Partitioned Group Password-based Authenticated Key Exchange with Privacy Protection

Hongfeng Zhu, Yuanle Zhang, Xueying Wang, and Liwei Wang
(Corresponding author: Hongfeng Zhu)

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

(Received June 20, 2019; Revised and Accepted Dec. 6, 2019; First Online Apr. 6, 2020)

Abstract

When a group Password-Based key exchange protocol is executed, the session key is typically extracted from two types of secrets: Shared keys (password) for authentication and freshly generated (nonces or timestamps) values. However, if one user (even subgroup users) runs the protocol with a non-matching password, all the others abort and no key is established. In this paper, we explore a more flexible, yet secure and privacy protection, GPAKE and put forward the notion of partitioned and privacy protection GPAKE, called PPP-GPAKE. PPP-GPAKE tolerates users that run the protocol on different passwords. Through a protocol run, any subgroup of users that indeed share a password, establish a temporary session key, and all the communication processes are user anonymity for outsiders by a temporary database helping. At the same time any two keys, each established by a different subgroup of users, are pair-wise independent if the corresponding subgroups hold different passwords. Compared with the related literatures recently, our proposed scheme can not only own high efficiency (only two communication rounds) and unique functionality, but is also robust to various attacks. Finally, we give the security proof and the comparison with the related works.

Keywords: Authentication; Group Key Agreement; Password; Privacy Protection

1 Introduction

With the popularization of network application, how to establish a secure channel between two nodes is a special problem worth considering. There is an increasing need for group PAKE [2] protocols to protect communications for a group of users. Although two-party password-authenticated key exchange (PAKE) protocols have been carefully studied for the past few years, group PAKE protocols have received much attention owing to it's a general settings. PAKE protocol is favored by cryptographers because of its short, low-entropy and easy-to-remember passwords for authentication. 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. However, due to the low entropy of passwords, PAKE protocol is vulnerable to dictionary attacks. In dictionary attacks, an adversary tries to break the security of a protocol by exhaustive search. Dictionary attacks can be classified into three types: Online (can be resisted by limiting number of attempts easily), off-line and undetectable on-line dictionary attacks, the two latter are not easy to resist. And group PAKE (GPAKE) is the natural extension to PAKE that empowers groups of more than two users to establish a session key, given that they share a common password. GPAKE focuses on a simpler, more realistic scenario where users only hold passwords, not credentials, and do not revoke them.

Consider the situation: Before participating in the GPAKE protocol, users must identify group members who claim to hold passwords. Then, the GPAKE protocol allows us to prove our understanding of passwords (and establish session keys). However, if only one user participates in the execution of the protocol with a different password, the user will be regarded as an active opponent and cause the user to terminate (even if all other users share the same password). The same principle is applied in (non-cryptographic) group key exchange protocol: Whenever authentication fails, the user usually terminates the protocol execution and does not establish a joint key between those who successfully authenticate each other. In this paper, we design group key exchange more flexibly. That is, a packet PAKE protocol based on partition and privacy protection, PPP-GPAKE protocol. That is to say, if a user runs the protocol with a mismatched password, other users do not have to wait to establish the key again.

And in the paper, on the premise of GPAKE, we add the concepts of partition and privacy protection. For the concept of partition, we consider that it more convenient and flexible for users to perform some operations. Partitioned GPAKE defines natural applications in specific scenarios. For example, in the Internet of Things (IoT) cluster or a smart home which including many smart devices, and these devices belonging to the same user may need to establish a shared key (assuming that all devices of a given user have been initialized with the same password). In addition, in multi-user scenarios, different IoT or a smart home clusters belonging to different users will coexist, and key establishment in one group should not affect other groups. If a user (or even a subgroup) runs a protocol with a mismatched password, the shared password must be updated, which can be a very expensive process. For the other concept of security attribute, the privacy protection, which can ensure all the messages transmitting on the public channel are not plaintext. So, we call the new protocol PPP-GPAKE (partitioned and privacy protection GPAKE) which tolerates users that run the protocol on different passwords and owns the privacy protection. It is easy to see that PPP-GPAKE setting avoids the above problems and reduce the waiting time of users. We introduce here the new notion of PPP-GPAKE, aiming at designs suited for scenarios where the specific group of users sharing a password. That is, if there is a user password error, other users do not need to wait, and immediately form a new shared password.

