# $n$PAKE$^+$: A Hierarchical Group Password-Authenticated Key Exchange Protocol Using Different Passwords

Zhiguo Wan[1], Robert H. Deng[2], Feng Bao[3], and Bart Preneel[1]

[1] K.U.Leuven, ESAT/SCD, Kasteelpark Arenberg 10, Leuven, Belgium
[2] School of Information Systems, Singapore Management University, Singapore
[3] Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore

**Abstract.** Although two-party password-authenticated key exchange (PAKE) protocols have been intensively studied in recent years, group PAKE protocols have received little attention. In this paper, we propose a hierarchical group PAKE protocol $n$PAKE$^+$ protocol under the setting where each party shares an *independent* password with a trusted server. The $n$PAKE$^+$ protocol is a novel combination of the hierarchical key tree structure and the password-based Diffie-Hellman exchange, and hence it achieves substantial gain in computation efficiency. In particular, the computation cost for each client in our protocol is only $O(\log n)$. Additionally, the hierarchical feature of $n$PAKE$^+$ enables every subgroup obtains their own subgroup key in the end. We also prove the security of our protocol under the random oracle model and the ideal cipher model.

## 1 Introduction

Low-entropy and human-memorable passwords are widely used for user authentication and secure communications in real applications, e.g. internet banking and remote user access, due to their user friendliness and low deployment cost. The problem of strong authentication and key exchange between two parties sharing a password, referred to as the two-party *password-authenticated key exchange* (2PAKE) problem, has been well studied and many solutions have been proposed in the literature. With proliferation of group-oriented applications, e.g. teleconferencing, collaborative workspaces, there is an increasing need for group PAKE protocols to protect communications for a group of users.

In group-oriented communications, either the group shares a single password, or each client in the group shares an independent password with a trusted server. The single-password setting is not preferable in real applications for several reasons. First, if a client in the group leaves or the password of a client is compromised, the shared password has to be updated, which could be a very expensive process. Moreover, compromise of any client leads to breakdown of the entire system. Secondly, individual client identification is impossible in this setting. As a result, no one is able to distinguish one client from another, and it is impossible for a subset of the group to securely establish a session key and hence have secure communications. It is easy to see that the independent-password setting avoids the above problems and reflects more accurately what is happening in the real world.

S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 31–43, 2007.

Group PAKE protocols in the independent-password setting need more careful treatment since they suffer from attacks which are not present in the single password setting, such as attacks initiated by legitimate clients against other clients' passwords (e.g. [35]). Not only group PAKE protocols should be resistant to outsider attacks, but they should also be secure against insider attacks.

In this paper, we propose an efficient group PAKE protocol, referred to as $n$PAKE$^+$ protocol, for the independent-password setting. By employing a Diffie-Hellman key tree in group key establishment, the protocol achieves group key establishment and mutual authentication with only three message flows, and every client needs only to perform $5 + \lceil \log n \rceil$ exponentiations.

The remainder of the paper is organized as follows. In the next section, we discuss related work on PAKE protocols. Then we present our $n$PAKE$^+$ protocol, followed by the security proof. We analyze the performance of the proposed protocol and draw our concluding remarks at the end.

## 2 Related Work

Two-party PAKE (2PAKE) protocols were first studied by Bellovin and Merritt [5,6]. Since then, 2PAKE has been intensively investigated in the literature, see for examples [3,4,7,17,18,19,20,16,15,26,27,39]. Among them, the proposals in [4,7,16,19] are proven to be secure with formal treatment.

Some efforts were spent to extend 2PAKE protocols to the three party setting [31,25,13] where two clients each share an independent password with a trusted server. These protocols, referred to as the 2PAKE$^+$ protocols, establish a session key between two clients with the help of a trusted server. However, straightforward extension from 2PAKE protocols to 2PAKE$^+$ ones often leads to insecure designs since the latter are susceptible to more attacks, such as insider attacks. The 2PAKE$^+$ protocol presented in [31] was shown to suffer from a dictionary attack [24].

Though group PAKE protocols have important applications, they only received limited attention. Under the single-password setting, Asokan and Ginzboorg [2] proposed a group PAKE protocol for ad hoc networks, but its security is not formally proved. Bresson *et al.* [8] proposed a password-based group Diffie-Hellman PAKE protocol and proved its security formally. Schemes proposed in [23,36,14] are actually password-based version of the famous BD protocol [10]. By means of broadcast (or multicast), these protocols can achieve group key establishment in 3 rounds using a single password, just like the BD protocol.

As discussed earlier, group PAKE protocols in the single password setting are too restrictive and are expected to have limited applications in practice. To our best knowledge, there are only two schemes in the independent-password setting by Byun et al. [11,12]. However, the protocols EKE-U and EKE-M in [11] have been showed to be insecure against off-line dictionary attacks and undetectable on-line password guessing attacks [35]. The scheme in [12] is also insecure against undetectable on-line guessing attacks.

## 3 Model

In this section, we prove security of the proposed group password-authenticated key agreement protocol. We first define a model of $n$PAKE$^+$ based on the random oracle and ideal cipher model. and it is based on that of [8,9]. In this model, entities are modeled as oracles and attacks against the protocol is modeled as queries to these oracles. We prove that the protocol is secure under the random oracle model and ideal-cipher model, assuming intractability of the computational Diffie-Hellman problem.

### 3.1 Security Model

**Players.** Players in the model includes a server $S$ and a set of clients $\mathbb{C}$ comprising clients $C_1, C_2, ..., C_n$. Each player participates in some distinct and possibly concurrent executions of the protocol, and each instance of their participation is modeled as an oracle. The $j$-th instance of the server is modeled as $S^j$, and the $t_i$-th instance of $C_i$ is modeled as $C_i^{t_i}$, where $1 \leqslant i \leqslant n$ and $j, t_i \in \mathbb{N}$.

