 2 \\ n_s - 1, & \text{if } n_s > 2 \text{ and } n_r \geq (n_s - 1)(n_s - 2) \\ \lfloor \sqrt{n-1} \rfloor, & \text{otherwise.} \end{cases}$$

Fig. 1(b) shows one extreme case where $m = 0$ (i.e., $n_s = 1$ or $n_s = 2$) and thus, the users are organized into an $(n-1)$-ary tree with height 1. In this case, we place one user in $\mathcal{S}$ at the root node of the tree and all other users in $\mathcal{G}$ at leaf nodes. Consequently, much of the computational burden is shifted to the root user who has sufficient computing power. The case $n_s > 2$ is divided into two subcases, depending on whether n_r is no less than $(n_s - 1)(n_s - 2)$ or not. In the subcase $n_r \geq (n_s - 1)(n_s - 2)$, the value of m is set to the value of $n_s - 1$. Namely, all the $n_s - 1$ high power users are placed at level 1 if n_r is large enough so that at least $(n_s - 2)$ low power users can be assigned as children to each of $n_s - 1$ high power users. In the other subcase $n_r < (n_s - 1)(n_s - 2)$, the value of m is set to the value of $\lfloor \sqrt{n-1} \rfloor$, meaning that $n_s - 1 - \lfloor \sqrt{n-1} \rfloor$ high power users are placed at level 2 instead of at level 1.

Once the value of m is determined as above, the users $U_{m+2}, \cdots, U_n$ are placed at level 2 in an obvious way that an equal number of users (more precisely, $\lceil \frac{n-m-1}{m} \rceil$ or $\lfloor \frac{n-m-1}{m} \rfloor$ users) are assigned as children to each of the users at level 1.

In describing the protocol, we assume that the following public information has been fixed in advance and is known to all parties in the network: (1) The structure of the tree and the users' positions within the tree and (2) a finite cyclic group $\mathbb{G}$ of prime order q, where the DDH assumption holds, and a generator g of $\mathbb{G}$. A standard way of generating $\mathbb{G}$ where the DDH assumption is assumed to hold is to choose two primes p and q such that $p = kq + 1$ for some small $k \in \mathbb{N}$ (e.g., $k = 2$) and let $\mathbb{G}$ be the subgroup of order q in $\mathbb{Z}_p^*$. In order to simplify the description of the protocol, we omit 'mod p' from expressions and divide the set $\mathcal{G}$ into three disjoint subsets $\mathcal{L}_0$, $\mathcal{L}_1$, and $\mathcal{L}_2$ which denote the sets of users at levels 0, 1, and 2, respectively.

In this work, we focus on security against passive adversaries and assume all messages are digitally signed by their source in a way that the protocols effectively resist active attacks. To achieve security against active adversaries, one may use the recent result of Katz and Yung [20] where they presented an efficient one-round compiler that transforms any group key exchange protocol secure against a passive adversary into one that is secure against an active adversary. Indeed, by Theorems 1 and 2 in Section IV, both the subprotocols BP and GP can be easily converted into group key exchange protocols secure against an active adversary if we apply the Katz and Yung's compiler to the protocols.

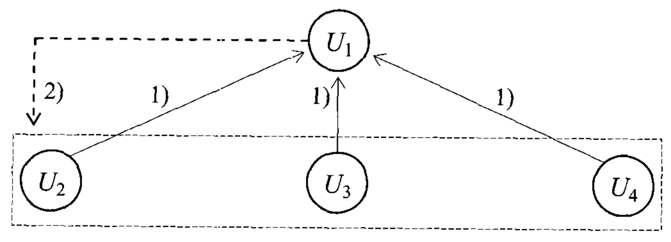
B. The Basic Protocol BP

The basic protocol BP is a group key exchange protocol designed to be well suited for the case $0 < n_s \leq 2$. Fig. 2 shows a simple illustrative example of an execution of protocol BP. A more detailed description of the protocol, on input three sets $\mathcal{L}_0 = \{U_1\}$, $\mathcal{L}_1 = \{U_2, \cdots, U_n\}$, and $\mathcal{L}_2 = \emptyset$, is as follows:

Round 1: Each user $U_i \in \mathcal{L}_1$ chooses a random $r_i \in \mathbb{Z}_q$, computes $z_i = g^{r_i}$, and sends z_i to its parent U_1.

Round 2: User U_1 first computes the group secret X by performing the following operations:

1. U_1 chooses two random numbers $s, r_1 \in \mathbb{Z}_q$ and computes $x_1 = g^{sr_1}$.
2. U_1 computes $x_i = z_i^s$ upon arrival of each z_i.



Random selections: U_1: $s, r_1 \in \mathbb{Z}_q$; $U_i \in \mathcal{L}_1$: $r_i \in \mathbb{Z}_q$.

1) Round 1: $g^{r_2}, g^{r_3}, g^{r_4}$.
Keying materials: $y_2 = g^{s(r_1+r_3+r_4)}$, $y_3 = g^{s(r_1+r_2+r_4)}$, $y_4 = g^{s(r_1+r_2+r_3)}$.

2) Round 2: $g^s \| \{y_2, y_3, y_4\}$.
Group secret: $X = g^{s(r_1+r_2+r_3+r_4)}$.
Session key: $K = X \cdot y_2 \cdot y_3 \cdot y_4$.

Fig. 2. An execution of the basic protocol BP with $\mathcal{G} = \mathcal{L}_0 \cup \mathcal{L}_1$, where $\mathcal{L}_0 = \{U_1\}$ and $\mathcal{L}_1 = \{U_2, U_3, U_4\}$.

3. Using all the x_i's, the group secret X is computed as

$$X = \prod_{i \in [1,n]} x_i.$$

Then U_1 generates the public keying material $w = g^s$ and $\mathcal{Y} = \{y_i \mid i \in [2,n]\}$, where $y_i = X \cdot x_i^{-1}$, and sends the message $w \| \mathcal{Y}$ to its children.

Key computation: Upon receiving $w \| \mathcal{Y}$, each user $U_i \in \mathcal{L}_1$ computes the group secret $X = y_i \cdot w^{r_i}$. Finally, all users in $\mathcal{G}$ compute their session key as

$$K = X \prod_{i \in [2,n]} y_i.$$

It is straightforward to verify that all users compute the same secret value $X = g^{s(r_1+r_2+\cdots+r_n)}$ and hence the same session key K.

CONTRIBUTORY PROPERTY. Being *contributory* is a desirable property for group key exchange protocols to possess.

Definition 1: A key exchange protocol is said to be contributory if and only if it satisfies the following two conditions:

1. The session key is derived as a function of contributions provided by all participants.
2. A correctly behaving participant is assured that as long as his contribution is chosen at random, even a coalition of all other participants will not be able to have any means of controlling the final value of the session key.

The protocol BP is, by Definition 1, contributory. Clearly, the first condition is satisfied. It is not difficult to see that the second condition is also satisfied. Although the value of the common secret X can be predetermined by user U_1, the values of X and y_i can never be controlled simultaneously. To see this, note that as soon as X is set to some specific value, the value of y_i is determined inevitably by the equation $y_i = X \cdot x_i^{-1}$. That is, given a fixed value of X, the value of y_i jumps around in $\mathbb{G}$ depending on x_i. Since all the y_i's are multiplied with X to derive K, each user can be assured that the distribution of their session key cannot be biased by any other users.

C. The Generalized Protocol GP

This subsection presents our main construction which uses the above-described protocol BP as a basic building block. The idea in constructing GP is to distribute the users into m $(= |\mathcal{L}_1|)$ subgroups and to run the basic protocol BP for each subgroup. Each parent $U_i \in \mathcal{L}_1$ forms a subgroup $\mathcal{G}_i$ with its children and plays a central role in the subgroup. First, each $\mathcal{G}_i$ generates a secret subgroup key k_i by running the basic protocol BP. After having derived k_i, each subgroup $\mathcal{G}_i$ participates again in the protocol BP as a single entity to generate the group key (i.e., the session key). In this final run of protocol BP, the subgroup key $k_i \in \mathbb{G}$ is used as the random exponent $r_i \in \mathbb{Z}_q$ for the subgroup $\mathcal{G}_i$. So we need a function I mapping elements of $\mathbb{G}$ to elements of $\mathbb{Z}_q$. For our security proof, we have to assume that $I : \mathbb{G} \rightarrow \mathbb{Z}_q$ is a bijection. Whether there are appropriate bijections from $\mathbb{G}$ into $\mathbb{Z}_q$ depends on the group $\mathbb{G}$. If p is a safe prime (i.e., $p = 2q + 1$), then a bijection I can be constructed as follows:

$$I(x) = \begin{cases} x & \text{if } x \le q \\ p - x & \text{if } q < x < p. \end{cases}$$

Fig. 3 shows an example of the protocol execution with $\mathcal{L}_0 = \{U_1\}$, $\mathcal{L}_1 = \{U_2, U_3, U_4\}$, and $\mathcal{L}_2 = \{U_5, \cdots, U_{12}\}$. The following describes the protocol for the general case where $\mathcal{L}_0 = \{U_1\}$, $\mathcal{L}_1 = \{U_2, \cdots, U_{m+1}\}$, and $\mathcal{L}_2 = \{U_{m+2}, \cdots, U_n\}$.

Round 1: Each user $U_j \in \mathcal{L}_2$ chooses a random $\tilde{r}_j \in \mathbb{Z}_q$, computes $\tilde{z}_j = g^{\tilde{r}_j}$, and sends $\tilde{z}_j$ to its parent in $\mathcal{L}_1$.

Round 2: Each user $U_i \in \mathcal{L}_1$ first computes the subgroup key k_i to be shared with its children as follows:

1. U_i chooses two random numbers $\tilde{s}_i, \tilde{r}_i \in \mathbb{Z}_q$ and computes $\tilde{x}_i = g^{\tilde{s}_i \tilde{r}_i}$.
2. U_i computes $\tilde{x}_j = \tilde{z}_j^{\tilde{s}_i}$ upon receiving each $\tilde{z}_j$ for $j \in \mathcal{C}_i$, where $\mathcal{C}_i$ is the set of indices of the children of U_i.
3. U_i calculates the subgroup secret $\tilde{X}_i$ as

$$\tilde{X}_i = \prod_{j \in \mathcal{C}_i \cup \{i\}} \tilde{x}_j$$

and the subgroup keying material $\tilde{w}_i = g^{\tilde{s}_i}$ and $\tilde{\mathcal{Y}}_i = \{\tilde{y}_j \mid j \in \mathcal{C}_i\}$, where $\tilde{y}_j = \tilde{X}_i \cdot \tilde{x}_j^{-1}$.
4. The subgroup key k_i is then computed as

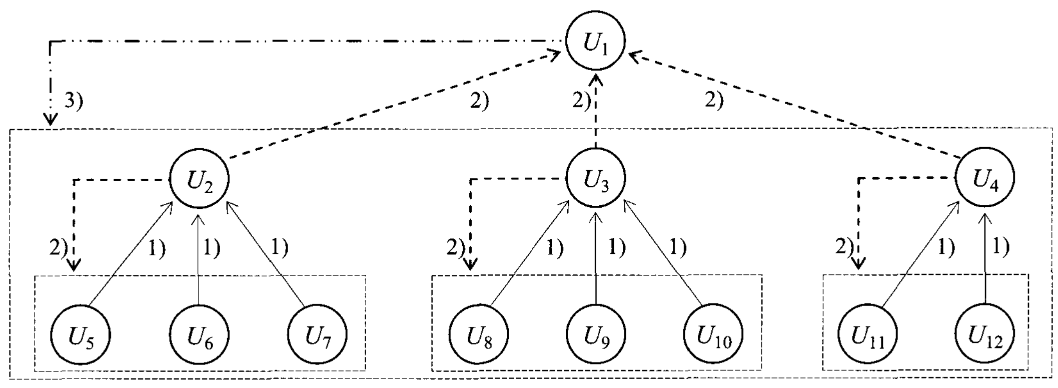
$$k_i = \tilde{X}_i \prod_{j \in \mathcal{C}_i} \tilde{y}_j.$$

Next, user $U_i \in \mathcal{L}_1$ computes $r_i = I(k_i)$ and $z_i = g^{r_i}$, and sends the message $z_i \| \tilde{w}_i \| \tilde{\mathcal{Y}}_i$ to its children and to the parent U_1.

