Efficient and Provably Secure Generic Construction of Client-to-Client Password-Based Key Exchange Protocol* 

Zhoujun Li

School of Computer Science and Engineering
Beihang University
Beijing, China

Hua Guo

School of Computer Science and Engineering
Beihang University
Beijing, China

Xiyong Zhang

Institute of Information Engineering
Information and Engineer University
Zhengzhou, China

Abstract

Client-to-client password authenticated key exchange (C2C-PAKE) protocol enables two clients who only share their passwords with their own servers to establish a shared key for their secure communications. Recently, Byun et al. and Yin-Li respectively proposed first provably secure C2C-PAKE protocols. However, both protocols are found to be vulnerable to undetectable online dictionary attacks and other attacks. In this paper, we present an efficient generic construction for cross-realm C2C-PAKE protocols and prove its security in the Random-or-Real model due to Abdalla et al., without making use of the Random Oracle model.

Keywords: Password-authenticated key exchange, cross realm, client-to-client, provably secure, general construction.

* Supported by the National Natural Science Foundation of China under Grant No. 60473057, 90604007, 60703075, 90718017 and Supported by The Research Fund for the Doctoral Program of Higher Education(No.20070006055)

1 Email: lizj@buaa.edu.cn
2 Email: hg999@uow.edu.au
3 Email: xzychxy3711@sina.com

1571-0661 © 2008 Elsevier B.V. Open access under CC BY-NC-ND license.
1 Introduction

Client-to-client password-authenticated key exchange protocols (C2C-PAKE) are important cryptographic techniques for secure communications. Conceptually, a typical C2C-PAKE protocol works as follows. It is required that each client should share a human-memorable password with his own trusted server. When two clients want to establish a shared session key, they resort to their own trusted server for authenticating each other. Therefore, a communicating party who wants to build secure communications with other parties does not need to remember so many passwords whose number would be large linearly in the number of all possible partners, instead it only holds a password shared with his trusted server. Due to this advantage, it has attracted a lot of attention and many C2C-PAKE protocols have been proposed \cite{1,2,3,4,5,6,7} in recent years.

Byun et al. \cite{1} first proposed a C2C-PAKE protocol in the cross-realm setting by using the key distribution centers(KDCs) in the different realms as the go-between. They have heuristically proved that the schemes were secure against all considered attacks. Such protocols are more popularly known as cross-realm C2C-PAKE protocols. For simplicity, we will call these C2C-PAKEs for the rest of this paper.

Nevertheless, most of the existing C2C-PAKE protocols were only analyzed in ad hoc without a formal security model. Hitherto, only Byun et al. \cite{6} and Yin-Li \cite{7} respectively proposed provably secure C2C-PAKE protocols, with security based on computationally intractable assumptions. However, Phan and Goi \cite{8} found that both protocols fall to undetectable online dictionary attacks by any adversary and that the protocol of Byun et al. \cite{6} can not keep the malicious servers from launching a successful man-in-the-middle attack and the Yin-Li \cite{7} scheme inherits a weakness against unknown key-share attacks.

To our knowledge, there exists no generic construction of C2C-PAKE in the cross-realm setting. Recently, Abdalla et al. \cite{9} proposed a generic method to construct provably secure single-server C2C-PAKE protocol. However, Wang and Hu \cite{10} found their scheme suffer from undetectable on-line dictionary attacks, and they introduced a new efficient generic construction scheme for the 3-party PAKE protocols. In this paper, based on Wang-Hu’s scheme we presented a new generic construction for the cross-realm C2C-PAKE protocols which is not only efficient but also resistant to both off-line and undetectable on-line dictionary attacks. Moreover, we prove its security in Abdalla et al.’s Real-or-Random(ROR) model \cite{9}.

The paper is organized as follows. In Section 2, we describe our generic construction for the cross-realm C2C-PAKE protocols. To prove its security, in Section 3, we recall the ROR model, necessary basic assumptions and the definition of security. In Section 4, we focus on the security of the new scheme and provides details of the security proof. Finally, concluding remarks are given in Section 5.
2 General Construction of C2C-PAKE Protocols

In this section, we present a generic construction for client-to-client password-based key exchange protocols (referred as C2C-GPAKE) in the scenario in which we have an honest-but-curious server. The construction could be viewed as an extension of the scheme proposed in [10], which in turn is an enhancing of Abdalla et al.’s generic construction[9] which is designed for 3-party PAKE. More precisely, we extend Wang-Hu’s scheme to two separate servers, and present the construction using a 2-party password-based key exchange and a 2-party MAC-based key exchange protocol. Similarly to the construction of Abdalla et al., the proposed scheme is essentially a form of compiler transforming any secure 2-party PAKE protocol into a secure C2C-PAKE protocol, and thus can be used to create a series of provably secure C2C-PAKE protocols.