Each client obtains its distinct password $p_i$ from a dictionary $\mathbb{D}$ containing $N$ low-entropy passwords, and shares it with the server. The password is randomly chosen from the dictionary $\mathbb{D}$ with a uniform distribution.

The protocol $n$PAKE$^+$ comprises two algorithms:

- **PwdChoose**($\mathbb{D}$): a probabilistic password choosing algorithm which chooses a different password $p_i$ uniformly distributed in the dictionary $\mathbb{D}$ for each client $C_i$.
- **GrpKeyAgrmt**($S, \mathbb{C}$): the group key agreement algorithm which involves the server $S$ and clients from $\mathbb{C}$ produces a group session key for each client.

**Queries.** The adversary $\mathcal{A}$ can attack the protocol by making the following queries to the participants:

- **Execute**($S^j$, $C_1^{t_1}, ..., C_n^{t_n}$): this query models passive attacks, in which the adversary $\mathcal{A}$ makes clients and the server to execute the protocol. The adversary can eavesdrop messages exchanged between all participants.
- **Send**($C_i^{t_i}$, $m$): this query models the adversary $\mathcal{A}$ sends a message $m$ to the $t_i$-th instance of a client $C_i$. $\mathcal{A}$ then gets the output of oracle $C_i^{t_i}$ after it processes $m$ according to the protocol $n$PAKE$^+$.
- **Send**($S^j$, $m$): this query models the adversary $\mathcal{A}$ sends a message $m$ to an instance of the server $S$. $\mathcal{A}$ then gets the output of oracle $S^j$ after it processes $m$ according to the protocol $n$PAKE$^+$.
- **Reveal**($C_i^{t_i}$), **Reveal**($S^j$): These two queries model compromise of the session key derived by clients and the server. This query is only valid when the clients and the server hold a session key or are able to compute the session key.
- **Corrupt**($C_i^{t_i}$), **Corrupt**($S^j$): These two queries model compromise of the long-term passwords $p_i$. The adversary $\mathcal{A}$ gets $p_i$ by asking such a query, but he does not get any internal data of the instance being queried.

– **Test**$(C_i^{t_i})$: This query models the semantic security of the group session key. This query can be asked only once and only if the queried oracle is fresh. This query is answered as follows: one flips a coin $b$ and responses with **Reveal**$(C_i^{t_i})$ if $b = 1$ or a random value if $b = 0$.

During the execution of the protocol, the adversary $\mathcal{A}$ tries to defeat the protocol $n$PAKE$^+$ by invoking the above queries. This execution is referred to as a game $\mathbf{Game}^{\mathbf{ake}}(nPAKE^+, \mathcal{A})$. The game runs as follows:

– **PwdChoose**$(\mathbb{D})$ is run to choose a password $p_i$ for the client $C_i$.
– Set each participant's group session key as null.
– Run the adversary $\mathcal{A}$ and answer queries made by $\mathcal{A}$.
– Adversary $\mathcal{A}$ outputs a guess $b'$ for the bit $b$ in the **Test** query.

**Security Notion**

– **Freshness**. An instance of the participant (i.e. an oracle) is said to be fresh if its session key is not corrupted, which means the oracle and its partners are not asked of a **Reveal** query.
– **AKE Security.** Depend on whether *Corrupt* query is available to the adversary, AKE security can be defined into two types, AKE security with (AKE-FS) and without (AKE) forward secrecy. The AKE(resp. AKE-FS) security is defined as the advantage of an adversary $\mathcal{A}$ winning the game $\mathbf{Game}^{\mathbf{ake}}(n\text{PAKE}, \mathcal{A})$(resp. $\mathbf{Game}^{\mathbf{ake-fs}}(n\text{PAKE}, \mathcal{A})$). We say that $\mathcal{A}$ wins if he correctly guess the bit $b$ in the **Test** query in the game $\mathbf{Game}^{\mathbf{ake}}(nPAKE^+, \mathcal{A})$ (resp. $\mathbf{Game}^{\mathbf{ake-fs}}(n\text{PAKE}, \mathcal{A})$). The advantage of the adversary winning the game is $\mathsf{Adv}^{ake}_{nPAKE^+}(\mathcal{A})=2 \cdot Pr[b = b'] - 1$ (resp. $\mathsf{Adv}^{ake-fs}_{nPAKE^+}(\mathcal{A})=2 \cdot Pr[b = b'] - 1$), where the probability space is over all random coin tosses.
  The protocol $n$PAKE$^+$ is said to be **AKE-Secure**(resp. **AKE-FS-Secure**) if the adversary's advantage is negligible in the security parameter.
– **Authentication.** The probability of the adversary $\mathcal{A}$ successfully impersonating a client or a server is denoted as $\mathsf{Succ}^{auth}_{nPAKE^+}(\mathcal{A})$. The protocol $n$PAKE$^+$ is said to be **Auth-Secure** if $\mathsf{Succ}^{auth}_{nPAKE^+}(\mathcal{A})$ is negligible in the security parameter.

## 4 The $n$PAKE$^+$ Protocol

In this section, we present a group PAKE protocol, referred to as $n$PAKE$^+$ protocol, for the independent-password setting. They agree on two large primes $p$ and $q$ with $p = 2q+1$, a subgroup $\mathbb{G}$ of $Z_p^*$, a generator $g$ of $\mathbb{G}$ and a cryptographic secure keyed hash function $\mathcal{H}(\cdot)$. Notations used in the description of the protocol are given in Table 1.

### 4.1 The Diffie-Hellman Key Tree

Key graphs are extensively used in non-password based group key agreement protocols to achieve great efficiency in both computation and communications.