Round 3: User $U_1 \in \mathcal{L}_0$ first computes the group secret X as follows:

1. U_1 chooses two random numbers $s, r_1 \in \mathbb{Z}_q$ and computes $x_1 = g^{sr_1}$.
2. U_1 computes $x_i = z_i^s$ for all $i \in [2, m+1]$.
3. U_1 calculates the group secret as

$$X = \prod_{i \in [1,m+1]} x_i.$$



Random selections: U_1: $s, r_1 \in \mathbb{Z}_q$; $U_i \in \mathcal{L}_1$: $\tilde{s}_i, \tilde{r}_i \in \mathbb{Z}_q$; $U_j \in \mathcal{L}_2$: $\tilde{r}_j \in \mathbb{Z}_q$.

1) Round 1: $g^{\tilde{r}_5}, g^{\tilde{r}_6}, g^{\tilde{r}_7}, g^{\tilde{r}_8}, g^{\tilde{r}_9}, g^{\tilde{r}_{10}}, g^{\tilde{r}_{11}}, g^{\tilde{r}_{12}}$.
   Subgroup keying materials: $\tilde{y}_5 = g^{\tilde{s}_2(\tilde{r}_2+\tilde{r}_6+\tilde{r}_7)}$, $\tilde{y}_6 = g^{\tilde{s}_2(\tilde{r}_2+\tilde{r}_5+\tilde{r}_7)}$, $\tilde{y}_7 = g^{\tilde{s}_2(\tilde{r}_2+\tilde{r}_5+\tilde{r}_6)}$, $\tilde{y}_8 = g^{\tilde{s}_3(\tilde{r}_3+\tilde{r}_9+\tilde{r}_{10})}$, $\tilde{y}_9 = g^{\tilde{s}_3(\tilde{r}_3+\tilde{r}_8+\tilde{r}_{10})}$, $\tilde{y}_{10} = g^{\tilde{s}_3(\tilde{r}_3+\tilde{r}_8+\tilde{r}_9)}$, $\tilde{y}_{11} = g^{\tilde{s}_4(\tilde{r}_4+\tilde{r}_{12})}$, $\tilde{y}_{12} = g^{\tilde{s}_4(\tilde{r}_4+\tilde{r}_{11})}$.
   Subgroup secrets: $\tilde{X}_2 = g^{\tilde{s}_2(\tilde{r}_2+\tilde{r}_5+\tilde{r}_6+\tilde{r}_7)}$, $\tilde{X}_3 = g^{\tilde{s}_3(\tilde{r}_3+\tilde{r}_8+\tilde{r}_9+\tilde{r}_{10})}$, $\tilde{X}_4 = g^{\tilde{s}_4(\tilde{r}_4+\tilde{r}_{11}+\tilde{r}_{12})}$.
   Subgroup keys: $k_2 = \tilde{X}_2 \cdot \tilde{y}_5 \cdot \tilde{y}_6 \cdot \tilde{y}_7$, $k_3 = \tilde{X}_3 \cdot \tilde{y}_8 \cdot \tilde{y}_9 \cdot \tilde{y}_{10}$, $k_4 = \tilde{X}_4 \cdot \tilde{y}_{11} \cdot \tilde{y}_{12}$.
   Subgroup random exponents: $r_2 = I(k_2)$, $r_3 = I(k_3)$, $r_4 = I(k_4)$.

2) Round 2: $g^{r_2} \| g^{\tilde{s}_2} \| \{\tilde{y}_5, \tilde{y}_6, \tilde{y}_7\}$, $g^{r_3} \| g^{\tilde{s}_3} \| \{\tilde{y}_8, \tilde{y}_9, \tilde{y}_{10}\}$, $g^{r_4} \| g^{\tilde{s}_4} \| \{\tilde{y}_{11}, \tilde{y}_{12}\}$.
   Group keying materials: $y_2 = g^{s(r_1+r_3+r_4)}$, $y_3 = g^{s(r_1+r_2+r_4)}$, $y_4 = g^{s(r_1+r_2+r_3)}$.

3) Round 3: $g^s \| \{y_2, y_3, y_4\}$.
   Group secret: $X = g^{s(r_1+r_2+r_3+r_4)}$
   Session key: $K = X \cdot y_2 \cdot y_3 \cdot y_4$.

Fig. 3. An execution of the generalized protocol GP with $\mathcal{G} = \mathcal{L}_0 \cup \mathcal{L}_1 \cup \mathcal{L}_2$, where $\mathcal{L}_0 = \{U_1\}$, $\mathcal{L}_1 = \{U_2, U_3, U_4\}$, and $\mathcal{L}_2 = \{U_5, \cdots, U_{12}\}$.

Then, U_1 computes $w = g^s$ and $\mathcal{Y} = \{y_i \mid i \in [2, m+1]\}$, where $y_i = X \cdot x_i^{-1}$, and sends the message $w \| \mathcal{Y}$ to the rest of the group.

Key computation: Now for all $i \in [2, m+1]$ and all $j \in \mathcal{C}_i$, user U_j is able to generate the group secret X by computing, in sequence, the subgroup secret $\tilde{X}_i = \tilde{y}_j \cdot \tilde{w}_i^{\tilde{r}_j}$, the subgroup key $k_i = \tilde{X}_i \prod_{j \in \mathcal{C}_i} \tilde{y}_j$, the subgroup random exponent $r_i = I(k_i)$, and then the group secret $X = y_i \cdot w^{r_i}$. Each U_i in $\mathcal{L}_1$ can directly calculate the group secret $X = y_i \cdot w^{r_i}$ using r_i obtained already in Round 2. Finally, all users in $\mathcal{G}$ compute the session key K as

$$K = X \prod_{i \in [2, m+1]} y_i.$$

The protocol GP given above is contributory for the same reason as discussed before for the protocol BP.

D. Efficiency

To the best of our knowledge, the protocol of Burmester and Desmedt [46] (often called the BD protocol) is the most efficient one among forward-secure group key exchange protocols that possess a rigorous proof of security in the standard model, i.e., without the random oracle assumption [54]. (The security of the BD protocol was proved under the DDH assumption in the relatively recent work of Katz and Yung [20].) In Table 1, we compare the efficiency between our protocols and the BD protocol, assuming use of the one-round compiler of Katz and Yung [20] that transforms unauthenticated group key exchange protocols into authenticated ones. As for computational costs, the table lists the amount of computation per user.

Our group key exchange protocols are very efficient in terms of the computational load on mobile devices. Each mobile device in protocol GP performs only 3 modular exponentiations, 1 signature generation, and 2 signature verifications. As shown in Table 1, the protocol BP imposes even less computational burden on mobile devices. If precomputations are possible, all the exponentiations in the first round of the protocols (i.e., the computations of z_i's in BP and $\tilde{z}_j$'s in GP) can be performed off-line and thus, the number of exponentiations to be done on-line by a mobile device can be reduced to one or two. Furthermore, the protocol GP avoids any potential performance bottleneck by distributing the computational load equally amongst high power users. If the number of high power users is at least $\sqrt{n_r}$, the maximum computation rate per user in protocol GP is bounded by $O(\sqrt{n})$. This distinguishes the protocol GP from the protocol BP and other extremely asymmetric protocols [34], [39], [47] where one or two special users must perform $O(n)$ public-key cryptographic operations.

In the BD protocol, all users behave in a completely symmetric manner, sending one message per round and performing the same amount of computation. While this protocol is very efficient in general, the full symmetry negatively impacts on the overall performance of the protocol involving mobile devices. The number of messages received by each mobile device

Table 1. Complexity comparison.

	Communication		Computation	
	Rounds	Messages	Low power users	High power users
BD^+	3	$3n$	3E + 2S + $O(n)$V	3E + 2S + $O(n)$V
BP^+	3	$2n$	2E + 1S + 1V	$O(n)$E + 1S + $O(n)$V
GP^+	4	$2n$	3E + 1S + 2V	$O(\sqrt{n})$E + 1S + $O(\sqrt{n})$V

E: Exponentiations, S: Signatures, V: Verifications

BD^+, BP^+, GP^+: The authenticated protocols obtained by applying the compiler of Katz and Yung [20] to the unauthenticated protocols BD, BP, and GP, respectively.

is $O(n)$ compared to $O(1)$ in our protocols. This implies that in the BD protocol, all users including mobile users have to perform $O(n)$ signature verifications. Indeed, $O(n)$ signature verifications per user are commonly required in all other symmetric protocols [36], [40], [41].[3]

The discussion above can be summarized as follows. In situations where users with equal computational capabilities communicate over a broadcast network, the fully-symmetric protocol of Burmester and Desmedt might be more favorable than our construction which, in contrast, is well suited for combined wired and wireless networks which consist of both stationary computers with sufficient computational capabilities and mobile devices with relatively restricted computing resources.

III. SECURITY PRELIMINARIES

Any form of security analysis of a cryptographic construction should be preceded by clear definitions of adversary capabilities and security goals. In this section we provide such a preliminary formulation for a concrete security analysis of our group key exchange protocols.

A. Communication and Adversary Model

A recent paradigm for security analysis of group key exchange protocols is the use of a concrete and realistic model which formalizes protocol executions in the presence of an adversary who has various attacking capabilities (e.g., [12], [19], [20], [32]–[34], [40]–[42]). In general, each capability of the adversary is modeled via an oracle to which the adversary is allowed to make queries. Since we focus on security against a passive adversary, there is no Send oracle in our adversary model. In particular, our model follows the so-called ROR model of [14] in that the adversary is allowed to query the Test oracle as many times as it wants. A detailed discussion on this is deferred to later in this section.

A.1 Participants

Let $\mathcal{U}$ be a set of all users who are potentially interested in participating in a group key exchange protocol. The users in any subset of $\mathcal{U}$ may run the group key exchange protocol at any point in time to establish a session key. Each user may run the protocol multiple times either serially or concurrently, with possibly different groups of participants. Thus each user can have many instances at a given time. We use Π_U^i to denote *instance* i of user U.