2.1 Scheme Description

The general construction involves in four participants, denoted as A, S1, B and S2, respectively, where A is a client in the realm of server S1, B is a client in the realm of server S2. We assume that the key K is pre-distributed between S1 and S2 by using a two party key exchange protocol. The detailed steps of the C2C-GPAKE, as shown in Figure 1, are described as follows:

- **Step 1**: The users A and B establish two secure high-entropy session keys skA and skB with the trusted server S1 and S2, respectively, by using any semantic secure 2-party PAKE protocol.
- **Step 2**: Using the session keys skA and skB generated in the first step as the MAC key, A and B can concurrently authenticate and send their respective temporary Diffie-Hellman public keys to their own server S1 and S2, respectively.
- **Step 3**: Upon receiving and confirming the temporary public key from the client A, the server S1 authenticates and transfers a temporary public key of A to S2 by using the MAC scheme with the symmetrical key K between S1 and S2. Similarly, the server S2 authenticates and transfers a temporary public key of B to S1 in

---

**Fig. 1. C2C-GPAKE: a generic client-to-client password-based key exchange**
the same way.

• Step 4: Finally, \( S_1 \) and \( S_2 \) send the temporary public keys of \( B \) and \( A \) to \( A \) and \( B \), respectively. In this manner, \( A \) and \( B \) establish a session key in an authenticated way, with the cooperation of the trusted servers \( S_1 \) and \( S_2 \).

### 3 Formal Models for Cross-Realm C2C-PAKE Protocol

The first security model about key exchange protocol was proposed by Bellare and Rogaway in [11]. After that, Bellare et al. extended their model to password-based key exchange protocol [12] in 2000. Recently, Abdalla et al. provided a new and stronger security model [9] by modifying the previous one [12] slightly and called it Real-or-Random (ROR) model. They suggested to use the ROR model for proving the security of their password-based schemes. In this section, we utilize this formal model to prove the security of our generic construction for our cross-realm client-to-client password-based key exchange.

### 3.1 Communication Model

We denote \( A \), \( B \) as two clients belonging to two different realms. Client \( A \) shares his password \( pw_A \) with server \( S_1 \), and client \( B \) shares his password \( pw_B \) with another server \( S_2 \). \( S_1 \) and \( S_2 \) share the common key \( K \) which is pre-distributed between them by using a 2-party key exchange protocol. All clients’ passwords are chosen from the same small dictionary \( D \) whose distribution is \( D_{pw} \).

The generic construction of cross-realm C2C-PAKE is an interactive scheme among four participants’ instances: \( A^i \), \( B^j \), \( S^i_1 \), \( S^j_2 \). In the end, \( A^i \) and \( B^j \) establish a session key \( sk \). In the model, it is assumed that an adversary \( A \) has full control over the communication channels and can create several concurrent instances of the protocol. During the execution of the protocol, adversary \( A \) interacts with participants only via oracle queries, which model adversary’s possible attacks in the real execution. All possible oracle queries are listed in the following, where \( U^i \) (\( S^j \), respectively) denotes the \( i \)-th (\( j \)-th, respectively) instance of a participant \( U \) (\( S \), respectively):

* **Execute**(\( A^i \), \( B^j \), \( S^i_1 \), \( S^j_2 \)): This query models passive attacks in which the attacker eavesdrops on honest executions. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.

* **SendClient**(\( U^i \), \( m \)): This query models an active attack. After querying the oracle, a message \( m \) is sent to the client instance \( U^i \). Finally, client instance \( U^i \) forwards its response to \( A \).

* **SendServer**(\( S^i \), \( m \)): This query models an active attack against a server. It outputs the message that server instance \( S^i \) would generate upon receipt of message \( m \).