**Table 1.** Notations

| | |
|---|---|
| $C_i$ | The $i$-th client, $i = 1, 2, ..., n$ |
| $S$ | The trusted server |
| $p_i$ | The password shared between $C_i$ and $S$ |
| $p, q$ | Two large primes with $p = 2q + 1$ |
| $\mathbb{G}, g$ | The subgroup of order $q$ in $Z_p^*$ and its generator, respectively |
| $\mathcal{H}(\cdot)$ | A secure hash function mapping $\{0,1\}^*$ to $\{0,1\}^{len}$ |
| $K_i, BK_i$ [1] | The secret key and blinded key for client $C_i, i = 1, 2, ..., n$ |
| $\langle l, v\rangle$ | The $v$-th node at the $l$-th level on the binary key tree (Fig. 1) |
| $K_{\langle l,v\rangle}, BK_{\langle l,v\rangle}$ | The secret key and blinded key for node $\langle l, v\rangle$ |
| $SK_i$ | The session key shared between $C_i$ and $C_{i+1}, i = 1, 2, ..., n-1$ |

[1] They are interchangeable with $K_{\langle l,v\rangle}, BK_{\langle l,v\rangle}$ if $C_i$ is located at $\langle l, v\rangle$ on the key tree.

Wong et al. [38] and Wallner et al. [37] are the first to introduce the concept of key graph, called the *Logical Key Hierarchy*(LKH), to improve efficiency in group key management. The *One-way Function Tree*(OFT) proposed by McGrew and Sherman [28] improves the hierarchical tree approach further. In OFT, the key of a parent is derived from the keys of its children, and hence it reduces the size of the rekeying messages to half of that of LKH. Based on the key tree, some group key agreement proposals [30,10,33,34,29,22] use the Diffie-Hellman exchange technique in group key establishment.

The Diffie-Hellman key tree used in our protocol is a binary tree in which each leaf represents a group member. Every interior node of the key tree has exactly two children and is not associated with any group member. An example of the key tree used in our protocol is shown in Fig. 1. The nodes are denoted $\langle l, v\rangle$, where $0 \leqslant v \leqslant 2^l - 1$ since each level $l$ hosts at most $2^l$ nodes (the root is at the 0-th level). For any interior node $\langle l, v\rangle$, its left child and right child are denoted $\langle l+1, 2v\rangle$ and $\langle l+1, 2v+1\rangle$ respectively. Each node $\langle l, v\rangle$ on the key tree is associated with a secret key $K_{\langle l,v\rangle}$ and a corresponding blinded key $BK_{\langle l,v\rangle}$ computed as $g^{K_{\langle l,v\rangle}} \mod p$. The secret key $K_{\langle l,v\rangle}$ at a leaf node $\langle l, v\rangle$, which is associated with a client $C_i$, is constructed between the client $C_i$ and the server $S$ in our protocol. While the secret key of an interior node is derived from the keys of the interior node's two children by the Diffie-Hellman computation. The corresponding blinded key is then computed following the formula $BK_{\langle l,v\rangle} = g^{K_{\langle l,v\rangle}} \mod p$. Specifically, the secret key and the blinded key of an interior node $\langle l, v\rangle$ are computed recursively as follows:

$$\begin{aligned} K_{\langle l,v\rangle} &= \mathcal{H}(g^{K_{\langle l+1,2v\rangle} K_{\langle l+1,2v+1\rangle}} \mod p) \\ &= \mathcal{H}((BK_{\langle l+1,2v\rangle})^{K_{\langle l+1,2v+1\rangle}} \mod p) \\ &= \mathcal{H}((BK_{\langle l+1,2v+1\rangle})^{K_{\langle l+1,2v\rangle}} \mod p), \\ BK_{\langle l,v\rangle} &= g^{K_{\langle l,v\rangle}} \mod p. \end{aligned} \tag{1}$$

Note that if a client $C_i$ is located at the leaf node $\langle l, v\rangle$ on the key tree, then its secret key and blinded key $K_{\langle l,v\rangle}, BK_{\langle l,v\rangle}$ are also denoted as $K_i$ and $BK_i$

respectively. These two types of denotations (see Fig. 1) are interchangeable for a client $C_i$ at a leaf node $\langle l, v\rangle$.

Therefore, computing a secret key at $\langle l, v\rangle$ requires the knowledge of the key of one child and the blinded key of the other child. The secret key $K_{\langle 0,0\rangle}$ at the root node is the group key which is known only to the group members.

In order to compute the group key, a client $C_i$ needs to know a set of blinded keys, which form a set called the co-path. With the blinded keys in the co-path, the client $C_i$ can compute a set of keys from itself to the root of the key tree and these keys form another set called key-path. For the client $C_i$ located at a leaf node $\langle l, v\rangle$, we denote its key-path as $KP_i$ or $KP_{\langle l,v\rangle}$, its co-path as $CP_i$ or $CP_{\langle l,v\rangle}$. On the key tree, the key path $KP_i$ is a path from $C_i$ itself to the root node ($\langle 0, 0\rangle$) of the key tree. While the co-path $CP_i$ is formed from all the nodes that are directly connected with the key-path $KP_i$ on the key tree. The key-path $KP_i$ splits the co-path $CP_i$ into two halves: $R_i$ on the right side and $L_i$ on the left side.