[3]Although there are some other symmetric protocols proposed [43], [44], we exclude these protocols from consideration simply because it is unfair to compare the efficiency between signature-based authenticated protocols and password-based authenticated protocols.

A.2 Partners

Intuitively, the *partners* of an instance is the set of all instances that should compute the same session key as the instance in an execution of the protocol. Like most of previous work, we use the notion of *session IDs* to define partnership between instances. Literally, a session ID (denoted as sid) is a unique identifier of a communication session. Following [11], [13], [15], [55], we assume that session IDs are assigned and provided by some higher-level protocol. While this assumption is unnecessary in some protocols [20], [34] which use only broadcast messages (in these protocols, a session ID can readily be defined as the concatenation of all message flows), it seems very useful in other cases where messages sent and received in an execution of the protocol may be different among protocol participants. Indeed, as pointed out in [11], the use of session IDs is typical in common security protocols such as SSL [56] and IPSec [57]. We let $\mathcal{SID}$ be the algorithm used by the higher-level protocol to generate session IDs, and assume that $\mathcal{SID}$ is publicly available.

We also need the notion of *group IDs* to define partnership properly. A group ID (denoted as gid) is a set consisting of the identities of the users who intend to establish a session key among themselves. This notion is clearly natural because it is impossible (not even defined) to ever execute a group key exchange protocol without participants. Indeed, a group ID is a both necessary and important input to any protocol execution.

In order for an instance to start to run the protocol, we require that a pair of sid and gid should be given as input to the instance. We use sid_U^i and gid_U^i to denote respectively sid and gid provided to instance Π_U^i. Note that gid_U^i should always include U itself. Session IDs and group IDs are public and hence available to the adversary. Indeed, the adversary in our model generates these IDs on its own; it generates a session ID by running $\mathcal{SID}$ and a group ID by choosing a subset of $\mathcal{U}$.

An instance is said to *accept* when it successfully computes a session key in a protocol execution. Let acc_U^i be a boolean variable that evaluates to TRUE if Π_U^i has accepted, and FALSE otherwise. We say that any two instances Π_U^i and $\Pi_{U'}^j$, where $U \neq U'$, are *partners* of each other, or equivalently, *partnered* iff the following conditions are met: (1) $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j$; (2)

$\mathsf{gid}_U^i = \mathsf{gid}_{U'}^j$; and (3) $\mathsf{acc}_U^i = \mathsf{acc}_{U'}^j = \text{TRUE}$. We also say that two instances Π_U^i and $\Pi_{U'}^j$ ($U \neq U'$) are *potential partners* of each other, or equivalently, *potentially partnered* iff the first two conditions above hold. We use pid_U^i and ppid_U^i to denote respectively the partners and the potential partners of the instance Π_U^i. Then it follows by the definitions that $\mathsf{pid}_U^i \subseteq \mathsf{ppid}_U^i$.

A.3 Adversary

It is not very persuasive to claim the protocol's security without clarifying what kind of adversary we are dealing with. In a formal model for key exchange, the attacking ability of an adversary is modeled via various oracles to which the adversary is allowed to make queries. But unlike previous models for group key exchange, we allow the adversary to query the Test oracle as many times as it wants.[4] This approach was recently suggested by Abdalla et al. [14] for password authenticated key exchange in the three-party setting and was also proved there to lead to a stronger model (for more details, see Lemmas 1 and 2 in Appendix B of [14]). What we found interesting is that allowing multiple Test queries is very useful in proving Theorem 2 which asserts the security of our main protocol GP. Other oracles (Execute, Reveal, and Corrupt) are as usual. A more detailed description of each of these oracles follows:

- Execute(sid, gid): This query prompts an honest execution of the protocol between a set of instances consisting of one instance for each user in gid, where the instances are all given the session ID sid and the group ID gid as their input. The transcript of the honest execution is returned to the adversary as the output of the query. This models passive attacks on the protocol.
- Reveal(Π_U^i)[5]: This query returns the session key held by instance Π_U^i to the adversary. This captures the idea that loss of some session keys should not lead to the compromise of other session keys [53]. This query can be asked only if $\mathsf{acc}_U^i = \text{TRUE}$ and the adversary has not queried Test($\Pi_{U'}^j$) for some $\Pi_{U'}^j$ in pid_U^i (see below for the description of the Test oracle and see Section III-B for more details on this matter).
- Corrupt(U): This query returns to the adversary all of U's long-term secrets. This models the adversary's capability of breaking into a user's machine and gaining access to the long-term data set stored there. The adversary can issue this query at any time regardless of whether U is currently executing the protocol or not. This oracle call captures the idea that damage due to loss of U's long-term secrets should be restricted to those sessions where U will participate in the future.
- Test(Π_U^i): This query provides a means of defining security (see Section III-B). The output of this query depends on the hidden bit b that the Test oracle chooses uniformly at random from $\{0, 1\}$ during its initialization phase. The Test oracle returns the real session key held by Π_U^i if $b = 1$, or returns a random session key drawn from the key space if $b = 0$. The adversary is allowed to query the Test oracle as many times as necessary. But, the query can be asked only for *fresh* (see Definition 2 given below) instances. All the queries to the oracle are answered using the same value of the hidden bit b that was chosen at the beginning. Namely, the keys returned by the Test oracle are either all real or all random.

[4]In all of previous models for group key exchange (except the very recent one in [43]), the adversary is restricted to ask only a single query to the Test oracle.

[5]While the Reveal oracle does not exist in the ROR model of Abdalla *et al.* [14], it is still available to the adversary in our model and is used in proving Lemma 1, enabling a modular approach in complicated security proofs for our protocols.

Definition 2: The instance Π_U^i is considered *unfresh* iff any of the following conditions hold:

1. $\mathsf{acc}_U^i = \text{FALSE}$.
2. The adversary queried Corrupt(U') for some U' in gid_U^i before some Π_V^j in ppid_U^i accepts.
3. The adversary queried Reveal($\Pi_{U'}^j$) for some $\Pi_{U'}^j$ in pid_U^i.
4. The adversary queried Test($\Pi_{U'}^j$) for some $\Pi_{U'}^j$ in pid_U^i.

All other instances are considered *fresh*.

Remark 1: By the fourth condition of Definition 2, we require that for each different set of partners, the adversary should access the Test oracle only once. One may think that this restriction weakens the ability of the adversary. However, this is not the case because when all information on partnering is public, obtaining the same data multiple times (from the instances partnered together) is no different than obtaining it once.

We represent the amount of queries used by an adversary as an ordered sequence of four non-negative integers, $Q = (q_{\text{exec}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{test}})$, where the four elements refer to the numbers of queries that the adversary made, respectively, to its Execute, Reveal, Corrupt, and Test oracles. We call this usage of queries by an adversary the *query complexity* of the adversary.

B. Security Definition

We now proceed to define what is meant for a group key exchange protocol to be secure. The security of a group key exchange protocol P against an adversary $\mathcal{A}$ is defined in terms of $\mathcal{A}$'s probability of succeeding in distinguishing real session keys from random session keys. That is, the adversary $\mathcal{A}$ is considered successful in attacking the protocol P if it breaks the semantic security of session keys generated by P. This notion of security is defined in the context of the following two-stage game, where the goal of adversary $\mathcal{A}$ is to correctly guess the value of the hidden bit b used by the Test oracle.

- **Stage 1:** $\mathcal{A}$ makes the oracle queries at will as many times as it wants.
- **Stage 2:** Once $\mathcal{A}$ decides that Stage 1 is over, it outputs a bit b' as a guess for the value of the hidden bit b used by the Test oracle. $\mathcal{A}$ wins the game if $b = b'$.

In the game above, the adversary can keep querying the oracles even after it asked some Test queries. However, when there was the query Test(Π_U^i) asked, the adversary is prohibited from querying Reveal($\Pi_{U'}^j$) for some $\Pi_{U'}^j \in \mathsf{pid}_U^i$. This restriction reflects the fact that the adversary can win the game unfairly by using the session key obtained via the query Reveal($\Pi_{U'}^j$).

Given the game above, the advantage of $\mathcal{A}$ in attacking protocol P is defined as

$$\mathsf{Adv}_P(\mathcal{A}) = \left|2 \cdot \Pr[b = b'] - 1\right|.$$

Note that this definition is equivalent to say that the advantage of $\mathcal{A}$ is the difference between the probabilities that $\mathcal{A}$ outputs 1 in the following two experiments constituting the game: The *real experiment* where all queries to the Test oracle are answered with the real session key, and the *random experiment* where all Test queries are answered with a random session key. Thus, if we denote the real and the random experiments respectively as $\mathsf{Exp}_P^{\mathsf{real}}(\mathcal{A})$ and $\mathsf{Exp}_P^{\mathsf{rand}}(\mathcal{A})$, the advantage of $\mathcal{A}$ can be equivalently defined as

$$\mathsf{Adv}_P(\mathcal{A}) = \left|\Pr[\mathsf{Exp}_P^{\mathsf{real}}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}_P^{\mathsf{rand}}(\mathcal{A}) = 1]\right|$$

where the outcomes of the experiments is the bit output by $\mathcal{A}$.

We say that the group key exchange protocol P is *secure* if $\mathsf{Adv}_P(\mathcal{A})$ is negligible for all probabilistic polynomial time adversaries $\mathcal{A}$. To quantify the security of protocol P in terms of the amount of resources expended by adversaries, we use the notation $\mathsf{Adv}_P(t, Q)$ which is defined as

$$\mathsf{Adv}_P(t, Q) = \max_{\mathcal{A}}\{\mathsf{Adv}_P(\mathcal{A})\}$$

where the maximum is over all adversaries $\mathcal{A}$ with time complexity at most t and query complexity at most Q.

C. Decisional Diffie-Hellman (DDH) Assumption

Let $\mathbb{G}$ be a cyclic (multiplicative) group of prime order q. Since the order of $\mathbb{G}$ is prime, all the elements of $\mathbb{G}$, except 1, are generators of $\mathbb{G}$. Let g be a random fixed generator of $\mathbb{G}$ and let x, y, and z be randomly chosen elements in $\mathbb{Z}_q^*$ where $z \neq xy$. Informally stated, the DDH problem for $\mathbb{G}$ is to distinguish between the distributions of (g^x, g^y, g^{xy}) and (g^x, g^y, g^z), and the DDH assumption is said to hold in $\mathbb{G}$ if it is computationally infeasible to solve the DDH problem for $\mathbb{G}$. To be more formal, we define the advantage of $\mathcal{D}$ in solving the DDH problem for $\mathbb{G}$ as

$$\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{D}) =$$
$$\left|\Pr[\mathcal{D}(\mathbb{G}, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(\mathbb{G}, g, g^x, g^y, g^z) = 1]\right|.$$

We say that the DDH assumption holds in $\mathbb{G}$ (or equivalently, the DDH problem is hard in $\mathbb{G}$) if $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{D})$ is negligible for all probabilistic polynomial time algorithms $\mathcal{D}$. We denote by $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)$ the maximum value of $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{D})$ over all $\mathcal{D}$ running in time at most t.