* **Test**(\( U^i \)): If a Test query is asked to a client instance that has not accepted, then return the undefined \( \bot \). If a Test query is asked to an instance of an honest
client whose intended partner is dishonest or to an instance of a dishonest client, then returns the real session key. Otherwise, returns either the real session key if \( b = 1 \) or a random one if \( b = 0 \), where \( b \) is the hidden bit selected at random prior to the first call.

**Partnering:** The definition of partnering uses the session identifications (\( \text{sid} \)). More specifically, two instances \( U^i \) and \( U^j \) are said to be partners if the following conditions are satisfied:

1. Both \( U^i \) and \( U^j \) accept;
2. Both \( U^i \) and \( U^j \) own the same \( \text{sid} \);
3. \( U^i \) is \( U^j \)'s partner and vice-verse; and
4. No instance other than \( U^i \) and \( U^j \) accepts with a partner identity equal to \( U^i \) or \( U^j \).

**Freshness.** An instance \( U^i \) is said to be fresh if it has accepted and no \( \text{Reveal} \) queries have been made to it or its partner.

### 3.2 Building Blocks

In our generic construction for the client-to-client PAKE protocols, two cryptographic primitives are used as building blocks: decisional Diffie-Hellman assumption and message authentication codes.

**Decisional Diffie-Hellman assumption:** DDH. The DDH assumption can be precisely defined by two experiments, \( \text{Exp}^{\text{ddh-real}}_G(\mathcal{A}) \) and \( \text{Exp}^{\text{ddh-rand}}_G(\mathcal{A}) \). An adversary \( \mathcal{A} \) is provided with \( g^x, g^y \) and \( g^{xy} \) in the former experiment, and \( g^x, g^y \) and \( g^z \) in the latter one, where \( x, y \) and \( z \) are drawn at random from \( \{1, ..., |G|\} \).

Define the advantage of \( \mathcal{A} \) in violating the DDH assumption as follows:

\[
\text{Adv}^{\text{ddh}}_G(t) = \max \{|\Pr[\text{Exp}^{\text{ddh-real}}_G(\mathcal{A}) = 1] - \Pr[\text{Exp}^{\text{ddh-rand}}_G(\mathcal{A}) = 1]|\}
\]

where the maximum is over all adversaries \( \mathcal{A} \) running in time at most \( t \). The DDH assumption in \( G \) holds if \( \text{Adv}^{\text{ddh}}_G(t) \) is a negligible function of \( t \).

**Message authentication codes.** A message authentication code scheme \( \text{MAC}=(\text{Key},\text{Tag},\text{Ver}) \) is composed of a MAC key generation algorithm \( \text{Key} \), a MAC generation algorithm \( \text{Tag} \) and a MAC verification algorithm \( \text{Ver} \). A secure MAC should prevent existential forgeries under chosen-message attacks (EUF-CMA) if adversaries has access to the generation and verification oracles. That is, it can not create a new valid message-tag pair, even after obtaining many valid message-tag pairs. The maximal value of the advantage \( \text{Adv}^{\text{euf-cma}}_{\text{MAC}}(\mathcal{A}) \) with at most \( q_g \) and \( q_v \) queries to its MAC generation and verification oracles, respectively, is a negligible function of the parameters above.

### 3.3 Security Definition

According to [7], a secure generic construction of cross-realm C2C-PAKE should satisfy the following security requirements: (1) The session key cannot be distinguished from a random number by an adversary; (2) The servers do not know the
session key between clients; (3) The client can authenticate his server and vice-verse; (4) The client does not know other client’s password; and (5) Clients’ passwords are not revealed to other servers except for their own servers. We define the following security notions:

**Semantic Security in the ROR model:** During the executing, the adversary $A$ is allowed to send multiple queries to the $Execute$, $SendClient$, $SendServer$, and $Test$ oracles as it wants, while it is no longer allowed to ask $Reveal$ queries which is allowed in the model of Abdalla et al. [9]. Notice that, when $b=0$, the same random key value should be returned for $Test$ queries that are asked to two instances which are partnered.