Although the PAKE protocols have been studied to deal with multiple participants in a single domain for many years [5,7,13], it has many details are worth studying. Subsequently, with the increasing popularity of various types of group communication applications [9], including partitioned and privacy protection, a new research direction for PAKE protocols has moved gradually. In the next section, we discuss related work on PAKE protocols. Then we present our PPP-PAKE protocol, followed by the security proof. We analyze the performance of the proposed protocol and draw our concluding remarks at the end.

2 Security Model and Security Goals

Assuming that a public password dictionary $D \subseteq \{01\}^*$ to be efficiently identifiable and of constant or polynomial size. Especially, we assume that D can be enumerated by a polynomial bounded opponent.The set $S = \{U_1, \ldots, U_N\}$ of users is partitioned in $l \geq 2$ disjoint subsets, such that $S = U_1 \cup U_2 \ldots \cup U_l$. All users in U_δ, for $\delta = 1, \ldots, l$, share a public password $pw^\delta \in D$, with $pw^\delta \neq pw^\gamma$ given $\delta \neq_\gamma \in \{1, \ldots, l\}$. For simplicity's sake, we assume that all passwords are randomly selected from D, and are represented by a bit string of the same size (denoted by K).

2.1 Communication Model and Adversarial Capabilities

Protocol instances. Users are modeled as probabilistic polynomial time (ppt) Turing machines. Every user $U \in S$ parallelly and we use $\prod_i^j$ to consult the jth instance of user i, which can be regarded as a process executed by U_i. Every instance we distribute seven variables which are shown in Table 1. For more details on the variable usage, we refer to the work of Bellare *et al*. In [2].

Communication network. Assume that any point-to-point connections between users are available. The network is, nevertheless, non-private and completely asynchronous. More specifically, it is controlled by the opponent, who may delay, insert and delete messages at will.

Adversarial capabilities. We limit to probabilistic polynomial time (ppt) adversaries. The capabilities of an opponent A are made explicit through a number of oracles allowing A to communicate with protocol instances run by the users:

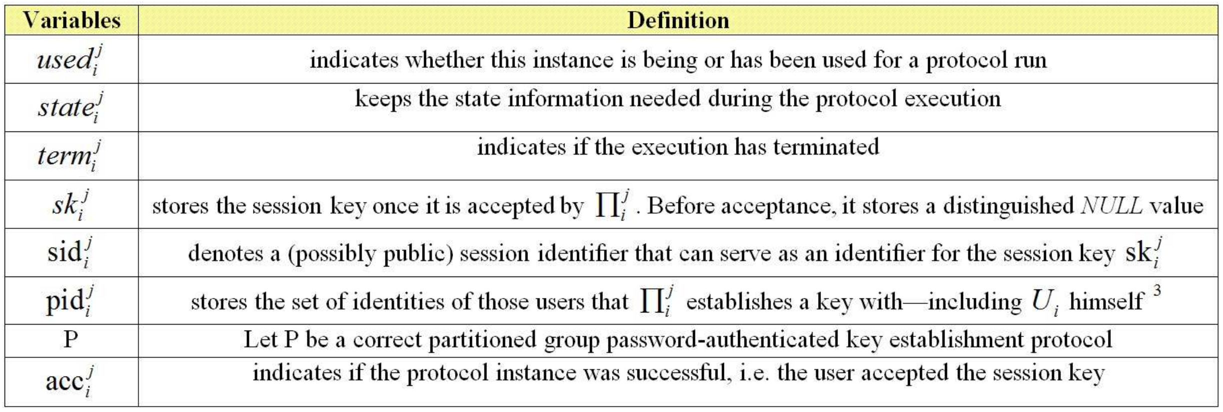
$Send(U_i, j, M)$. This oracle sends message M to the instance $\prod_i^j$ of U_i and returns the reply generated by this example. If A queries this oracle with an unused example $\prod_i^j$ and M being the set of users $\{U_{i_1}, \ldots, U_{i_\mu}\} \subseteq S$, going in for the protocol (including U_i), then the flag $used_i^j$ is set, and the first protocol message of $\prod_i^j$ for initializing a protocol run involving $\{U_{i_1}, \ldots, U_{i_\mu}\}$ is returned.