IV. SECURITY PROOFS

In this section, we prove that the group key exchange protocol P-CWWN is secure against a passive adversary in the formal model given in the preceding section. As we have seen, the protocol P-CWWN consists of two subprotocols: The basic protocol BP for the case $0 < n_s \leq 2$ and the generalized protocol GP for the case $n_s > 2$, where n_s is the number of high power users. We first provide the proof of security for the basic protocol BP under the DDH assumption and then reduce the security of the generalized protocol GP to the security of protocol BP. Corollaries 1 and 2 together constitute our main security result, giving concrete security bounds for the protocols BP and GP, respectively.

Due to the length and complexity of our proof, we first describe the top level structure of the proof procedure. Towards the goal of proving the corollaries, we start with Lemma 1 which states that any group key exchange protocol secure against a passive adversary who makes only a *single* query to the Test oracle is also secure against a passive adversary who makes *multiple* Test queries. Proving this lemma allows us to limit our security concern only to those cases where adversaries access their Test oracle only once. In Section IV-A, we continue by proving Theorem 1 which plays a key role in deriving Corollary 1. The theorem says that under the DDH assumption, the basic protocol BP is secure against a passive adversary who asks only one query to its Test oracle. Combining Theorem 1 with Lemma 1, we immediately obtain Corollary 1. In Section IV-B, we turn to proving Corollary 2 which, as mentioned above, states the security result for our main protocol GP. Corollary 2 is proved analogously to Corollary 1. Given Lemma 1, Corollary 2 directly follows from Theorem 2, the equivalent of Theorem 1 for the protocol GP. Thus, our final task is to prove Theorem 2, in which we assert that as long as the basic protocol BP is secure against a passive adversary asking multiple Test queries, the generalized protocol GP is secure against a passive adversary who asks only one Test query.

Reminding that $\mathsf{Adv}_P(t, Q)$ denotes the maximum value of $\mathsf{Adv}_P(\mathcal{A})$ over all $\mathcal{A}$ with time complexity at most t and query complexity at most Q, we now proceed to prove the security of the protocols. We begin by proving the following lemma, which states that in attacking any group key exchange protocol, the maximum advantage obtainable by a passive adversary asking q_{test} Test queries is at most q_{test} times the maximum advantage that a passive adversary can obtain when it is restricted to access the Test oracle only once.

Lemma 1: For any group key exchange protocol P,

$$\mathsf{Adv}_P(t, Q) \leq q_{\text{test}} \cdot \mathsf{Adv}_P(t, Q')$$

where $Q = (q_{\text{exec}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{test}})$ and $Q' = (q_{\text{exec}}, q_{\text{reve}} + q_{\text{test}} - 1, q_{\text{corr}}, 1)$.

Proof: The idea of the proof is essentially the same as in the proof of Lemma 2 in Appendix B of [14], where they dealt with the case of $Q = (q_{\text{exec}}, 0, 0, q_{\text{test}})$ and $Q' = (q_{\text{exec}}, q_{\text{test}} - 1, 0, 1)$.

Let $\mathcal{A}$ be an adversary against the security of a group key exchange protocol P, with time complexity t and query complexity $Q = (q_{\text{exec}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{test}})$. Recall that the advantage of $\mathcal{A}$ in attacking P is the probability that $\mathcal{A}$ outputs 1 in the real experiment $\mathsf{Exp}_P^{\mathsf{real}}(\mathcal{A})$ minus the probability that $\mathcal{A}$ outputs 1 in the random experiment $\mathsf{Exp}_P^{\mathsf{rand}}(\mathcal{A})$, namely, $\mathsf{Adv}_P(\mathcal{A}) = \left|\Pr[\mathsf{Exp}_P^{\mathsf{real}}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}_P^{\mathsf{rand}}(\mathcal{A}) = 1]\right|$.

The proof proceeds by a standard hybrid argument [58]. Consider a sequence of $q_{\text{test}} + 1$ hybrid experiments $\mathsf{Exp}_P^i(\mathcal{A})$, $0 \leq i \leq q_{\text{test}}$, where each $\mathsf{Exp}_P^i(\mathcal{A})$ is defined as follows.

Experiment $\mathsf{Exp}_P^i(\mathcal{A})$:

---

1. The adversary $\mathcal{A}$ interacts with the oracles, asking queries at will as many times as it wants. The interaction proceeds as specified in the model except that $\mathcal{A}$'s queries to the Test oracle are handled differently, as follows:
   - *The first i queries to the* Test *oracle are answered with*

a random session key and all remaining Test *queries are answered with the real session key.*

2. Some time after $\mathcal{A}$ asked all its queries, it outputs 0 or 1 as the outcome of the experiment.

Clearly, the experiments $\mathsf{Exp}_P^0(\mathcal{A})$ and $\mathsf{Exp}_P^{q_{\text{test}}}(\mathcal{A})$ at the extremes of the sequence are identical to $\mathsf{Exp}_P^{\text{real}}(\mathcal{A})$ and $\mathsf{Exp}_P^{\text{rand}}(\mathcal{A})$, respectively. Notice that as we move from $\mathsf{Exp}_P^{i-1}(\mathcal{A})$ to $\mathsf{Exp}_P^{i}(\mathcal{A})$ in the sequence, we change the response of i-th Test query from the real session key to a random session key. Since there are q_{test} such moves from $\mathsf{Exp}_P^{\text{real}}(\mathcal{A})$ to $\mathsf{Exp}_P^{\text{rand}}(\mathcal{A})$, the inequality of the lemma follows immediately if we prove that the difference between the probabilities that $\mathcal{A}$ outputs 1 in any two neighboring experiments $\mathsf{Exp}_P^{i-1}(\mathcal{A})$ and $\mathsf{Exp}_P^{i}(\mathcal{A})$ is at most $\mathsf{Adv}_P(t, Q')$ where $Q' = (q_{\text{exec}}, q_{\text{reve}} + q_{\text{test}} - 1, q_{\text{corr}}, 1)$. That is, to complete the proof, it remains to show that for every $1 \le i \le q_{\text{test}}$,

$$\left|\Pr[\mathsf{Exp}_P^{i-1}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}_P^{i}(\mathcal{A}) = 1]\right| \le \mathsf{Adv}_P(t, Q'). \tag{1}$$

Let $\varepsilon = |\Pr[\mathsf{Exp}_P^{i-1}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}_P^{i}(\mathcal{A}) = 1]|$. Then, using the adversary $\mathcal{A}$, we construct an adversary $\mathcal{A}_i$ attacking the protocol P, with advantage ε, time complexity t, and query complexity $Q' = (q_{\text{exec}}, q_{\text{reve}} + q_{\text{test}} - 1, q_{\text{corr}}, 1)$. $\mathcal{A}_i$ begins by invoking the adversary $\mathcal{A}$, then proceeds to answer the oracle queries of $\mathcal{A}$ using its own oracle queries, and finally ends by outputting whatever bit $\mathcal{A}$ eventually outputs. $\mathcal{A}_i$ answers the queries of $\mathcal{A}$ as follows:

- When $\mathcal{A}$ asks a query to the Execute, Reveal, or Corrupt oracle, $\mathcal{A}_i$ answers it in a straightforward way by sending the same query to its own corresponding oracle and then simply forwarding to $\mathcal{A}$ the outcome of its oracle query.
- If $\mathcal{A}$ queries the Test oracle, then there are three cases to handle:
  - For the first $i-1$ Test queries, $\mathcal{A}_i$ answers them with a random session key.
  - On the i-th Test query, $\mathcal{A}_i$ queries its own Test oracle and returns the result of its Test query.
  - For all the remaining Test queries, $\mathcal{A}_i$ answers them with the real session key by accessing its own Reveal oracle.

It is easy to see that $\mathcal{A}_i$ has time complexity t and query complexity at most $Q' = (q_{\text{exec}}, q_{\text{reve}} + q_{\text{test}} - 1, q_{\text{corr}}, 1)$.

To quantify the advantage of $\mathcal{A}_i$, it now suffices to notice the following two facts:

- The probability that $\mathcal{A}_i$ outputs 1 when its Test oracle has returned the real session key is exactly $\Pr[\mathsf{Exp}_P^{i-1}(\mathcal{A}) = 1]$.
- The probability that $\mathcal{A}_i$ outputs 1 when its Test oracle has returned a random session key is identical to $\Pr[\mathsf{Exp}_P^{i}(\mathcal{A}) = 1]$.

The advantage of $\mathcal{A}_i$ in attacking protocol P, $\mathsf{Adv}_P(\mathcal{A}_i)$, is therefore exactly $\varepsilon = |\Pr[\mathsf{Exp}_P^{i-1}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}_P^{i}(\mathcal{A}) = 1]|$. Since $\mathsf{Adv}_P(\mathcal{A}_i) \le \mathsf{Adv}_P(t, Q')$ by definition, we obtain (1) above. This gives the desired result of the lemma. □

A. *Proof of Security for the Basic Protocol* BP

As the following theorem states, the basic protocol BP is a group key exchange protocol secure against a passive adversary who calls the Test oracle only once, as long as the DDH assumption holds in $\mathbb{G}$.

Theorem 1: Let $Q = (q_{\text{exec}}, q_{\text{reve}}, q_{\text{corr}}, 1)$. Then, we have

$$\mathsf{Adv}_{\text{BP}}(t, Q) \le 2q_{\text{exec}} \cdot \mathsf{Adv}_{\mathbb{G}}^{\text{ddh}}(t')$$

where $t' = t + O(|\mathcal{U}| q_{\text{exec}} t_{\text{BP}})$ and t_{BP} is the time needed for execution of protocol BP by any party.

Combining this theorem with Lemma 1 immediately yields the following corollary which states that the group key exchange protocol BP is secure against a passive adversary under the DDH assumption for $\mathbb{G}$.

Corollary 1: Let $Q = (q_{\text{exec}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{test}})$. Then, we have

$$\mathsf{Adv}_{\text{BP}}(t, Q) \le 2q_{\text{test}} q_{\text{exec}} \cdot \mathsf{Adv}_{\mathbb{G}}^{\text{ddh}}(t')$$

where t' is as in Theorem 1.

We now proceed to prove Theorem 1.

Proof: Let $\mathcal{A}_b$ be a passive adversary attacking the basic protocol BP, with time complexity t and query complexity $Q = (q_{\text{exec}}, q_{\text{reve}}, q_{\text{corr}}, 1)$. Assume that the probability that $\mathcal{A}_b$ correctly guesses the value of the hidden bit b used by the Test oracle is $1/2 + \epsilon$. Then, we construct from $\mathcal{A}_b$ a distinguisher $\mathcal{D}$ that solves the DDH problem for $\mathbb{G}$ with probability $\epsilon / q_{\text{exec}}$.