We say the adversary $A$ succeeds if he correctly guesses the bit $b$ hidden in the $Test$ oracle. Let $Succ$ denote the event that $A$ succeeds. Provided that passwords are drawn from dictionary $D$, we define the advantage of $A$ as:

$$Adv_{P,D}^{rOR\text{-ake}}(A) = 2 \cdot \Pr[Succ] - 1,$$

$$Adv_{P,D}^{rOR\text{-ake}}(t, R) = \max \{Adv_{P,D}^{rOR\text{-ake}}(A)\},$$

where the maximum is over all adversaries with time-complexity at most $t$ and using at most $R$ times oracle queries.

The scheme of C2C-GPAKE is said to be semantically secure if the advantage $Adv_{P,D}^{rOR\text{-ake}}(t, R)$ is only negligibly larger than $kn/|D|$, where $n$ is number of active sessions and $k$ is a constant.

**Key Privacy with respect to the server:** This security requires no information about the session key revealed to the server who knows all passwords of his members but behaves in an honest-but-curious manner. The adversary $A$ has access to all the passwords. To capture the adversary’s ability to tell apart the real session key shared between any two instances from a random one, Abdalla et al.[9] introduced a new type of oracle, called $TestPair$, defined as follows, where $b$ is a bit chosen uniformly at random at the beginning of the experiment.

$TestPair(A^i, B^j)$: If client instances $A^i$ and $B^j$ do not share the same key, then return the undefined symbol ⊥. Otherwise, return the real session key shared between $A^i$ and $B^j$ if $b=1$ or a random key of the same size if $b=0$.

During the executing, the adversary $A$ has access to the passwords of all users and multiple queries to the $Execute$, $SendClient$ and $TestPair$ oracles as it wants, and let $b_0$ be its output. Such an adversary is said to win the experiment if $b_0 = b$, where $b$ is the hidden bit used by the $TestPair$ oracle. Let $Succ$ denote the event in which the adversary guesses $b$ correctly. We can then define the kp-advantage $Adv_{P,D}^{kp\text{-ake}}(A)$ of $A$ in violating the key privacy of the key exchange protocol $P$ and the advantage function $Adv_{D}^{kp}(t, R)$ of $P$ as in previous definitions.

Finally, we say an adversary $A$ succeeds in breaking the key privacy of a protocol $P$ if its advantage $Adv_{P,D}^{kp\text{-ake}}(A)$ is non-negligible.

**Authentication Security:** Most of the existing password-based authenticated key exchange protocols are vulnerable to the undetectable on-line dictionary at-
tacks due to the absence of authentication of messages between the client and the server. In order to solve this problem, we introduce the definition of the unilateral authentication from the client to the trusted server as [10] does. We denote by $\text{Succ}_{\text{auth}}^{\text{c}\rightarrow\text{s}}(A)$ the probability that an adversary $A$ successfully impersonates a client instance during executing the protocol $P$ while the trusted server does not detect it. Further, $\text{Succ}_{\text{auth}}^{\text{c}\rightarrow\text{s}}(A) = \max\{\text{Adv}_{\text{auth}}^{\text{c}\rightarrow\text{s}}(A)\}$ is defined as the maximum over all $A$ running in time at most $t$ and using resources at most $R$. We say a scheme of C2C-GPAKE is client-to-server authentication secure if $\text{Succ}_{\text{auth}}^{\text{c}\rightarrow\text{s}}(t, R)$ is negligible in the security parameter.

**Password Protection Against Malicious Client:** The malicious client $C$ succeeds if he successfully learns another client’s password. Since Test oracle query is used to define the session key’s security, the malicious client does not have access to Test query. Let $D$ be user’s password dictionary. For any malicious client $C$, define his advantage $\text{Succ}_{D}^{\text{pw}-\text{mc}}$ as

$$\text{Adv}_{D}^{\text{pw}-\text{mc}}(C) = \Pr[\text{Succ}_{D}^{\text{pw}-\text{mc}}],$$

$$\text{Adv}_{D}^{\text{pw}-\text{mc}}(t, R) = \max\{\text{Adv}_{D}^{\text{pw}-\text{mc}}(C)\},$$

where the maximum is over all adversaries with time-complexity at most $t$ and querying oracles at most $R$ times. We say $P$ satisfies password protection against malicious client if the advantage $\text{Adv}_{D}^{\text{pw}-\text{mc}}$ is only negligibly larger than $O(q_s) \cdot D_{pw}$, where $q_s$ is the number of all send queries, $D_{pw}$ is the distribution of password dictionary.