$Execute\left(\{\prod_{i_1}^{j_1}, \ldots, \prod_{i_\mu}^{j_\mu}\}\right)$. This oracle executes a complete protocol that run between specified unused instances of the respective users. The opponent gets a transcript of all messages sent over the network. A query to the Execute oracle should reflect passive eavesdropping. Especially, this Oracle can't guess passwords online.

$Reveal(U_i, j)$. Hand over the session key sk_i^j.

$Test(U_i, j)$. Active opponent A is only allowed to use one query of this form. Provided that sk_i^j is defined ($i.e.acc_i^j$ = true and $sk_i^j \neq$null), A can issue this query at any time when being activated. Then

Table 1: Variables



Variables	Definition
$used_i^j$	indicates whether this instance is being or has been used for a protocol run
$state_i^j$	keeps the state information needed during the protocol execution
$term_i^j$	indicates if the execution has terminated
sk_i^j	stores the session key once it is accepted by $\prod_i^j$. Before acceptance, it stores a distinguished $NULL$ value
sid_i^j	denotes a (possibly public) session identifier that can serve as an identifier for the session key sk_i^j
pid_i^j	stores the set of identities of those users that $\prod_i^j$ establishes a key with—including U_i himself [3]
P	Let P be a correct partitioned group password-authenticated key establishment protocol
acc_i^j	indicates if the protocol instance was successful, i.e. the user accepted the session key

with possibility 1/2 the session key sk_i^j and with possibility 1/2, a evenly chosen random session key is returned.

Corrupt (U_i). Returns the password *Corrupt* (U_i) held by U_i.

2.2 Correctness and Key Secrecy

2.2.1 Correctness

Our definition of correctness expands the standard one in GPAKE. Namely, without active countermeasure jamming, it should be the case that users holding matching passwords eventually set up a public session key as expected and assigning it the same name (sid). In addition, messages from users with mismatched password should not interrupt session key computations.

Definition 1. *(Correctness). Let D be a dictionary and S be a group of users as described earlier. After that, a partitioned group password-based key establishment protocol P is correct if there is a passive adversary A,*i.e. *A only uses the Execute oracle—a single execution of the protocol among $U_{i_1}, \ldots, U_{i_\mu}$ involves μ instances $\prod_{i_1}^{j_1}, \ldots, \prod_{i_\mu}^{j_\mu}$ and assures that with overwhelming possibility all examples:*

- *Accept,* i.e.$acc_{i_1}^{j_1} = \cdots = acc_{i_\mu}^{j_\mu} = true$;
- *Users belonging to the same subset U_τ the password-induced partition on S have accepted the same session key associated with the common session and partner identifier, that is $\forall s,\ r \in \{1, \ldots, \mu\}$ whenever U_{i_s}, $U_{i_r} \in U_\tau$, it holds $sk_{i_s}^{j_s}$=$sk_{i_r}^{j_r} \neq NULL$, $sid_{i_s}^{j_s} = sid_{i_r}^{j_r}$ and*
$$pid_{i_s}^{j_s} = pid_{i_r}^{j_r} \neq NULL.$$
(Note that if U_{i_s} is the only user in U_τ, then she will end up with unique $pid_{i_s}^{j_s}$, $sid_{i_s}^{j_s}$and $sk_{i_s}^{j_s}$.)

2.2.2 Key Secrecy

Here we define the main security concepts of partitioned GPAKE protocols. In order to do so, we introduce the concepts of cooperation and novelty to indicate which instances are associated in a common protocol session, and how to exclude trivial attacks, separately.

Partnering. We adopt the concept of cooperation from [11] where instances $\prod_i^j$, $\prod_t^m$ are partnered if $sid_i^j = sid_t^m$,$pid_i^j = pid_t^m$ and $acc_i^j = acc_t^m$= true. Nevertheless, in [1], pid lists user instances engaging in a public protocol execution. In our scene, pid explicits instances that go in for a common protocol execution and share a password. In other words, in [3] and in other GPAKE suggests, a user defines pid at the beginning of the protocol, while in our settings, a user finds pid at the end of the protocol.