For example, in Fig. 1 the client $C_2$'s key-path is $KP_2 = KP_{\langle 3,1\rangle} = \{K_{\langle 3,1\rangle}, K_{\langle 2,0\rangle}, K_{\langle 1,0\rangle}, K_{\langle 0,0\rangle}\}$, and its co-path is $CP_2 = CP_{\langle 3,1\rangle} = \{BK_{\langle 3,0\rangle}, BK_{\langle 2,1\rangle}, BK_{\langle 1,1\rangle}\}$. The key-path $KP_2$ is a path from $C_2$ (or $\langle 3, 1\rangle$) to the root of the key tree. Each node from the co-path $CP_2$ is directly connected with the key-path $KP_2$ on the key tree. The co-path $CP_2$ is split into two halves by the key-path $KP_2$: $R_2 = \{BK_{\langle 2,1\rangle}, BK_{\langle 1,1\rangle}\}$, and $L_2 = \{BK_{\langle 3,0\rangle}\}$.

The following two properties of the key tree are important for group key agreement in our protocol:

- For any binary Diffie-Hellman key tree with $n$ leaves labeled from $C_1$ to $C_n$, client $C_i$ can compute $L_{i+1}$ using $L_i$, $K_i$, and $\{BK_j : 1 \le j \le n\}$. Similarly, $C_i$ can compute $R_{i-1}$ using $R_i$, $K_i$, and $\{BK_j : 1 \le j \le n\}$.
- For any binary Diffie-Hellman key tree with $n$ leaves labeled from $C_1$ to $C_n$, client $C_i$ can compute the group key using $L_i$, $R_i$, and $K_i$.

With all the blinded keys of its co-path, a client $C_i$ can compute all the keys along the key-path, including the group secret $K_{\langle 0,0\rangle}$. For the example in Fig. 1,

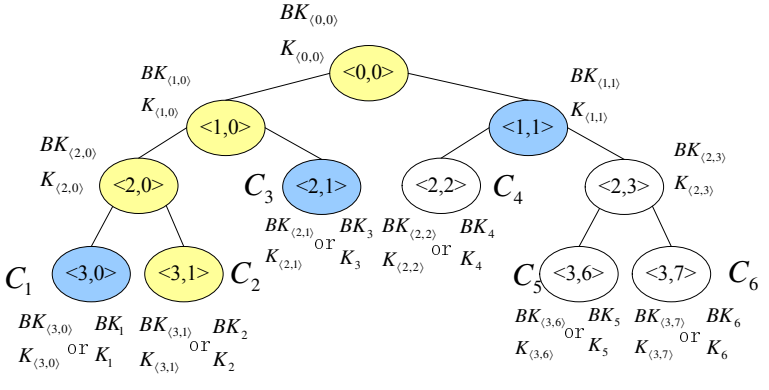


**Fig. 1.** An example of the key tree

with its own key $K_{\langle 3,1\rangle}$, $C_2$ can compute $K_{\langle 2,0\rangle}$, $K_{\langle 1,0\rangle}$ and $K_{\langle 0,0\rangle}$ using $BK_{\langle 3,0\rangle}$, $BK_{\langle 2,1\rangle}$ and $BK_{\langle 1,1\rangle}$, respectively.

### 4.2 Description of the Protocol

After introducing the Diffie-Hellman key tree, we describe our $n$PAKE$^+$ protocol in this section. Our protocol achieves group key establishment and authentication with 3 message flows. The first flow starts from the client $C_1$, traverses through $C_2, C_3, ..., C_n$, and finally reaches the server $S$. The second flow initiated by the server propagates in the reverse direction from $S$ until $C_1$. After the second flow terminates at $C_1$, $C_1$ starts the third flow towards $C_n$ and terminates at $S$.

– **Flow 1:** As the initiator, client $C_1$ chooses $r_1 \in_R \mathbb{Z}_q^*$ and computes $X_1 = g^{r_1}$ and $X_1^* = \mathcal{E}_{p_1}(X_1)$. Then it initiates the protocol by sending the request $\{C_i\}_{i=1}^n | X_1^*$ to the next client. The request traverses all the clients from $C_1$ to $C_n$ until it reaches the server. Upon receiving the request, each client $C_i$ selects $r_i \in_R \mathbb{Z}_q^*$, computes $X_i = g^{r_i}$ and the encrypted exponential $X_i^* = \mathcal{E}_{p_i}(X_i) = \mathcal{E}_{p_i}(g^{r_i})$ and adds it to the request. When the request finally reaches the server $S$, it consists of $n$ identities and $n$ encrypted exponentials contributed by the $n$ clients.

$$\begin{array}{lll} C_i \longrightarrow C_{i+1} & : & \{C_j\}_{j=1}^n | \{X_j^*\}_{j=1}^i, \quad i = 1, 2, ..., n-1, \\ C_n \longrightarrow S & : & \{C_j\}_{j=1}^n | \{X_j^*\}_{j=1}^n. \end{array} \tag{2}$$

– **Flow 2:** The second message flow runs in the reverse direction, from the server $S$ to $C_1$. After receiving the request in the first message flow, the server parses $\boldsymbol{\{C_i\}_{i=1}^n | \{X_i^*\}_{i=1}^n}$ , and uses the corresponding passwords to decrypt $X_i^*$ to obtain $X_i = g^{r_i} (i = 1, 2, ..., n)$. Then for each client $C_i (i = 1, 2, ..., n)$, $S$ chooses $s_i \in_R \mathbb{Z}_q^*$ and computes a session key $K_i = (X_i)^{s_i} = (g^{r_i})^{s_i}$. Then the server computes $Y_i = g^{s_i}$, $Y_i^* = \mathcal{E}_{p_i}(Y_i)$, $\pi = BK_1 | C_1 | \cdots | BK_n | C_n$ and $\tau_i = \mathcal{H}(\pi | X_i | Y_i | K_i)$, and sends $\boldsymbol{\pi | \{Y_j^* | \tau_j\}_{j=1}^n}$ to $C_n$.