**Password Protection Against Honest-but-Curious Server:** An honest-but-curious server $S$ succeeds if he successfully learns the passwords of the clients which belongs to other servers. For any honest-but-curious server $S$, we define his advantage $\text{Adv}_{D}^{\text{pw}-\text{ms}}(S)$ as

$$\text{Adv}_{D}^{\text{pw}-\text{ms}}(S) = \Pr[\text{Succ}_{D}^{\text{pw}-\text{ms}}],$$

$$\text{Adv}_{D}^{\text{pw}-\text{ms}}(t, R) = \max\{\text{Adv}_{D}^{\text{pw}-\text{ms}}(S)\},$$

where the maximum is over all adversaries with time-complexity at most $t$ and querying oracles at most $R$ times.

We say $P$ satisfies password protection against malicious server if the advantage $\text{Adv}_{D}^{\text{pw}-\text{ms}}$ is only negligibly larger than $O(q_s) \cdot D_{pw}$, where $q_s$ is the number of all send queries, $D_{pw}$ is the distribution of password dictionary.

### 4 Security proof

In this section, we examine all security requirements proposed in the subsection 3.2 and show they are all met.

**Semantic Security in the ROR model.** As the following theorem states, the generic scheme C2C-GPAKE is a secure client-to-client password-based key exchange protocol as long as the Decisional Diffie-Hellman assumption holds in $G$ and the underlying primitives it uses are secure.
Theorem 4.1 Let 2PAKE be a semantic secure 2-party PAKE protocol and MAC be a secure MAC algorithm. Let $q_{exe}$ and $q_{test}$ denote the numbers of queries to Execute and Test oracles, and $q_{A}^{send}$, $q_{B}^{send}$, and $q_{ake}$ be the numbers of queries to the SendClient and SendServer oracles with respect to each of the two 2PAKE protocols and the MAC-based authenticated key exchange protocols. Then,

$$Adv_{\text{ror-ake}}^{\text{2C-GPAKE,D}}(t, q_{exe}, q_{test}, q_{A}^{send}, q_{B}^{send}, q_{ake})$$

$$\leq 2 \cdot Adv_{\text{ror-ake}}^{\text{2PAKE,D}}(t, q_{exe}, q_{exe} + q_{A}^{send}, q_{A}^{send})$$

$$+ 2 \cdot Adv_{\text{ror-ake}}^{\text{2PAKE,D}}(t, q_{exe}, q_{exe} + q_{B}^{send}, q_{B}^{send})$$

$$+ 2 \cdot q_{ake} \cdot Adv_{\text{MAC}}^{\text{euf-cma}}(t, 2, 0)$$

$$+ 2 \cdot Adv_{\text{ddh}}^{\text{G}}(t + 8(q_{exe} + q_{ake}) \cdot \tau_{G})$$

where $\tau_{G}$ denotes the exponentiation computational time in $G$.

Proof. We follow the proof of Wang-Hu, which in turn is of Abdalla et al. [9]. Without loss of generality, we assume the set of honest users contains only users $A$ and $B$. It can be easily extended to the more general case. Let $A$ be an adversary against the semantic security of C2C-GPAKE in the Real-or-Random model with time-complexity at most $t$, and asking at most $q_{exe}$ queries to its Execute oracle, $q_{test}$ queries to its Test oracle, $q_{A}^{send}$, $q_{B}^{send}$ queries to SendClient and SendServer oracles corresponding to the 2PAKE protocol between $A$ and the trusted server $S_{1}$, and between $B$ and the trusted server $S_{2}$, respectively. $q_{ake}$ queries to SendClient and SendServer oracles corresponding to the authenticated key exchange protocol between $A$ and $S_{1}$, and $q_{ake}$ queries to the oracles corresponding to the protocol between $B$ and $S_{2}$. Our proof consists of a sequence of hybrid experiments, starting with the real attack against C2C-GPAKE scheme and ending in a game in which the adversary’s advantage is 0. For each game $G_{i}$, define an event $\text{Succ}_{i}$ corresponding to the case in which the adversary correctly guesses the hidden bit $b$ involved in the Test queries in game $G_{i}$.