Note that the above concept of cooperation defines an equivalence relation on the set of possible instances (namely, it is reflexive, symmetric and transitive). In addition, to avoid trivial situations we assume that an instance $\prod_i^j$ always accepts the session key constructed at the end of the corresponding protocol operation, if no deviation from the protocol specification occurs. In addition, Non-confrontational interference, all users in the same protocol session belonging to the same subset U_k, *i.e.* with the same password, should come up with the same session key, store it under the same session identifier and know whom they share it with.

Freshness. This notion helps specifying under which conditions a Test-query can be executed by the opponent in the security experiment. An instance is called fresh if the opponent never made one of the following queries: $\prod_i^j$:

Corrupt(U_t) to any U_t holding the same password as U_t(*i.e.* so that U_i and U_t are both in U_τ for some $\tau \in \{1, \ldots, l\}$);

Reveal(U_t, m) with $\prod_i^j$ and $\prod_t^m$ being partnered.

The concept of novelty allows us to rule out trivial attacks. Especially, displaying a session key from an instance $\prod_i^j$ evidently yields the session key of all instances partnered with $\prod_i^j$ and, hence, this kind of 'attack' is not take the security definition into account. In addition, note that this freshness definition means that corrupting users with different password from the one held by the uses specified in the Test query should be of no help to their opponents.

Key secrecy. Now that we have introduced cooperation and new concepts, we are ready to fully determine key confidentiality. As classic in password-based protocols, we observe that since the dictionary D has polynomial size we cannot prevent an adversary from correctly guessing a password $pw \in D$ used by any user. Hence, our goal is to limit the opponent A to verify password guesses online.

In the above setting, a protocol P is established for a fixed group key, let $Succ(\ell)$ be the possibility that an opponent A queries Test on a new instance $\prod_i^j$ and guesses correctly the bit b used by the Test oracle in a moment when $\prod_i^j$ is still fresh. Now we identify the advantages of A as the function

$$Adv_A(\ell) := |2Succ(\ell) - 1|.$$

We now introduce a function ε to capture the weaknesses that may due to the employed authentication technique; namely, as the opponent may guess passwords online, ε will explicit a bound on A's probability of guessing a shared password.

Definition 2. *(Key-secrecy). Let P be a correct partitioned group password-authenticated key establishment protocol, with D and S as mentioned above. Let A be a probabilistic polynomial time opponent with access to the Execute, Send, Reveal and Corrupt oracles. We say that P provides key secrecy, if for each such A, running in the experiment described in Section 2.1 and querying the Send oracle to at most q instances, the following inequality apply to some negligible function negl and some function ε which is at most linear in its second variable q:*

$$Adv_A(\ell) \leq \varepsilon(\ell, q) + \text{negl}(\ell).$$

Note that assuming passwords are chosen randomly and uniformly, and in each online attack, the opponent can only check a constant number of passwords, it holds $\varepsilon(\ell, q) = O(\frac{q}{|D|})$.

Remark 1. *Typically, in GAKE, the Corrupt oracle is used to model different flavors of forward security,* i.e. *to establish to what extent the leakage of authentication keys compromises the security of previously agreed session keys. In our scenario, however, corrupted users are to be understood as adversaries who might actually be legitimate members of a different password-defined subset U_δ. Thus, our model implicitly states that everyone who is not in the same password defined subset is understood as under adversarial control.*

2.3 Password Privacy and Privacy-Preserving

Unofficially, password-privacy ensures that an active opponent should not get any information about the use of passwords by legitimate users, so he should not even be able to tell if a given set of users really share the same password or not, unless he has guessed the involved password(s). Basically, if we consider the partition on the users set caused by the password allocation, then the opponents should not know about these partitions just by guessing wildly.

Interestingly, this concept is not relevant in many GPAKE proposals since, according to design, messages constructed from a non-matching password are typically recognized as maliciously generated and cause an abort (see for instance [6]). In fact, in this case, an active opponent may learn if two users U_i and U_t share the same password by starting a new session involving U_t and replaying messages generated by U_t in different executions. Now, the adversary just observes whether this rouge session is aborted or not. In contrast, in partitioned GPAKE protocols, executions always succeed and at their end, each participant eventually gets a valid key. However, only participants sharing the same password will share the same session key.