The reply originated from the server $S$ passes through $C_n$ to $C_1$. Upon receiving the reply, $C_i (i = n, n-1, ..., 1)$ parses it as $\boldsymbol{\pi | \{Y_j^* | \tau_j\}_{j=1}^i | R_i | \xi_i}$, (for $i = n$, $R_n | \xi_n = nil$). $C_i$ decrypts $Y_i^*$ to obtain $Y_i = g^{s_i}$ using its password. Then the client computes the session key $K_i = (Y_i)^{r_i} = (g^{s_i})^{r_i}$ and the blinded $BK_i = g^{K_i}$, and verifies whether the computed $BK_i$ equals to $BK_i$ in $\pi$. Then $C_i$ verifies the validity of $\pi$ by checking $\mathcal{H}(\pi | X_i | Y_i | K_i) \stackrel{?}{=} \tau_i$. In the case where $i \neq n$, $C_i$ also computes $SK_i = (BK_{i+1})^{K_i}$ and verifies $R_i$ by checking whether $\mathcal{H}(R_i | SK_i)$ equals $\xi_i$.

If the reply passes all verifications, $C_i (i = n, n-1, ..., 2)$ prepares an outgoing message for the next client $C_{i-1}$. $C_i$ computes $R_{i-1}$ with $R_i$, $K_i$ and $\pi$, and computes $SK_{i-1} = (BK_{i-1})^{K_i}$. Then he computes $\xi_{i-1} = \mathcal{H}(R_{i-1} | SK_{i-1})$ and sends $\boldsymbol{\pi | \{Y_j^* | \tau_j\}_{j=1}^{i-1} | R_{i-1} | \xi_{i-1}}$ to $C_{i-1}$.

$$\begin{array}{lll} S \longrightarrow C_n & : & \pi | \{Y_j^* | \tau_j\}_{j=1}^n, \\ C_i \longrightarrow C_{i-1} & : & \pi | \{Y_j^* | \tau_j\}_{j=1}^{i-1} | R_{i-1} | \xi_{i-1}, \quad i = n, ..., 2. \end{array} \tag{3}$$

where $\pi = BK_1 | C_1 | \cdots | BK_n | C_n$.

– **Flow 3:** When the reply in the second message flow finally reaches $C_1$, $C_1$ performs the verifications as specified in Flow 2. If the verifications are successful, $C_1$ computes the group key $GK_1$ with $R_1$ and $K_1$ as well as $\pi$. Then $C_1$ computes $L_2$, $\sigma_1 = \mathcal{H}(L_2|SK_1)$, $\eta_1 = \mathcal{H}(C_1|C_2|\cdots|C_n|K_1)$, and starts the last message flow by sending out $L_2|\sigma_1|\eta_1$ to $C_2$.

Then each client $C_i (i = 2, 3, ..., n)$ receives the message $L_i|\sigma_{i-1}|\{\eta_j\}_{j=1}^{i-1}$, and verifies $L_i$ by checking $\sigma_{i-1} \stackrel{?}{=} \mathcal{H}(L_i|SK_{i-1})$. If the verification is successful, the client computes the group key $GK_i$ with $K_i$, $L_i, R_i$ and $\pi$. If $i \neq n$, $C_i$ computes $\sigma_i = \mathcal{H}(L_{i+1}|SK_i)$, computes $L_{i+1}$ from $L_i, K_i$ and $\pi$, computes $\eta_i = \mathcal{H}(C_1|C_2|\cdots|C_n|K_i)$, and sends the outgoing message $L_{i+1}|\sigma_i|\{\eta_j\}_{j=1}^{i}$ to $C_{i+1}$. Otherwise, $C_n$ computes $\eta_n$ and sends $\{\eta_j\}_{j=1}^{n}$ to the server $S$.

$$\begin{array}{lll} C_i \longrightarrow C_{i+1} : & L_{i+1}|\sigma_i|\{\eta_j\}_{j=1}^{i}, & i = 1, ..., n-1. \\ C_n \longrightarrow S : & \{\eta_j\}_{j=1}^{n} & \end{array} \tag{4}$$

After the third message flow finally reaches the server $S$, the server verifies each $\eta_i$ from client $C_i$ to authenticate each client. If any verification is failed, then the server can identify which client(s) is(are) invalid and not authenticated. This measure is intended to thwart on-line password guessing attacks.

After the last flow reaches the server, each client has already computed its $L_i$ and $R_i$, so each client obtains its co-path $CP_i = L_i \cup R_i$, independently calculates the same group key $K_{\langle 0,0\rangle}$ and uses it for secure group communications.

## 5 Security Results

To prove the security of the proposed protocol $n$PAKE$^+$, we incrementally define a series of games starting from the real protocol $n$PAKE$^+$ $\mathbf{G_0}$ until game $\mathbf{G_8}$. The probability of the adversary $\mathcal{A}$ winning a game $\mathbf{G_m}$ is obtained according to a negligible probability difference from the winning probability in the next game $\mathbf{G_{m+1}}$. Game $\mathbf{G_8}$ is a purely random game and the winning probability of $\mathcal{A}$ is 1/2. Therefore, we prove security of the protocol by showing the advantage of $\mathcal{A}$ in $\mathbf{G_0}$, i.e. the real protocol $n$PAKE$^+$, is negligible.