Game $G_{0}$. This game corresponds to the real attack. By definition, we have

$$Adv_{\text{ror-ake}}^{\text{C2C-GPAKE,D}}(A) = 2 \cdot \Pr[\text{Succ}_{0}] - 1.$$ 

Game $G_{1}$. We now modify the simulation of the oracles as the proof of Wang-Hu which uses a random session key $sk_{A}'$, instead of the session key $sk_{A}$, as the MAC key in all of the sessions between $A$ and $S_{1}$. So, we have the following lemma:

Lemma 4.2 $|\Pr[\text{Succ}_{1}] - \Pr[\text{Succ}_{0}]| \leq 2 \cdot Adv_{\text{ror-ake}}^{\text{2PAKE,D}}(t, q_{exe}, q_{exe} + q_{A}^{send}, q_{A}^{send}).$

Game $G_{2}$. This game is the same as the previous one except that we replace the session key $sk_{B}$ with a random session key $sk_{B}'$ in all of the sessions between $B$ and $S_{2}$. So, we have the similar argument:

Lemma 4.3 $|\Pr[\text{Succ}_{2}] - \Pr[\text{Succ}_{1}]| \leq 2 \cdot Adv_{\text{ror-ake}}^{\text{2PAKE,D}}(t, q_{exe}, q_{exe} + q_{B}^{send}, q_{B}^{send}).$
Game $G_3$. In this game, we use a random key $K'$, instead of the key $K$ in all of the sessions between $S_1$ and $S_2$. In fact, since $K$ is pre-distributed between $S_1$ and $S_2$ by using two party key exchange protocol, we can view it as a random and independent value, so this game is equivalent to the previous one. Thus, we have $\Pr[\text{Succ}_3] = \Pr[\text{Succ}_2]$.

Game $G_4$. This game is modified as follows. If the adversary asks a $\text{SendClient}$ or $\text{SendServer}$ query for AKE between $A$ and $S_1$ involving a new pair of message tag not previously generated by an oracle, then we consider the MAC tag invalid and abort the game. So we have the following arguments:

**Lemma 4.4** $|\Pr[\text{Succ}_4] - \Pr[\text{Succ}_3]| \leq q_{\text{ake}}^{\text{AS}} \cdot \text{Adv}_{\text{MAC}}^{\text{euf-cma}}(t, 2, 0)$.

Game $G_5$. This game is the same as the previous one except that the adversary asks a $\text{SendClient}$ or $\text{SendServer}$ query for AKE between $B$ and $S_2$. Hence

**Lemma 4.5** $|\Pr[\text{Succ}_5] - \Pr[\text{Succ}_4]| \leq q_{\text{ake}}^{\text{BS}} \cdot \text{Adv}_{\text{MAC}}^{\text{euf-cma}}(t, 2, 0)$.

The following two games $G_6$ and $G_7$ is the same as the last two games of proof of Wang-Hu. So we have the following two conclusions:

$$\Pr[\text{Succ}_6] = \Pr[\text{Succ}_5] \quad \text{and}$$

**Lemma 4.6** $|\Pr[\text{Succ}_7] - \Pr[\text{Succ}_6]| \leq \text{Adv}_{\text{ddh}}^{\text{G}}(t + 8(q_{\text{exe}} + q_{\text{ake}}) \cdot \tau_e)$, where $q_{\text{ake}} = q_{\text{ake}}^{\text{AS}} + q_{\text{ake}}^{\text{BS}}$.

Since no information on the bit $b$ in the $\text{Test}$ oracle is leaked to the adversary, $\Pr[\text{Succ}_7] = 1/2$. This result combined with the previous lemmas yields the result in Theorem 4.1. □

**Key Privacy respect to Server:** An honest-but-curious server only has access to $\text{SendClient}$, $\text{Execute}$ and $\text{TestPair}$ oracles. As the following theorem states, the generic scheme C2C-GPAKE has key privacy with respect to the server as long as the DDH assumption holds in $\mathbb{G}$.

**Theorem 4.7** In our cross-realm C2C-GPAKE protocol, an honest-but-curious server cannot learn the session key between clients as long as the DDH assumption holds in the group $\mathbb{G}$. Formally, $\text{Adv}_{\text{C2C-GPAKE,D}}^{\text{kp-ake}}(t, q_{\text{exe}}, q_{\text{test}}, q_{\text{send}}^{A}, q_{\text{send}}^{B}, q_{\text{ake}}) \leq 2 \cdot \text{Adv}_{\text{ddh}}^{\text{G}}(t + 8(q_{\text{exe}} + q_{\text{ake}}) \cdot \tau_e)$ where the parameters are defined as in Theorem 4.1.