To construct the distinguisher $\mathcal{D}$, we first need to consider the following two distributions:

$$\mathsf{Real}_{\text{BP}} \overset{\text{def}}{=} \left\{ (\mathsf{T}, K) \;\middle|\; \begin{array}{l} s, r_1, \cdots, r_n \in_R \mathbb{Z}_q; \\ w = g^s, z_1 = g^{r_1}, \cdots, z_n = g^{r_n}; \\ x_1 = g^{sr_1}, \cdots, x_n = g^{sr_n}; \\ X = x_1 \cdots x_n; \\ y_2 = X \cdot x_2^{-1}, \cdots, y_n = X \cdot x_n^{-1}; \\ \mathsf{T} = (w, z_2, \cdots, z_n, y_2, \cdots, y_n); \\ K = X \cdot y_2 \cdots y_n \end{array} \right\}$$

and

$$\mathsf{Fake}_{\text{BP}} \overset{\text{def}}{=} \left\{ (\mathsf{T}, K) \;\middle|\; \begin{array}{l} s, r_1, \cdots, r_n, a_1, \cdots, a_n \in_R \mathbb{Z}_q; \\ w = g^s, z_1 = g^{r_1}, \cdots, z_n = g^{r_n}; \\ x_1 = g^{a_1}, \cdots, x_n = g^{a_n}; \\ X = x_1 \cdots x_n; \\ y_2 = X \cdot x_2^{-1}, \cdots, y_n = X \cdot x_n^{-1}; \\ \mathsf{T} = (w, z_2, \cdots, z_n, y_2, \cdots, y_n); \\ K = X \cdot y_2 \cdots y_n \end{array} \right\}.$$

The distribution $\mathsf{Real}_{\text{BP}}$ matches exactly the real execution of BP among n users $U_1, \cdots, U_n$. The distribution $\mathsf{Fake}_{\text{BP}}$ is obtained from $\mathsf{Real}_{\text{BP}}$ by changing the way of computing x_i for $i \in [1, n]$; in $\mathsf{Fake}_{\text{BP}}$, each x_i is computed as g^{a_i} where a_i is drawn at random from $\mathbb{Z}_q$ instead of being equal to sr_i.

We now claim that distinguishing between two distributions Real_{BP} and Fake_{BP} is at least as difficult as solving the DDH problem for $\mathbb{G}$.

Lemma 2: Let $\mathcal{D}'$ be a distinguisher that given as input (T, K) coming from one of the two distributions Real_{BP} and Fake_{BP}, runs in time t and outputs 0 or 1. Then, we have

$$\begin{aligned}|\Pr[\mathcal{D}'(\mathsf{T}, K) = 1 \mid (\mathsf{T}, K) \leftarrow \mathsf{Real}_{BP}] \\ - \Pr[\mathcal{D}'(\mathsf{T}, K) = 1 \mid (\mathsf{T}, K) \leftarrow \mathsf{Fake}_{BP}]| \\ \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t')\end{aligned}$$

where $t' = t + O(nt_{\exp})$ and $t_{\exp}$ is the time required to perform an exponentiation in $\mathbb{G}$.

Proof: In order to prove the lemma, we show how to build from $\mathcal{D}'$ a distinguisher $\mathcal{D}''$ that solves the DDH problem in $\mathbb{G}$. Let $(g^s, g^{r_2}, g^{s'r_2}) \in \mathbb{G}^3$ be an instance of the DDH problem given as input to $\mathcal{D}''$. Using the triple $(g^s, g^{r_2}, g^{s'r_2})$, $\mathcal{D}''$ first generates (T, K) according to the following distribution Dist_{BP}:

$$\mathsf{Dist}_{BP} \overset{\text{def}}{=} \left\{ (\mathsf{T}, K) \;\middle|\; \begin{array}{l} r_1, \alpha_3, \beta_3, \cdots, \alpha_n, \beta_n \in_R \mathbb{Z}_q; \\ w = g^s, z_1 = g^{r_1}, z_2 = g^{r_2}, \\ z_3 = g^{r_1\alpha_3 + r_2\beta_3}, \cdots, z_n = g^{r_1\alpha_n + r_2\beta_n}; \\ x_1 = g^{sr_1}, x_2 = g^{s'r_2}, \\ x_i = g^{sr_1\alpha_i + s'r_2\beta_i} \text{ for } i \in [3, n]; \\ X = x_1 \cdots x_n; \\ y_2 = X \cdot x_2^{-1}, \cdots, y_n = X \cdot x_n^{-1}; \\ \mathsf{T} = (w, z_2, \cdots, z_n, y_2, \cdots, y_n); \\ K = X \cdot y_2 \cdots y_n \end{array} \right\}.$$

Then, $\mathcal{D}''$ runs $\mathcal{D}'(\mathsf{T}, K)$ and outputs whatever bit $\mathcal{D}'$ eventually outputs.

If $(g^s, g^{r_2}, g^{s'r_2})$ is a true Diffie-Hellman triple (i.e., $s = s'$), then we have $\mathsf{Dist}_{BP} \equiv \mathsf{Real}_{BP}$ since $x_i = z_i^s$ for all $i \in [1, n]$. If instead $(g^s, g^{r_2}, g^{s'r_2})$ is a random triple, then it is clear that $\mathsf{Dist}_{BP} \equiv \mathsf{Fake}_{BP}$. This means that:

1. The probability that $\mathcal{D}''$ outputs 1 on a true Diffie-Hellman triple is exactly the probability that $\mathcal{D}'$ outputs 1 on (T, K) generated according to the distribution Real_{BP}.
2. The probability that $\mathcal{D}''$ outputs 1 on a random triple is identical to the probability that $\mathcal{D}'$ outputs 1 on (T, K) generated according to the distribution Fake_{BP}.

The claim of the lemma immediately follows because the running time of $\mathcal{D}''$ is the running time of $\mathcal{D}'$ added to the time to generate (T, K) according to Dist_{BP}. □

We now make the following claim about the Fake_{BP} distribution.

Lemma 3: For any computationally unbounded adversary $\mathcal{A}$, we have

$$\begin{aligned}\Pr[\mathcal{A}(\mathsf{T}, K_{(b)}) = b \mid \\ (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Fake}_{BP}; K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0, 1\}] = 1/2.\end{aligned}$$

Proof: Let us write $\log_g \mu$ to denote the exponent ν such that $\mu = g^\nu$. Then, in distribution Fake_{BP}, the transcript T constrains the exponents a_i only by the following $n - 1$ equations:

$$\begin{aligned}\log_g y_2 &= -a_2 + \sum_{i=1}^{n} a_i \\ \log_g y_3 &= -a_3 + \sum_{i=1}^{n} a_i \\ &\vdots \\ \log_g y_n &= -a_n + \sum_{i=1}^{n} a_i.\end{aligned}$$

Since the equation $\log_g X = \sum_{i=1}^n a_i$ is linearly independent from the set of $n - 1$ equations above, the group secret X is independent of the transcript T. This implies that for any computationally unbounded adversary $\mathcal{A}$:

$$\begin{aligned}\Pr[\mathcal{A}(\mathsf{T}, X_{(b)}) = b \mid \\ (\mathsf{T}, X_{(1)}) \leftarrow \mathsf{Fake}_{BP}; X_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0, 1\}] = 1/2.\end{aligned}$$

Since $K = X \cdot y_2 \cdots y_n$ and y_i's are all public, the statement of the lemma immediately follows by a simple standard argument. □

We are now ready to describe the construction of the distinguisher $\mathcal{D}$. Assume without loss of generality that $\mathcal{A}_b$ makes its Test query to an instance activated by the γ-th $\mathsf{Execute}$ query. The distinguisher $\mathcal{D}$ begins by choosing a random $\delta \in \{1, \cdots, q_{\text{exec}}\}$ as a guess for the value of γ and by choosing a bit b uniformly at random from $\{0, 1\}$. $\mathcal{D}$ then simply runs $\mathcal{A}_b$ as a subroutine and answers the oracle queries of $\mathcal{A}_b$. Since there is no long-term secret information used in the protocol BP, $\mathcal{D}$ may ignore all queries of $\mathcal{A}_b$ to the $\mathsf{Corrupt}$ oracle. For all other queries, except the δ-th $\mathsf{Execute}$ query, $\mathcal{D}$ answers them in the natural way by executing the protocol BP on its own. But when $\mathcal{A}_b$ asks the δ-th $\mathsf{Execute}$ query, $\mathcal{D}$ slightly deviates from the protocol, embedding an instance of the DDH problem given as input into the transcript as follows: Using its problem instance $(g^s, g^{r_2}, g^{s'r_2}) \in \mathbb{G}^3$, $\mathcal{D}$ generates (T, K) according to the distribution Dist_{BP} and answers the δ-th $\mathsf{Execute}$ query of $\mathcal{A}_b$ with T. The distinguisher $\mathcal{D}$ aborts and outputs a random bit if $\delta \neq \gamma$. Otherwise, $\mathcal{D}$ answers the Test query of $\mathcal{A}_b$ with K if $b = 1$, and with a random key otherwise. Now at some point in time, when $\mathcal{A}_b$ terminates and outputs its guess b', $\mathcal{D}$ outputs 1 if $b = b'$, and 0 otherwise.

We now analyze the advantage of $\mathcal{D}$ in solving the DDH problem for $\mathbb{G}$. Suppose that $\mathcal{A}_b$ asked its Test query to an instance activated by the δ-th $\mathsf{Execute}$ query; this happens with probability $1/q_{\text{exec}}$. If $(g^s, g^{r_2}, g^{s'r_2})$ is a true Diffie-Hellman triple, then, by Lemma 2, $\mathsf{Dist}_{BP} \equiv \mathsf{Real}_{BP}$ and thus, by assumption, $\Pr[b = b'] = 1/2 + \epsilon$. So, the probability that $\mathcal{D}$ outputs 1 on a true Diffie-Hellman triple is also $1/2 + \epsilon$. If instead $(g^s, g^{r_2}, g^{s'r_2})$ is a random triple, then $\mathsf{Dist}_{BP} \equiv \mathsf{Fake}_{BP}$ and hence, $\Pr[b = b'] = 1/2$ by Lemma 3. Thus, the probability that $\mathcal{D}$ outputs 1 on a random triple is exactly $1/2$. Now since $\Pr[\delta = \gamma] = 1/q_{\text{exec}}$, we obtain

$$\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{D}) = \epsilon / q_{\text{exec}}.$$

Finally, since $\mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}_b) = 2\epsilon$ by definition, Theorem 1 follows immediately if we notice that $\mathcal{D}$ takes at most time $t' = t + O(|\mathcal{U}|q_{\mathrm{exec}}t_{\mathrm{BP}})$. □

B. *Proof of Security for the Generalized Protocol* GP

We now turn to proving the security of our main protocol GP against a passive adversary. Corollary 2 below presents the concrete result of security for protocol GP. The first step towards proving the corollary has already been taken with the proof of Lemma 1 given earlier in this section. Recall that by Lemma 1, we showed that the security of a protocol against a passive adversary asking multiple Test queries can be reduced to the security of the same protocol against a passive adversary asking only a single Test query. So we are left with proving the following Theorem 2 which claims the security of the protocol GP against a passive adversary who queries the Test oracle only once. We prove this claim by finding a reduction from the problem of breaking protocol GP to the problem of breaking the underlying protocol BP. The proof proceeds very much along the lines of that of Theorem 1, extending the techniques used there to this more general case.