Our concept of password-privacy is rather inspired to that of affiliation hiding [10, 12] considered in authenticated key exchange. Association hiding means that an active opponent should not be able to obtain any information about group membership through a protocol execution (without considering trivial attacks where the opponent shares the affiliation of the victims). Especially, an opponent should not tell whether two users share the same affiliation or not. In our scenario, this means ensuring that no active opponent has access to information about the user's shared password, assuming he has not guessed the password used by any/some of them.

We use an indistinguishable game to simulate password privacy where the opponent A interacts with a challenger. First, he chooses the victim subgroup $U \subseteq S$ and two partitions p_0 and p_1 of it. Then the challenger randomly chooses one of the two partitions and assigns passwords (Random uniform selection) consistently with the corresponding subgroups. A wins if it can tell which of the two partitions has actually been chosen by the challenger, under the restriction that A cannot query the Reveal or Damage oracles on any of the users in U. We emphasize that in our game we do not assume passwords of all the remaining users in $S \backslash U$; These passwords can be even chosen by the opponent (*i.e.* the opponent can simulate any of these users himself).

Definition 3. *(Password-privacy). Let P be accurate partitioned GPAKE protocol. Consider a public dictionary D and (potential) set of users $S = \{U_1, \ldots, U_N\}$, where N is polynomial in the security parameter ℓ. Let A be a probabilistic polynomial time adversary interacting with a challenger Ch in the following game:*

1) *A selects a set of users $U \subseteq S$, and two partitions p_0 and p_1 of U.*

2) *Ch chooses a bit $b \in \{0,1\}$ uniformly at random and assigns a password, also chosen uniformly from the dictionary at random, for each subgroup of the partition p_b. In addition, he follows the specification of p.*

3) *A, equipped with Send and Execute, must output a guess.*

We say that P achieves password-privacy if every p.p.t. A wins the above password-privacy game with (at most) negligible probability over a random guess, provided he did not guess any password from a user in U. More precisely, for every p.p.t., let $Succ(\ell)$ be the probability that an adversary A guesses correctly the bit b selected by Ch. Now we define A's advantage as the function

$$Adv_A^{pwpri}(\ell) := |2.Succ(\ell) - 1|.$$

Let q denote the number of instances to which A has made a Send query. Then a protocol P has password-privacy if the following holds for some negligible function negl and some function ϵ which is at most linear in q,

$$Adv_A^{pwpri}(\ell) \leq \varepsilon(\ell, q)+\text{negl}(\ell),$$

our protocol has the privacy-preserving property which is realized by initiating two main ideas: firstly, the two partitions p_0 and p_1 can exchange the identity and nonces with an encrypted message using the shared password. And secondly, they can record the data in the temporary database as Table.3 does.

Definition 4. *(Privacy-preserving). Let P be a correct partitioned GPAKE protocol. Consider a public identity dictionary $\widetilde{ID}$ and (potential) set of users $S = \{U_1, \dots, U_N\}$, where N is polynomial in the safety parameter ℓ. Let A be a probabilistic polynomial time adversary interacting with a challenger Ch in the following game:*

1) *A selects a set of users $U \subseteq S$, and two partitions p_0 and p_1 of U.*

2) *Ch chooses a bit $b \in \{0,1\}$ uniformly at random and assigns an identity, also be randomly selected from the identity dictionary $\widetilde{ID}$, for every subgroup of the partition p_b. Further, he follows the specification of p.*

3) *A, equipped with Send and Execute, must output a guess.*

We say that P achieves Privacy-preserving if every p.p.t. A wins the above Privacy-preserving game with (at most) negligible possibility over a random guess, the premise is that he did not guess any identity from a user in U. More precisely, for every p.p.t., let $Succ(\ell)$ be opponent A who correctly guesses the probability of bit B chosen by ch. Now we define A's advantage as the function $Adv_A^{idpri}(\ell) := |2Succ(\ell) - 1|$.