**Proof.** The proof is similar to the games $G_6$ and $G_7$ in the proof of semantic security of C2C-GPAKE. Let $A_{kp}$ be an adversary against the key privacy of C2C-GPAKE whose time-complexity is at most $t$. Moreover, $A_{kp}$ asks at most $q_{\text{exe}}$ queries to its $\text{Execute}$ oracle, $q_{\text{test}}$ queries to its $\text{TestPair}$ oracle, $q_{\text{ake}}^{\text{AS}}$ queries to $\text{SendClient}$ and $\text{SendSever}$ oracles corresponding to the authenticated key exchange protocol between $A$ and $S_1$, and $q_{\text{ake}}^{\text{BS}}$ queries to the oracles corresponding to the protocol between $B$ and $S_2$. We show that if $A_{kp}$ exists, we can construct an adversary $A_{\text{ddh}}$ to solve the DDH problem with non-negligible probability.
Given an instance of the DDH problem \((X, Y, Z)\), \(A_{ddh}\) first chooses the passwords for all users according to the distribution of \(D\), then it chooses a bit \(b\) at random used in the \(TestPair\) oracle. Now it starts running \(A_{kp}\) giving the pre-distributed key \(K\) and all the passwords of all users to it. Since \(A_{ddh}\) knows the password of all users, it can easily answer queries made by \(A_{kp}\). To deal with the security of the key privacy respect to server, we only consider the last flows of C2C-GPAKE. Like Abdalla’s et al.\([9]\) proof, here we introduce the input triple in the answers to \(SendClient\), \(Execute\), and \(TestPair\) queries by using the classical random self-reducibility of the Diffie-Hellman problem.

We simulate the \(Execute\) oracle by using the passwords that have been chosen and \(SendClient\) queries, and simulate the \(SendClient\) and \(TestPair\) as follows:

R1: When a \(SendClient(A_i, \text{Start})\) query is asked, \(A_{ddh}\) picks two random values \(a_0\) and \(x_0\) in \(Z_q\), computes \(X_0 = X^{a_0} g^{x_0}\) and stores them in a list \(\Lambda_A\). For \(SendClient(B_j, \text{Start})\) in the same session, the simulator selects \(b_0\) and \(y_0\), computes \(Y_0 = Y^{b_0}\) and stores them in a list \(\Lambda_B\) in the same measure.

R2: Upon receipt of both \(SendClient(A_i, (Y_0, m_b))\) and \(SendClient(B_j, (X_0, m_a))\) of the same session, the simulator checks the existence of \(X_0\) and \(Y_0\) by using \(\Lambda_A\) and \(\Lambda_B\), respectively. If their existence is exact, it computes \(Z_0 = Z^{a_0 b_0} \times Y^{x_0 b_0} \times X^{a_0 y_0} \times g^{x_0 y_0}\) in preparation for answering the \(TestPair\) query. Otherwise, it proceeds with the simulation as it would in a real attack.

R3: When a \(TestPair(U_{i1}, U_{i2})\) query is asked, \(A_{ddh}\) first checks whether \(U_{i1}\) and \(U_{i2}\) have both accepted and have the same key. If the check fails, then \(A_{ddh}\) returns \(\perp\). If the check passes, then \(A_{ddh}\) knows the corresponding value \(Z_0\) for the secret key and can answer it based on the hidden bit \(b\) it had previously chosen.

Let \(b_0\) be the output of \(A_{kp}\). If \(b_0 = b\), then \(A_{ddh}\) returns 1 and 0, otherwise.

As analyzed as the game \(G_6\) of the proof of Wang-Hu, we have the result of Theorem 4.7.