**Theorem 1.** *Let $nPAKE^+$ be the password-based group key agreement protocol with a password space $\mathbb{D}$ of size $N$. Let $\mathcal{A}$ be the adversary against* **AKE-Security** *of $nPAKE^+$ within a time bound $t$, with $q_s$ interactions with the protocol participants and $q_p$ passive eavesdroppings, $q_h$ hash queries, and $q_e$ encryption/decryption queries. Let $\mathsf{Succ}_{\mathbb{G}}^{\mathsf{CDH}}(T)$ be the success probability against the $\mathsf{CDH}$ problem of an adversary in time $T$. Then we have:*

$$\mathsf{Adv}_{nPAKE^+}^{ake}(\mathcal{A}) \leqslant 24n * q_h \mathsf{Succ}_{\mathbb{G}}^{\mathsf{CDH}}(t') + \frac{6q_s}{N} + \frac{4q_s + q_h^2}{2^{len}} + \frac{(2q_e + 3q_s + 6nq_p)^2}{q-1},$$

*where $t' = t + (q_s + (8n + n\log n) * q_p + q_e + n)\tau_{\mathbb{G}}$, with $\tau_{\mathbb{G}}$ being the computation time for an exponentiation in $\mathbb{G}$, $n$ being the maximum number of clients in all protocol executions.*

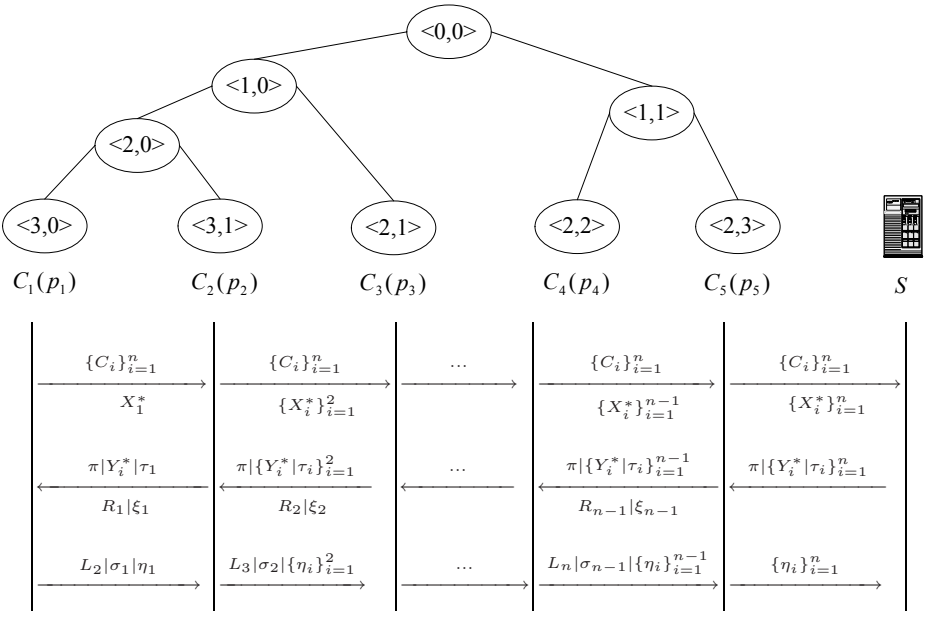


**Fig. 2.** The $n$PAKE$^+$ protocol with $n$ clients and a server illustrated with a sample 5-leaf key tree: $X_i^* = \mathcal{E}_{p_i}(X_i) = \mathcal{E}_{p_i}(g^{r_i})$; $Y_i^* = \mathcal{E}_{p_i}(Y_i) = \mathcal{E}_{p_i}(g^{s_i})$; where $r_i, s_i \in_R \mathbb{Z}_q^*$. $\pi = BK_1|C_1|BK_2|C_2|...|BK_n|C_n$; $K_i = g^{r_i s_i}$. $\tau_i = \mathcal{H}(\pi|X_i|Y_i|K_i)$; $\xi_i = \mathcal{H}(R_i|SK_i)$; $\eta_i = \mathcal{H}(C_1|C_2|...|C_n|X_i|Y_i|K_i)$; $\sigma_i = \mathcal{H}(L_{i+1}|SK_i)$.

*Proof.* Due to lack of space, we can only sketch the proof process. We define a series of games in which a simulator simulates the protocol $n$PAKE$^+$ and provides oracle queries to the attacker. The first game is the real protocol $n$PAKE$^+$, while in the last game each client obtains a random group session key so that the attacker's advantage against the last game is 0. By embedding a CDH instance into the game and using random self-reducibility, we can calculate probability differences of the attacker winning different games. Finally, we can obtain the attacker's advantage against the real protocol $n$PAKE$^+$ under the random oracle and ideal cipher model, which is related to the attacker's advantage against the CDH problem.

The theorem essentially shows that the $n$PAKE$^+$ protocol is secure against the dictionary attacks as the adversary's advantage against the protocol is constrained by the number of **Send**-queries, which represents the number of interactions with a client or a server. Normally the number of online guessing failures is restricted in existing applications, which ensures security of the proposed protocol. On the other hand, the adversary's advantage with offline dictionary attacks is proportional to its capability in solving the CDH problem. Under the assumption of hardness of the CDH problem, the protocol is secure. It is worth to note that the security of $n$PAKE$^+$ relies only on the CDH hardness assumption, while other protocols requires also the TGCDH (Trigon Group CDH) assumption and the MDDH (Multi-DDH) assumption [8,11].

**Theorem 2.** *Let $\mathcal{A}$ be the adversary against* **Auth-Security** *of $nPAKE^+$ within a time bound $t$, with $q_s$ interactions with the protocol participants and $q_p$ passive eavesdroppings, $q_h$ hash queries, and $q_e$ encryption/decryption queries. Let $\mathsf{Succ}_{\mathbb{G}}^{\mathsf{CDH}}(T)$ be the success probability against the* CDH *problem of an adversary in time $T$. Then we have:*

$$\mathsf{Succ}_{nPAKE^+}^{auth}(\mathcal{A}) \leqslant 11n * q_h \mathsf{Succ}_{\mathbb{G}}^{\mathsf{CDH}}(t') + \frac{3q_s}{N} + \frac{4q_s + q_h^2}{2^{len+1}} + \frac{(2q_e + 3q_s + 6nq_p)^2}{2(q-1)},$$

*where $t' = t + (q_s + (8n + n\log n) * q_p + q_e + n)\tau_{\mathbb{G}}$, with $\tau_{\mathbb{G}}$ being the computation time for an exponentiation in $\mathbb{G}$, $n$ being the maximum number of clients in all protocol executions.*

This theorem states that the adversary's advantage of breaking the authentication property is proportional to the number of **Send** queries, which is the number of online attacking attempts of the adversary. Same as in Theorem 1, the advantage of the adversary by offline dictionary attacks is negligible assuming hardness of the CDH problem.