Let q denote the number of instances to which A has made a Send query. Then a protocol P has password-privacy if the following holds for some negligible function negl and some function ϵ which is at most linear in q, $Adv_A^{idpri}(\ell) \leq \varepsilon(\ell, q)+\text{negl}(\ell)$.

3 The Proposed PPP-GPAKE Protocol

3.1 The Settings and Notations

Now we are ready to show our concrete construction, as shown in Figure 1. And the Notations are described in Table 2. First of all, we introduce the main building blocks of our scheme:

1) A hash function H, which will be modeled as a random oracle; we assume it to range on $\{0,1\}^d$, for d polynomial in the security parameter ℓ,

2) A private key encryption scheme $\prod$= (KEYGEN, ENC, DEC), assumed to be secure in the unforgivable sense of existence and achieving chosen ciphertext security (see [8] and Section 3.1 above). For each choice of the security parameter, we will denote by ρ and C the corresponding polynomial sized plaintext and ciphertext spaces, and assume ρ to be an additive group. Furthermore, we will assume that KEYGEN selects keys uniformly at random from the range of the random oracle H.

3) An ideal cipher $\varepsilon : D \times G \mapsto \hat{G}$, $\varepsilon : \widetilde{ID} \times G \mapsto \hat{G}$, where D is the password dictionary and $\widetilde{ID}$ is the identity dictionary, G is a cyclic group of order q (polynomial in) and $\hat{G}$ is a finite set of q elements.

3.2 The Construction

Figure 1 illus trates the process of authenticated key agreement phase.

Round 1. When N participants want to create a group session key, each U_i chooses uniformly a random value $x_i \in \{1, \dots, q-1\}$ and broadcasts $Y_i = \varepsilon_{pw_i}(U_i||g^{x_i})$. Each U_i will receive the messages (Y_t). If $\varepsilon^{-1}_{pw_i}(Y_t) = U_t||X_t \neq \perp$, then U_i sets $sid_{i,t} = U_i||Y_i||U_t||Y_t$ and computes $sk_{i,t} = H(U_i||U_t||X_i||X_t||X_t^{x_i})$. Otherwise U_i selects $sk_{i,t}$ equably at random in the range of H. Finally, for every U_t who holds a same two-party key as U_i, and user U_i defines the two-party key $sk_{i,t} = H(U_i||U_t||g^{x_i}||g^{x_t}||g^{x_i x_t})$, and a matching session identifier. Eventually, the user U_i stores $sid_{i,t}$ and the two-party key into a local temporary database, and the format are shown in the Table 3.

Table 2: Notations

Symbol	Definition
SK	the session key
Sid	session identifier
PW_A	Password of Alice
acc	the user accepted the session key
Adv_A	A's advantage as the function
p_0, p_1	two partitions
P	Let P be a correct partitioned group password-authenticated key establishment protocol
$\|\|$	concatenation operation
$S = \{U_1, \ldots, U_N\}$	S is the collection of users
$Succ(\ell)$	$Succ(\ell)$ is the probability that an adversary A guesses correctly the bit b

Round 2. In the Round 2, each user U_i selects uniformly at random a value $r_i \in \{1, \ldots, q-1\}$ and broadcasts $M_{it} = (sid'_{i,t}, a_{it} = ENC_{sk_{i,t}}(r_i))$, where $sid'_{i,t} = Y_i||Y_t$. For each $t \neq i$, receiving the message M_{it}, U_t will compare $sid'_{i,t}$ with the database's records for getting the identity information $U_i||U_t$. Then, user U_t compute $c_{it} = DEC_{sk_{i,t}}(a_{ti})$ using $sk_{i,t}$ and sets $pid = \{i\} \cup \{t : c_{it} \neq \{r_i, \perp\}\}$ for every received messages $(sid'_{t,i}, a_{ti})$. Then select the database to get the identity information. Further, for each $t \in pid$, $t \neq i$, it sets $r^*_t = c_{it}$ and also $r^*_t = r_i$. Next is the knowledge of session key and session identifier definitions. User U_i sets $acc_i = true$, derives the (sub-group) key as the addition $sk_i = \sum_{l \in pid} r^*_l$, and also the session $identifier^b$ $sid_i = \{sid_{i,t}||a_{i,t}\}_{t \in pid_i}||pid_i$.