Theorem 2: Let $Q_g = (q_{\mathrm{exec}}, q_{\mathrm{reve}}, q_{\mathrm{corr}}, 1)$ and $Q_b = (m, 0, 0, m)$, where m is the number of users at level 1 of the tree used in protocol GP (see the description of protocol GP). Then, we have

$$\mathsf{Adv}_{\mathrm{GP}}(t, Q_g) \le (2q_{\mathrm{exec}} + 1) \cdot \mathsf{Adv}_{\mathrm{BP}}(t', Q_b)$$

where $t' = t + O(|\mathcal{U}|q_{\mathrm{exec}}t_{\mathrm{GP}})$ and t_{GP} is the time required for execution of protocol GP by any party.

As mentioned above, this theorem together with Lemma 1 immediately yields the following corollary which gives one of our main results of this section: The group key exchange protocol GP is secure against a passive adversary under the security of the protocol BP, which in turn has been proven under the DDH assumption for $\mathbb{G}$.

Corollary 2: Let $Q_g = (q_{\mathrm{exec}}, q_{\mathrm{reve}}, q_{\mathrm{corr}}, q_{\mathrm{test}})$ and $Q_b = (m, 0, 0, m)$, where m is as in Theorem 2. Then, we have

$$\mathsf{Adv}_{\mathrm{GP}}(t, Q_g) \le q_{\mathrm{test}}(2q_{\mathrm{exec}} + 1) \cdot \mathsf{Adv}_{\mathrm{BP}}(t', Q_b)$$

where t' is as in Theorem 2.

We proceed with the proof of Theorem 2.

Proof: As usual, the proof of the theorem proceeds by a standard reduction argument. Let $\mathcal{A}_g$ be a passive adversary attacking the generalized protocol GP, with time complexity t and query complexity $Q_g = (q_{\mathrm{exec}}, q_{\mathrm{reve}}, q_{\mathrm{corr}}, 1)$. Given the adversary $\mathcal{A}_g$, we construct a passive adversary $\mathcal{A}_b$ attacking the basic protocol BP, with time complexity t' and query complexity $Q_b = (m, 0, 0, m)$.

For ease of exposition, we first introduce some additional notations. Consider the tree structure of n users given in Fig. 1(a) of Section II and recall that $\mathcal{G}_i$ denotes the subgroup consisting of the users in the subtree rooted at the node hosting U_i. For $i \in [2, m+1]$, let T_i denote the transcript of the basic protocol BP executed by the subgroup $\mathcal{G}_i$ to generate the subgroup key k_i. Then, using the distribution $\mathsf{Real}_{\mathrm{BP}}$ defined in the proof of Theorem 1, we can write the honest generation of (T_i, k_i) by subgroup $\mathcal{G}_i$ as $(\mathsf{T}_i, k_i) \leftarrow \mathsf{Real}_{\mathrm{BP}}$. We also write $(\mathsf{T}_i, a_i) \leftarrow \mathsf{Rand}_{\mathrm{BP}}$ to denote the generation of (T_i, a_i), where T_i is generated according to distribution $\mathsf{Real}_{\mathrm{BP}}$ and a_i is a random key chosen independently of T_i but chosen uniformly from $\mathbb{G}$.

With these notations, we now introduce the following two distributions:

$$\mathsf{Real}_{\mathrm{GP}} \overset{\mathrm{def}}{=} \left\{ (\mathsf{T}, K) \;\middle|\; \begin{array}{l} (\mathsf{T}_2, k_2), \cdots, (\mathsf{T}_{m+1}, k_{m+1}) \leftarrow \mathsf{Real}_{\mathrm{BP}}; \\ s, r_1 \in_R \mathbb{Z}_q; \\ r_2 = I(k_2), \cdots, r_{m+1} = I(k_{m+1}); \\ w = g^s, z_1 = g^{r_1}, \cdots, z_{m+1} = g^{r_{m+1}}; \\ x_1 = g^{sr_1}, \cdots, x_{m+1} = g^{sr_{m+1}}; \\ X = x_1 \cdots x_{m+1}; \\ y_2 = X \cdot x_2^{-1}, \cdots, y_{m+1} = X \cdot x_{m+1}^{-1}; \\ \mathsf{T} = (\mathsf{T}_2, \cdots, \mathsf{T}_{m+1}, \\ \qquad w, z_2, \cdots, z_{m+1}, y_2, \cdots, y_{m+1}); \\ K = X \cdot y_2 \cdots y_{m+1} \end{array} \right\}$$

and

$$\mathsf{Rand}_{\mathrm{GP}} \overset{\mathrm{def}}{=} \left\{ (\mathsf{T}, K) \;\middle|\; \begin{array}{l} (\mathsf{T}_2, a_2), \cdots, (\mathsf{T}_{m+1}, a_{m+1}) \leftarrow \mathsf{Rand}_{\mathrm{BP}}; \\ s, r_1 \in_R \mathbb{Z}_q; \\ r_2 = I(a_2), \cdots, r_{m+1} = I(a_{m+1}); \\ w = g^s, z_1 = g^{r_1}, \cdots, z_{m+1} = g^{r_{m+1}}; \\ x_1 = g^{sr_1}, \cdots, x_{m+1} = g^{sr_{m+1}}; \\ X = x_1 \cdots x_{m+1}; \\ y_2 = X \cdot x_2^{-1}, \cdots, y_{m+1} = X \cdot x_{m+1}^{-1}; \\ \mathsf{T} = (\mathsf{T}_2, \cdots, \mathsf{T}_{m+1}, \\ \qquad w, z_2, \cdots, z_{m+1}, y_2, \cdots, y_{m+1}); \\ K = X \cdot y_2 \cdots y_{m+1} \end{array} \right\}.$$

$\mathsf{Real}_{\mathrm{GP}}$ corresponds to the distribution of the transcript T and the session key K generated in the real execution of protocol GP. The distribution $\mathsf{Rand}_{\mathrm{GP}}$ is obtained from $\mathsf{Real}_{\mathrm{GP}}$ by replacing each subgroup key k_i with a random key a_i.

We now claim that distinguishing between the two distributions $\mathsf{Real}_{\mathrm{GP}}$ and $\mathsf{Rand}_{\mathrm{GP}}$ is no easier than breaking the security of the basic protocol BP.

Lemma 4: Let $\mathcal{D}$ be a distinguisher that given as input (T, K) coming from one of the two distributions $\mathsf{Real}_{\mathrm{GP}}$ and $\mathsf{Rand}_{\mathrm{GP}}$, runs in time t and outputs 0 or 1. Let $Q_b = (m, 0, 0, m)$ where m is as in Theorem 2. Then, we have

$$\begin{aligned} &|\Pr[\mathcal{D}(\mathsf{T}, K) = 1 \mid (\mathsf{T}, K) \leftarrow \mathsf{Real}_{\mathrm{GP}}] \\ &\quad - \Pr[\mathcal{D}(\mathsf{T}, K) = 1 \mid (\mathsf{T}, K) \leftarrow \mathsf{Rand}_{\mathrm{GP}}]| \\ &\qquad\qquad \le \mathsf{Adv}_{\mathrm{BP}}(t', Q_b) \end{aligned}$$

where $t' = t + O(mt_{\mathrm{exp}})$ and t_{exp} is the time required to perform an exponentiation in $\mathbb{G}$.

Proof: Suppose that μ and ν are the probabilities that $\mathcal{D}$ outputs 1 on (T, K) generated according to $\mathsf{Real}_{\mathrm{GP}}$ and

$\mathsf{Rand}_{\mathrm{GP}}$, respectively. Using the distinguisher $\mathcal{D}$, we construct an adversary $\mathcal{A}'_b$ whose advantage in attacking the basic protocol BP is $|\mu - \nu|$.

Recall first that the tree structure used in the generalized protocol GP is public and so the value of m and the users of each subgroup $\mathcal{G}_i$ are known to the adversary $\mathcal{A}'_b$. Using these information, $\mathcal{A}'_b$ makes m queries to its Execute oracle, one for each subgroup $\mathcal{G}_i$, and receives in return m transcripts T_2, T_3, $\cdots$, T_{m+1}. Let $\Pi_{U\in\mathcal{G}_i}$ denote any instance activated by the Execute query directed to $\mathcal{G}_i$. Now $\mathcal{A}'_b$ asks m Test queries Test($\Pi_{U\in\mathcal{G}_2}$), Test($\Pi_{U\in\mathcal{G}_3}$), $\cdots$, Test($\Pi_{U\in\mathcal{G}_{m+1}}$); recall that in our model, the adversary is allowed to ask multiple queries to its Test oracle as long as the tested instances are fresh and no two of them are partnered together. Let k'_i be either the real session key or a random session key returned in response to the query Test($\Pi_{U\in\mathcal{G}_i}$). We write $(\mathsf{T}_i, k'_i) \leftarrow \mathsf{Test}_{\mathrm{BP}}$ to denote this way of generating a transcript-key pair (T_i, k'_i).

Having made the queries and received the results as above, $\mathcal{A}'_b$ generates (T, K) according to the distribution $\mathsf{Dist}_{\mathrm{GP}}$ (defined below), runs $\mathcal{D}(\mathsf{T}, K)$, and outputs whatever bit $\mathcal{D}$ outputs. Distribution $\mathsf{Dist}_{\mathrm{GP}}$ is defined as follows:

$$\mathsf{Dist}_{\mathrm{GP}} \overset{\mathrm{def}}{=} \left\{ (\mathsf{T}, K) \;\middle|\; \begin{array}{l} (\mathsf{T}_2, k'_2), \cdots, (\mathsf{T}_{m+1}, k'_{m+1}) \leftarrow \mathsf{Test}_{\mathrm{BP}}; \\ s, r_1 \in_R \mathbb{Z}_q; \\ r_2 = I(k'_2), \cdots, r_{m+1} = I(k'_{m+1}); \\ w = g^s, z_1 = g^{r_1}, \cdots, z_{m+1} = g^{r_{m+1}}; \\ x_1 = g^{sr_1}, \cdots, x_{m+1} = g^{sr_{m+1}}; \\ X = x_1 \cdots x_{m+1}; \\ y_2 = X \cdot x_2^{-1}, \cdots, y_{m+1} = X \cdot x_{m+1}^{-1}; \\ \mathsf{T} = (\mathsf{T}_2, \cdots, \mathsf{T}_{m+1}, \\ \quad w, z_2, \cdots, z_{m+1}, y_2, \cdots, y_{m+1}); \\ K = X \cdot y_2 \cdots y_{m+1} \end{array} \right\}.$$

Notice that to generate (T, K) according to $\mathsf{Dist}_{\mathrm{GP}}$, $\mathcal{A}'_b$ performs $O(m)$ exponentiations in $\mathbb{G}$ and makes m Execute queries and m Test queries. Therefore, if we let t be the time complexity of $\mathcal{D}$, $\mathcal{A}'_b$ has time complexity $t' = t + O(mt_{\exp})$ and query complexity $Q_b = (m, 0, 0, m)$.