**Theorem 3.** *Let $\mathcal{A}$ be the adversary against* **AKE-FS-Security** *of $nPAKE^+$ within a time bound $t$, with $q_s$ interactions with the protocol participants and $q_p$ passive eavesdroppings, $q_h$ hash queries, and $q_e$ encryption/decryption queries. Let $\mathsf{Succ}_{\mathbb{G}}^{\mathsf{CDH}}(T)$ be the success probability against the* CDH *problem of an adversary in time $T$. Then we have:*

$$\mathsf{Adv}_{nPAKE^+}^{ake-fs}(\mathcal{A}) \leqslant 2(10 + (q_s + q_p)^{(n+1)}) \cdot nq_h \mathsf{Succ}_{\mathbb{G}}^{\mathsf{CDH}}(t') + \frac{6q_s}{N}$$
$$+ \frac{4q_s + q_h^2}{2^{len}} + \frac{(2q_e + 3q_s + 6nq_p)^2}{q-1},$$

*where $t' = t + (q_s + (8n + n\log n) * q_p + q_e + n)\tau_{\mathbb{G}}$, with $\tau_{\mathbb{G}}$ being the computation time for an exponentiation in $\mathbb{G}$, $n$ being the maximum number of clients in all protocol executions.*

With this theorem, we state that the protocol is secure with forward secrecy. In case of password compromise, *fresh* session keys are still secure against dictionary attacks. This makes the protocol secure against valid-but-curious clients interested in knowing other clients' passwords. A valid-but-curious client is prevented from knowing keys of a group of which he is not a member, and a compromised password cannot be used to gain non-negligible advantage in breaking *fresh* keys.

## 6 Discussion

Under the independent password setting, our protocol is both flexible and efficient in communications and computation. First, our protocol accommodates formations of secure subgroups. Any subgroup of the whole group can run the protocol to establish a group key for the subgroup. Secondly, the protocol employs key tree in group key construction to provide communication and computation efficiency.

In the $n$PAKE$^+$ protocol, the group key computation is closely related to tree structure. By default, the key tree is formed to be a balanced binary tree to reduce the computation cost to minimal. Alternatively, the first client (the initiator) or the server can decide the tree structure. This key structure information should be protected from manipulation. Either it is agreed upon via an out-of-band channel, or it is authenticated during the protocol.

The hierarchical structure of the protocol has a good feature that a subgroup of clients corresponding to a subtree of whole key tree. For each internal node of the key tree, its key can be used to secure communications among the clients that are its descendants. Therefore, a tree structure can be decided so that the clients requiring a separate key is allocated to the same subtree of the key tree.

The protocol needs only three message flows to establish the group key, and each client needs only $5 + \lceil \log n \rceil$ exponentiations while the server needs $3n$ exponentiations. A comparison on computation cost between the protocol by Bresson *et al.* [8], EKE-U [11] and our protocol is given in Table 2. Both Bresson's protocol and EKE-U require $O(n)$ exponentiations for each client on the average, while our protocol requires only $O(\log n)$ for each client. And the total number of exponentiations required in our protocol $O(n \log n)$ is also lower than $O(n^2)$ in Bresson's protocol and EKE-U.

**Table 2.** Computation Efficiency Comparison: Number of Exponentiations

| | Client (Avg.) | Server | Total |
|---|---|---|---|
| Bresson's Protocol | $(n+5)/2$ | - | $n(n+5)/2$ |
| EKE-U Protocol | $(n+3)/2$ | $(n+1)(n+2)/2$ | $n^2+3n$ |
| $n$PAKE$^+$ Protocol | $5+\lceil \log n \rceil$ | $3n$ | $n(8+\lceil \log n \rceil)$ |

## 7 Conclusion

In this paper, we proposed a hierarchical group password-authenticated key exchange protocol where each client shares an independent password with a trusted server. Under this independent-password setting, our protocol provides better flexibility than those protocols under the single-password setting. Moreover, the protocol employs a Diffie-Hellman key tree for group key agreement, and hence achieves great efficiency in both computation and communications. Finally, we prove its security under the random oracle and ideal cipher models, and compare its performance with existing protocols.

## Acknowledgement

This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy). Zhiguo Wan is supported in part by a research grant of the IBBT (Interdisciplinary institute for BroadBand Technology) of the Flemish Government.