Authentication security: This security aims to resist the undetectable online dictionary attacks which stem from an absence of authentication of messages between the client and the server, so it has nothing to do with the messages between the servers. From this viewpoint, we can treat \(S_1\) and \(S_2\) as a single server. According to the games from \(G_0\) to \(G_5\), we have the following theorem:

**Theorem 4.8** The cross-realm C2C-GPAKE satisfies the client-to-server authentication security as long as the DDH assumption holds in \(G\) and the underlying primitives it uses are secure. Formally,

\[
\text{Adv}^{\text{auth}(C \rightarrow S)}_{\text{C2C-GPAKE},D}(t, q_{\text{exe}}, q_{\text{test}}, q_{\text{send}}^A, q_{\text{send}}^B, q_{\text{ake}}) \\
\leq \text{Adv}^{\text{r-or-ake}}_{\text{2PAKE},D}(t, q_{\text{exe}}, q_{\text{exe}} + q_{\text{send}}^A) \\
+ \text{Adv}^{\text{r-or-ake}}_{\text{2PAKE},D}(t, q_{\text{exe}}, q_{\text{exe}} + q_{\text{send}}^B) \\
+ q_{\text{ake}} \cdot \text{Adv}^{\text{euf-cma}}_{\text{MAC}}(t, 2, 0)
\]
where the parameters are defined as in Theorem 4.1.

**Password Protection Against Malicious Client:** In the cross-realm C2C-GPAKE protocol, a malicious client may want to learn other client’s password. In our protocol, we suppose client $B$ is malicious and his goal is to learn client $A$’s password $pw_A$. This security notion is ensured by the following theorem.

**Theorem 4.9** In our cross-realm C2C-GPAKE scheme, the malicious client $B$ cannot learn the client $A$’s password as long as our cross-realm C2C-GPAKE satisfies the client-to-server authentication security. Formally,

$$Adv_{pw-mc}^{C2C-GPAKE,D}(t, q_{exe}, q_{test}, q_{send}^A, q_{send}^B, q_{ake})$$

$$\leq Adv_{2PAKE,D}^{ror-ake}(t, q_{exe}, q_{exe} + q_{send}^A, q_{send}^A)$$

$$+ Adv_{2PAKE,D}^{ror-ake}(t, q_{exe}, q_{exe} + q_{send}^B, q_{send}^B)$$

$$+ q_{ake} \cdot Adv_{MAC}^{euf-cma}(t, 2, 0) + (q_{send}^A + q_{send}^B + q_{ake})/N$$

where the parameters are defined as in Theorem 4.1.

**Proof.** Since the C2C-GPAKE satisfies the client-to-server authentication security, it can resist undetectable on-line dictionary attacks. Moreover, from the execution of the protocol, since $a$ is a random number and the 2PAKE is secure, so the malicious client $B$ thinks the values of $g^a$ and $MAC(g^a, sk_A, A, B)$ are two independent random numbers from which no information about $pw_A$ is revealed to him. As a result, the probability that client $B$ correctly guesses $pw_A$ is exactly $q_s/N$ after $q_s$ times send queries, where $N$ is the size of the dictionary and $q_s = q_{send}^A + q_{send}^B + q_{ake}$. $\square$

**Password Protection Against Malicious Server:** In our protocol, we suppose $S_2$ is malicious and his goal is to learn client $A$’s password $pw_A$. This theorem’s proof is similar to that of Theorem 4.9.

**Theorem 4.10** In our cross-realm C2C-GPAKE protocol, the malicious server $S_2$ cannot learn the client $A$’s password. Formally,

$$Adv_{pw-ms}^{C2C-GPAKE,D}(t, q_{exe}, q_{test}, q_{send}^A, q_{send}^B, q_{ake})$$

$$\leq Adv_{2PAKE,D}^{ror-ake}(t, q_{exe}, q_{exe} + q_{send}^A, q_{send}^A)$$

$$+ Adv_{2PAKE,D}^{ror-ake}(t, q_{exe}, q_{exe} + q_{send}^B, q_{send}^B)$$

$$+ q_{ake} \cdot Adv_{MAC}^{euf-cma}(t, 2, 0) + (q_{send}^A + q_{send}^B + q_{ake})/N$$

where the parameters are defined as in Theorem 4.1.

**5 Conclusion**

Some generic constructions for 3-party PAKE protocols [9,10] were proposed recently, but they do not accommodate client-to-client PAKE. Byun et.al [6] suggested
to design a generic construction of C2C-PAKE in the cross-realm setting by using 2-party PAKE and key distribution protocols, but they were not able to provide a scheme. In this paper, we present a general construction for the client-to-client PAKE protocols based on the generic construction for 3-party scenario. Moreover, we are able to prove its security by using the existing efficient protocols based on standard assumption instead of the random oracle models.

References