The only possible difference between the distribution $\mathsf{Dist}_{\mathrm{GP}}$ and the other two distributions $\mathsf{Real}_{\mathrm{GP}}$ and $\mathsf{Rand}_{\mathrm{GP}}$ is in the way of generating the subgroup keys. If each k'_i is the real session key, clearly we have $\mathsf{Dist}_{\mathrm{GP}} \equiv \mathsf{Real}_{\mathrm{GP}}$. On the other hand, if each k'_i is a random session key chosen independently of the transcript T_i, then $\mathsf{Dist}_{\mathrm{GP}} \equiv \mathsf{Rand}_{\mathrm{GP}}$. This means that:

1. The probability that $\mathcal{A}'_b$ outputs 1 when k'_2, $\cdots$, k'_{m+1} are real session keys is exactly μ, the probability that $\mathcal{D}$ outputs 1 on (T, K) generated according to the distribution $\mathsf{Real}_{\mathrm{GP}}$.
2. The probability that $\mathcal{A}'_b$ outputs 1 when k'_2, $\cdots$, k'_{m+1} are random session keys is exactly ν, the probability that $\mathcal{D}$ outputs 1 on (T, K) generated according to the distribution $\mathsf{Rand}_{\mathrm{GP}}$.

Thus $\mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}'_b) = |\mu - \nu|$. Combining this and $\mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}'_b) \leq \mathsf{Adv}_{\mathrm{BP}}(t', Q_b)$ yields the statement of Lemma 4. □

We continue with the following lemma.

Lemma 5: For any computationally unbounded adversary $\mathcal{A}$, we have

$$\begin{aligned} &\Pr[\mathcal{A}(\mathsf{T}, K_{(b)}) = b \mid \\ &\qquad (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Rand}_{\mathrm{GP}}; K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0,1\}] = \\ &\Pr[\mathcal{A}(\mathsf{T}, K_{(b)}) = b \mid \\ &\qquad (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Real}_{\mathrm{BP}}; K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0,1\}]. \end{aligned}$$

Proof: In distribution $\mathsf{Rand}_{\mathrm{GP}}$, the session key K is completely independent from the set of m transcripts $\{\mathsf{T}_i \mid i \in [2, m+1]\}$ since each $a_i \in \mathbb{G}$ is chosen at random independently of the transcript T_i. Therefore, if we define $\mathsf{Rand}'_{\mathrm{GP}}$ as the distribution derived from $\mathsf{Rand}_{\mathrm{GP}}$ by eliminating the transcripts $\{\mathsf{T}_i \mid i \in [2, m+1]\}$, it is clear that:

$$\begin{aligned} &\Pr[\mathcal{A}(\mathsf{T}, K_{(b)}) = b \mid \\ &\qquad (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Rand}'_{\mathrm{GP}}; K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0,1\}] = \\ &\Pr[\mathcal{A}(\mathsf{T}, K_{(b)}) = b \mid \\ &\qquad (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Rand}_{\mathrm{GP}}; K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0,1\}]. \quad (2) \end{aligned}$$

Since $\mathsf{Rand}'_{\mathrm{GP}}$ is identical to $\mathsf{Real}_{\mathrm{BP}}$ (not to be confused with $\mathsf{Real}_{\mathrm{GP}}$), it is also immediate that:

$$\begin{aligned} &\Pr[\mathcal{A}(\mathsf{T}, K_{(b)}) = b \mid \\ &\qquad (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Rand}'_{\mathrm{GP}}; K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0,1\}] = \\ &\Pr[\mathcal{A}(\mathsf{T}, K_{(b)}) = b \mid \\ &\qquad (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Real}_{\mathrm{BP}}; K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0,1\}]. \quad (3) \end{aligned}$$

By combining (2) and (3), we obtain the statement of Lemma 5. □

Before continuing further, let us define

$$\begin{aligned} &\mathsf{SuccPr}_{\mathrm{BP}}(\mathcal{A}_g) \overset{\mathrm{def}}{=} \\ &\quad \Pr[\mathcal{A}_g(\mathsf{T}, K_{(b)}) = b \mid (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Real}_{\mathrm{BP}}; \\ &\qquad\qquad K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0,1\}] \end{aligned}$$

and

$$\begin{aligned} &\mathsf{SuccPr}_{\mathrm{GP}}(\mathcal{A}_g) \overset{\mathrm{def}}{=} \\ &\quad \Pr[\mathcal{A}_g(\mathsf{T}, K_{(b)}) = b \mid (\mathsf{T}, K_{(1)}) \leftarrow \mathsf{Real}_{\mathrm{GP}}; \\ &\qquad\qquad K_{(0)} \leftarrow \mathbb{G}; b \leftarrow \{0,1\}]. \end{aligned}$$

Armed with the two lemmas above, we now give the details of the construction of the adversary $\mathcal{A}_b$. Assume without loss of generality that $\mathcal{A}_g$ makes its Test query to an instance activated by the γ-th Execute query. The adversary $\mathcal{A}_b$ begins by choosing a random $\delta \in \{1, \cdots, q_{\mathrm{exec}}\}$ as a guess for the value of γ and by choosing a bit b uniformly at random from $\{0,1\}$. It then runs $\mathcal{A}_g$ as a subroutine, simulating the oracles. $\mathcal{A}_b$ may ignore all Corrupt queries of $\mathcal{A}_g$ because there is no long-term secret information used in the protocol GP. For all other queries, except the δ-th Execute query, $\mathcal{A}_b$ answers them in the natural way by executing the protocol GP on its own. But when $\mathcal{A}_g$ asks the δ-th Execute query, $\mathcal{A}_b$ responds to the query by calling its own Execute and Test oracles; namely, $\mathcal{A}_b$ generates (T, K)

according to the distribution $\mathsf{Dist}_{\mathrm{GP}}$ and returns the transcript T in response to the query. The adversary $\mathcal{A}_b$ aborts and outputs a random bit if $\delta \neq \gamma$. Otherwise, $\mathcal{A}_b$ answers the Test query of $\mathcal{A}_g$ with K if $b = 1$, and with a random key otherwise. Now when $\mathcal{A}_g$ terminates and outputs its guess b', $\mathcal{A}_b$ outputs 1 if $b = b'$, and 0 otherwise.

From the above simulation of oracles, we can see that $\mathcal{A}_b$ has query complexity $Q_b = (m, 0, 0, m)$ and time complexity $t' = t + O(|\mathcal{U}| q_{\mathrm{exec}} t_{\mathrm{GP}})$ where t is the time complexity of $\mathcal{A}_g$.

We now analyze the advantage of $\mathcal{A}_b$ in attacking the protocol BP. Assume that $\mathcal{A}_g$ asked its Test query to an instance activated by the δ-th $\mathsf{Execute}$ query; this is the case with probability $1/q_{\mathrm{exec}}$. If $k'_2, k'_3, \cdots, k'_{m+1}$ in $\mathsf{Dist}_{\mathrm{GP}}$ are real session keys, then $\mathsf{Dist}_{\mathrm{GP}} \equiv \mathsf{Real}_{\mathrm{GP}}$ and thus, the probability that $\mathcal{A}_g$ correctly guesses the hidden bit b is $\mathsf{SuccPr}_{\mathrm{GP}}(\mathcal{A}_g)$. This means that the probability that $\mathcal{A}_b$ outputs 1 when its Test oracle returns actual session keys is also $\mathsf{SuccPr}_{\mathrm{GP}}(\mathcal{A}_g)$. On the other hand, if $k'_2, k'_3, \cdots, k'_{m+1}$ in $\mathsf{Dist}_{\mathrm{GP}}$ are all random session keys, then $\mathsf{Dist}_{\mathrm{GP}} \equiv \mathsf{Rand}_{\mathrm{GP}}$ and thus, by Lemma 5, the probability that $\mathcal{A}_g$ correctly guesses the hidden bit b is $\mathsf{SuccPr}_{\mathrm{BP}}(\mathcal{A}_g)$. Thus, the probability that $\mathcal{A}_b$ outputs 1 when its Test oracle returns random session keys is also $\mathsf{SuccPr}_{\mathrm{BP}}(\mathcal{A}_g)$. Therefore since $\Pr[\delta = \gamma] = 1/q_{\mathrm{exec}}$, we obtain:

$$\mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}_b) = \frac{1}{q_{\mathrm{exec}}} \left| \mathsf{SuccPr}_{\mathrm{GP}}(\mathcal{A}_g) - \mathsf{SuccPr}_{\mathrm{BP}}(\mathcal{A}_g) \right|. \quad (4)$$

Notice that this equation already implies that $|\mathsf{SuccPr}_{\mathrm{GP}}(\mathcal{A}_g) - \mathsf{SuccPr}_{\mathrm{BP}}(\mathcal{A}_g)|$ is negligible and so $\mathsf{SuccPr}_{\mathrm{GP}}(\mathcal{A}_g)$ is not much greater than $1/2$.

With (4), it is easy to bound the advantage of $\mathcal{A}_g$ in attacking the protocol GP. A straightforward calculation shows:

$$\begin{aligned} \mathsf{Adv}_{\mathrm{GP}}(\mathcal{A}_g) &= |2 \cdot \mathsf{SuccPr}_{\mathrm{GP}}(\mathcal{A}_g) - 1| \\ &\leq |2q_{\mathrm{exec}} \cdot \mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}_b) + 2 \cdot \mathsf{SuccPr}_{\mathrm{BP}}(\mathcal{A}_g) - 1| \\ &\leq 2q_{\mathrm{exec}} \cdot \mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}_b) + |2 \cdot \mathsf{SuccPr}_{\mathrm{BP}}(\mathcal{A}_g) - 1| \\ &= 2q_{\mathrm{exec}} \cdot \mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}_b) + \mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}_g). \end{aligned}$$

Since $\mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}_b) \leq \mathsf{Adv}_{\mathrm{BP}}(t', Q_b)$ and $\mathsf{Adv}_{\mathrm{BP}}(\mathcal{A}_g) \leq \mathsf{Adv}_{\mathrm{BP}}(t', Q_b)$, we obtain:

$$\mathsf{Adv}_{\mathrm{GP}}(\mathcal{A}_g) \leq (2q_{\mathrm{exec}} + 1) \cdot \mathsf{Adv}_{\mathrm{BP}}(t', Q_b).$$

This completes the proof of Theorem 2. □

V. CONCLUSION

In this paper, we have provided the first solution to the growing problem of group key exchange over combined wired and wireless networks, which consist of both low performance mobile devices with some form of battery power and high performance stationary computers with no power constraint. Our group key exchange protocol takes only a constant number of communication rounds and has the following key features that distinguish it from previous constant-round protocols.

1. By evenly distributing much of the total amount of computation among high performance computers, our protocol not only avoids any potential performance bottleneck of the system but also keeps the workload of mobile devices low and constant regardless of the group size n.
2. Let again n_s and n_r be the numbers of high power users and low power users, respectively. Then, if $n_s \geq \sqrt{n_r}$, our protocol bounds the maximum computation rate per user by $O(\sqrt{n})$, thus becoming the first constant-round protocol with computational complexity lower than $O(n)$.