## References


1. Abdalla, M., Pointcheval, D., Scalable, A.: A Scalable Password-Based Group Key Exchange Protocol in the Standard Model. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 332–347. Springer, Heidelberg (2006)
2. Asokan, N., Ginzboorg, P.: Key Agreement in Ad-hoc Networks. Computer Communications 23(18), 1627–1637 (2000)
3. Bellare, M., Rogaway, P.: The AuthA Protocol for Password-Based Authenticated Key Exchange. In: Contribution to the IEEE P1363 study group (March 2000)
4. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure Against Dictionary Attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, Springer, Heidelberg (2000)
5. Bellovin, S.M., Merritt, M.: Encrypted Key Exchange: Password Based Protocols Secure against Dictionary Attacks. In: Proceedings 1992 IEEE Symposium on Research in Security and Privacy, pp. 72–84. IEEE Computer Society Press, Los Alamitos (1992)
6. Bellovin, S.M., Merritt, M.: Augmented EncryptedKey Exchange: A Password-based Protocol Secure against Dictionary attacks and Password File Compromise. In: Proceedings of CCS 1993, pp. 244–250 (1993)
7. Boyko, V., MacKenzie, P.D., Patel, S.: Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
8. Bresson, E., Chevassut, O., Pointcheval, D.: Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, Springer, Heidelberg (2002)
9. Bresson, E., Chevassut, O., Pointcheval, D.: Security Proofs for an Efficient Password-Based Key Exchange. In: Proceedings of the 10th ACM Conference on Computer and Communications Security 2003, pp. 241–250 (2003)
10. Burmester, M., Desmedt, Y., Secure, A.: Efficient Conference Key Distribution System (extended abstract). In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, Springer, Heidelberg (1995)
11. Byun, J.W., Lee, D.H.: N-Party Encrypted Diffie-Hellman Key Exchange Using Different Passwords. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 75–90. Springer, Heidelberg (2005)
12. Byun, J.W., Lee, S.-M., Lee, D.H., Hong, D.: Constant-Round Password-Based Group Key Generation for Multi-layer Ad-Hoc Networks. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 3–17. Springer, Heidelberg (2006)
13. Byun, J.W., Jeong, I.R., Lee, D.H., Park, C.-S.: Password-Authenticated Key Exchange between Clients with Different Passwords. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 134–146. Springer, Heidelberg (2002)
14. Dutta, R., Barua, R.: Password-Based Encrypted Group Key Agreement. International Journal of Network Security 3(1), 23–34 (2006)
15. Gennaro, R., Lindell, Y.: A Framework for Password-Based Authenticated Key Exchange. In: Biham, E. (ed.) EUROCRPYT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)
16. Goldreich, O., Lindell, Y.: Session-Key Generation Using Human Passwords Only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)
17. Jablon, D.: Strong Password-Only Authenticated Key Exchange. Computer Communication Review, ACM SIGCOMM 26(5), 5–26 (1996)

18. Jablon, D.P.: Extended Password Key Exchange Protocols Immune to Dictionary Attacks. In: WETICE 1997, pp. 248–255. IEEE Computer Society, Los Alamitos (June 1997)
19. Katz, J., Ostrovsky, R., Yung, M.: Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
20. Katz, J., Ostrovsky, R., Yung, M.: Forward Security in Password-Only Key Exchange Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, Springer, Heidelberg (2003)
21. Kim, Y., Perrig, A., Tsudik, G.: Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups. In: Proceedings of CCS 2000 (2000)
22. Kim, Y., Perrig, A., Tsudik, G.: Communication-Efficient Group Key Agreement. In: Proceedings of IFIP SEC 2001 (2001)
23. Lee, S.-M., Hwang, J.Y., Lee, D.H.: Efficient Password-Based Group Key Exchange. In: Katsikas, S.K., Lopez, J., Pernul, G. (eds.) TrustBus 2004. LNCS, vol. 3184, pp. 191–199. Springer, Heidelberg (2004)
24. Lin, C.-L., Sun, H.-M., Hwang, T.: Three-party Encrypted Key Exchange: Attacks and A Solution. ACM Operating Systems Review 34(4), 12–20 (2000)
25. Lin, C.-L., Sun, H.-M., Hwang, T.: Three-party Encrypted Key Exchange Without Server Public-Keys. IEEE Communications Letters 5(12), 497–499 (2001)
26. Lucks, S.: Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys. In: Security Protocols Workshop, pp. 79–90 (1997)
27. MacKenzie, P.: The PAK suite: Protocols for Password-Authenticated Key Exchange. Submission to IEEE P1363.2, (April 2002)
28. McGrew, D., Sherman, A.: Key Establishment in Large Dynamic Groups Using One-way Function Trees. Techinical Report 0755, Network Associates, Inc (1998)
29. Perrig, A., Song, D., Tygar, D.: ELK, A New Protocol for Efficient Large-Group Key Distribution. In: Proceedings of IEEE Syposium on Security and Privacy (2001)
30. Steer, D., Strawczynski, L., Diffie, W., Wiener, M.: A Secure Audio Teleconference System. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, Springer, Heidelberg (1990)
31. Steiner, M., Tsudik, G., Waidner, M.: Refinement and Extension of Encrypted Key Exchange. ACM SIGOPS Operating Systems Review 29(3), 22–30 (1995)
32. Steiner, M., Tsudik, G., Waidner, M.: Diffie-Hellman Key Distribution Extended to Group Communication. In: Proceedings of CCS 1996 (March 1996)
33. Steiner, M., Tsudik, G., Waidner, M.: Cliques: A New Approach to Group Key Agreement. In: IEEE TPDS (August 2000)
34. Steiner, M., Tsudik, G., Waidner, M.: Key Agreement in Dynamic Peer Groups. In: IEEE Transactions on Parallel and Distributed Systems (August 2000)
35. Tang, Q., Chen, L.: Weaknesses in Two Group Diffie-Hellman Key Exchange Protocols. Cryptology ePrint Archive (2005)/197
36. Tang, Q., Choo, K.-K.: Secure Password-based Authenticated Group Key Agreement for Data-Sharing Peer-to-Peer Networks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, Springer, Heidelberg (2006)
37. Wallner, D.M., Harder, E.J., Agee, R.C.: Key Management for Multicast: Issues and Architectures, Internet Request for Comments 2627, (June 1999)
38. Wong, C.K., Gouda, M., Lam, S.: Secure Group Communications Using Key Graphs. In: Proceedings of SIGCOMM 1998 (1998)
39. Wu, T.: The Secure Remote Password Protocol. In: 1998 Internet Society Symposium on Network and Distributed System Security, pp. 97–111 (1998)