To provide a high level of assurance that important security properties are satisfied in the proposed protocol, we have rigorously proved its security against a passive adversary in a well-defined formal model. Therefore, applying the compiler by Katz and Yung [20] to our protocol immediately results in a group key exchange protocol secure against an active adversary.

A typical topic for further research is to extend our work into a dynamic group setting where current members may leave the group and new members may join the group at any time in an arbitrary manner. Towards efficient key exchange in such a dynamic group, it is important to minimize the cost of the rekeying operations associated with group updates. As suggested by an anonymous referee, another direction of research is to devise a group key exchange protocol that provides more fine-grained resource awareness. For example, it would be interesting to see a protocol which takes advantage of the differences in computing capabilities among four groups of users: Very weak, mobile, stationary, and very powerful users.

ACKNOWLEDGEMENTS

This research was supported by the Postdoctoral Research Program of Sungkyunkwan University, Korea.

REFERENCES

[1] J. Nam, S. Kim, and D. Won, "Secure group communications over combined wired and wireless networks," in *Proc. 2nd Int. Conf. on Trust, Privacy, and Security in Digital Business*, 2005, vol. 3592, *LNCS*, pp. 90–99.

[2] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[3] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *IEEE Trans. Inform. Theory*, vol. 28, no. 5, pp. 714–720, 1982.

[4] E. Okamoto and K. Tanaka, "Key distribution system based on identification information," *IEEE J. Select. Areas Commun.*, vol. 7, no. 4, pp. 481–485, 1989.

[5] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proc. IEEE Symp. Security and Privacy*, 1992, pp. 72–84.

[6] A. Joux, "A one round protocol for tripartite Diffie-Hellman," *J. Crypto.*, vol. 17, no. 4, pp. 263–276, 2003.

[7] K.-K. R. Choo, "Provably-secure mutual authentication and key establishment protocols lounge," 2006, available at http://sky.fit.qut.edu.au/~choo /lounge.html.

[8] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. Crypto'93*, 1993, *LNCS*, vol. 773, pp. 232–249.

[9] M. Bellare and P. Rogaway, "Provably secure session key distribution—the three party case," in *Proc. 27th ACM Symp. Theory of Computing*, 1995, pp. 57–66.

[10] V. Shoup, "On formal models for secure key exchange," *Cryptology ePrint Archive*, Report 1999/012, 1999, available at http://eprint.iacr.org/1999/012.

[11] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. Eurocrypt 2000, LNCS*, vol. 1807, 2000, pp. 139–155.

[12] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," in *Proc. 8th ACM Conf. Computer and Commun. Security*, 2001, pp. 255–264.

[13] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *Proc. Eurocrypt 2002, LNCS*, vol. 2332, 2002, pp. 337–351.

[14] M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Proc. 8th Int. Work-*

shop Practice and Theory in Public Key Crypto., LNCS, vol. 3386, 2005, pp. 65–84.

[15] J. Katz and J. S. Shin, "Modeling insider attacks on group key-exchange protocols," in *Proc. 12th ACM Conf. Computer and Commun. Security*, 2005, pp. 180–189.

[16] B. Blanchet, "A computationally sound mechanized prover for security protocols," in *Proc. IEEE Symp. Security and Privacy*, 2006, pp. 140–154.

[17] K.-K. R. Choo, "Refuting security proofs for tripartite key exchange with model checker in planning problem setting," in *Proc. 19th IEEE Computer Security Foundations Workshop*, 2006, pp. 297–308.

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

[19] E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic group Diffie-Hellman key exchange under standard assumptions," in *Proc. Eurocrypt 2002, LNCS*, vol. 2332, 2002, pp. 321–336.

[20] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," in *Proc. Crypto 2003, LNCS*, vol. 2729, 2003, pp. 110–125.

[21] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," in *Proc. Crypto 2005, LNCS*, vol. 3621, 2005, pp. 546–566.

[22] K.-K. R. Choo, C. Boyd, and Y. Hitchcock, "Errors in computational complexity proofs for protocols," in *Proc. Asiacrypt 2005, LNCS*, vol. 3788, 2005, pp. 624–643.

[23] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "Analyzing the energy consumption of security protocols," in *Proc. ACM Int. Symp. Low Power Electron. and Des.*, 2003, pp. 30–35.

[24] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: the insecurity of 802.11," in *Proc. 7th ACM Conf. Mobile Computing and Networking*, 2001, pp. 180–189.

[25] D. Johnston and J. Walker, "Overview of IEEE 802.16 security," *IEEE Security Privacy*, vol. 2, no. 3, pp. 40–48, 2004.

[26] S.-L. Ng and C. Mitchell, "Comments on mutual authentication and key exchange protocols for low power wireless communications," *IEEE Commun. Lett.*, vol. 8, no. 4, pp. 262–263, 2004.

[27] J. Nam, S. Kim, and D. Won, "A weakness in the Bresson-Chevassut-Essiari-Pointcheval's group key agreement scheme for low-power mobile devices," *IEEE Commun. Lett.*, vol. 9, no. 5, pp. 429–431, 2005.

[28] G. Ateniese, M. Steiner, and G. Tsudik, "New multiparty authentication services and key agreement protocols," *IEEE J. Select. Areas Commun.*, vol. 18, no. 4, pp. 628–639, 2000.

[29] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 8, pp. 769–780, 2000.

[30] W.-G. Tzeng and Z.-J. Tzeng, "Round-efficient conference key agreement protocols with provable security," in *Proc. Asiacrypt 2000, LNCS*, vol. 1976, 2000, pp. 614–627.

[31] O. Pereira and J.-J. Quisquater, "A security analysis of the Cliques protocols suites," in *Proc. 14th IEEE Computer Security Foundations Workshop*, 2001, pp. 73–81.

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

[33] E. Bresson, O. Chevassut, and D. Pointcheval, "Group Diffie-Hellman key exchange secure against dictionary attacks," in *Proc. Asiacrypt 2002, LNCS*, vol. 2501, 2002, pp. 497–514.

[34] C. Boyd and J.M.G. Nieto, "Round-optimal contributory conference key agreement," in *Proc. 6th Int. Workshop Practice and Theory in Public Key Crypto., LNCS*, vol. 2567, 2003, pp. 161–174.

[35] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Trans. Inform. Syst. Security*, vol. 7, no. 1, pp. 60–96, 2004.

[36] E. Bresson and D. Catalano, "Constant round authenticated group key agreement via distributed computation," in *Proc. 7th Int. Workshop on Practice and Theory in Public Key Crypto., LNCS*, vol. 2947, 2004, pp. 115–129.

[37] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik, "On the performance of group key agreement protocols," *ACM Trans. Inform. Syst. Security*, vol. 7, no. 3, pp. 457–488, 2004.

[38] Y. Kim, A. Perrig, and G. Tsudik, "Group key agreement efficient in communication," *IEEE Trans. Comput.*, vol. 53, no. 7, pp. 905–921, 2004.

[39] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval, "Mutual authentication and group key agreement for low-power mobile devices," *Comput. Commun.*, vol. 27, no. 17, pp. 1730–1737, 2004.

[40] H.-J. Kim, S.-M. Lee, and D. H. Lee, "Constant-round authenticated group key exchange for dynamic groups," in *Proc. Asiacrypt 2004, LNCS*, vol. 3329, 2004, pp. 245–259.

[41] R. Dutta and R. Barua, "Constant round dynamic group key agreement," in *Proc. 8th Inform. Security Conf., LNCS*, vol. 3650, 2005, pp. 74–88.

[42] J. Nam, J. Lee, S. Kim, and D. Won, "DDH-based group key agreement in a mobile environment," *J. Syst. Softw.*, vol. 78, no. 1, pp. 73–83, 2005.

[43] M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval, "Password-based group key exchange in a constant number of rounds," in *Proc. 9th Int. Workshop Practice and Theory in Public Key Crypto., LNCS*, vol. 3958, 2006, pp. 427–442.

[44] Q. Tang and K.-K. R. Choo, "Secure password-based authenticated group key agreement for data-sharing peer-to-peer networks," in *Proc. 4th Int. Conf. Applied Crypto. and Network Security, LNCS*, vol. 3989, 2006, pp. 162–177.

[45] K. Becker and U. Wille, "Communication complexity of group key distribution," in *Proc. 5th ACM Conf. Computer and Commun. Security*, 1998, pp. 1–6.

[46] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *Proc. Eurocrypt 1994, LNCS*, vol. 950, 1994, pp. 275–286.

[47] J. Herranz and J. L. Villar, "An unbalanced protocol for group key exchange," in *Proc. 1st Int. Conf. Trust, Privacy, and Security in Digital Business, LNCS*, vol. 3184, 2004, pp. 172–180.

[48] G. Horn, K. M. Martin, and C. J. Mitchell, "Authentication protocols for mobile network environment value-added services," *IEEE Trans. Veh. Technol.*, vol. 51, no. 2, pp. 383–392, 2002.

[49] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," in *Proc. ACM SIGCOMM'98*, 1998, pp. 68–79.

[50] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: issues and architectures," *RFC 2627*, IETF, 1999.

[51] A. Perrig, D. Song, and J.D. Tygar, "ELK, a new protocol for efficient large-group key distribution," in *Proc. IEEE Symp. Security and Privacy*, 2001, pp. 247–262.

[52] W. Diffie, P. Oorschot, and M. Wiener, "Authentication and authenticated key exchanges," *Des., Codes, Crypto.*, vol. 2, no. 2, pp. 107–125, 1992.

[53] D. Denning and G. Sacco, "Timestamps in key distribution protocols," *Commun. ACM*, vol. 24, no. 8, pp. 533–536, 1981.

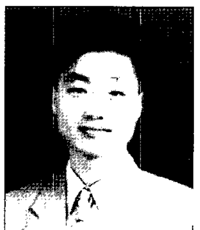
[54] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. 1st ACM Conf. Computer and Commun. Security*, 1993, pp. 62–73.

[55] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proc. Eurocrypt 2001*, 2001, *LNCS*, vol. 2045, pp. 453–474.

[56] A. O. Freier, P. Karlton, and P. C. Kocher, "The SSL protocol version 3.0," Internet draft, Netscape Communications, 1996.

[57] S. Kent and R. Atkinson, "Security architecture for the Internet protocol," *RFC 2401*, 1998.

[58] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.



Junghyun Nam received the B.E. degree in Information Engineering from Sungkyunkwan University, Korea, in 1997 and the M.S. degree in Computer Science from University of Louisiana, Lafayette, in 2002. His Ph.D. degree was received in Computer Engineering from Sungkyunkwan University, Korea, in 2006. His current research interest is in the area of cryptography and network security.

Dongho Won received his B.E., M.E., and Ph.D. degrees from Sungkyunkwan University in 1976, 1978, and 1988, respectively. After working at ETRI (Electronics & Telecommunications Research Institute) from 1978 to 1980, he joined Sungkyunkwan University in 1982, where he is currently a professor of School of Information and Communication Engineering. In the year 2002, he served as the president of KIISC (Korea Institute of Information Security & Cryptology). He was the program committee chairman of the 8th International Conference on Information Security and Cryptology (ICISC 2005). His research interests are on cryptology and information security.