If any authenticated process fails, the protocol will be terminated immediately.

4 Security Analysis

4.1 Tools

4.1.1 Bellare, Pointcheval and Rogaway PAKE

Our major building block is the EKE2 PAKE proposed by Bellare *et al.* in [2] which is secure in the so-called ideal cryptographic model (see [4]). In this model, it is assumed that there exists a publicly accessible random block cipher with a k-bit key and a n-bit input/output, that is random selection of all block ciphers in this form. Besides, it is necessary to assume the existence of ideal random functions, that is to say, we will model the hash function H used in the key derivation process as a random oracle [14]. It has been proved that the two models are equivalent, as evidenced in [8].

4.1.2 Unforgeable Encryption

For the choice of the second building block, a symmetric encryption scheme $\prod$, we will choose structure that fully reflects the strong concept of unforgeability; Namely, we should not even allow our opponents to generate any new valid ciphertext without the private key. Such property is defined in [14] as existential unforgeability; We rewrite Definition 5 from that paper here:

Definition 5. *Let $\prod$= (KEYGEN,ENC, DEC) be a private-key encryption scheme. Let ℓ be the security parameter and A be any pptm algorithm. Define* $\text{Adv}^{exist}_{A,\prod}(\ell) == \Pr[sk \leftarrow KEYGEN(1^\ell); y \leftarrow A : \text{DEC}_{sk}(y) \neq \perp]$. *At this, y is produced by the adversary A which may use an encryption oracle ε_{sk}, yet y must not have been directly returned by ε_{sk}. We say that $\prod$ is $(t, p, b; \delta)$-secure in the sense of existential unforgeability if for any adversary A which runs in time at most t and asks at most p queries to the encryption oracle, these totaling at most b bits, we have* $\text{Adv}^{exist}_{A,\prod}(\ell) \leq \delta(\ell)$. *Assuming t, p, and b, are polynomial in ℓ, if δ is negligible in ℓ we will simply say that $\prod$ is an unforgeable encryption scheme.*

Furthermore, in Theorem 1 of [14], it is proven that unforgeability along with chosen plaintext security means adaptive chosen ciphertext security. For our generic construction, we will make use of a symmetric key encryption scheme $\prod$ secure in this sense, therefore, we may assume that the adversary will cannot produce any valid ciphertext, nor to gain any information on the plaintexts underlying encrypted values. As evidenced in [14], such encryption scheme $\prod$ may simply be derived by instantiating appropriate block ciphers with unforgettable encryption patterns.

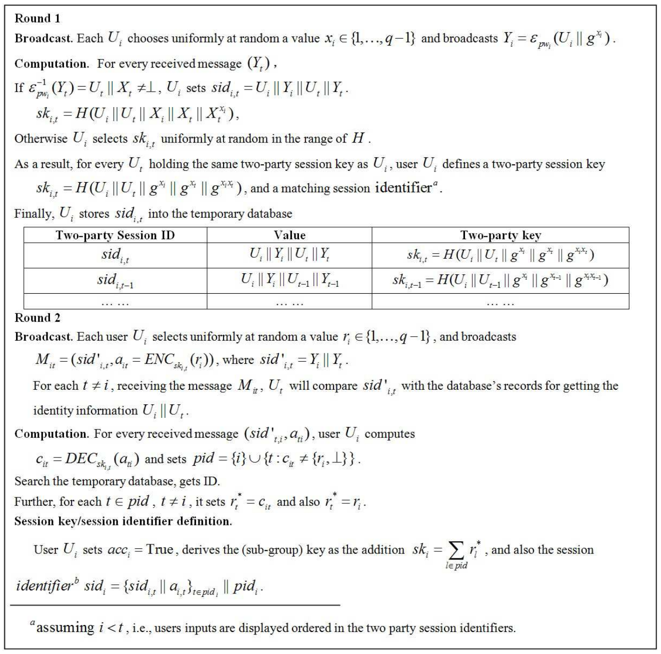
4.2 Security Proof

Theorem 1. *Let $\prod$= (KEYGEN, ENC, DEC) be a symmetric encryption scheme which is both unforgeable and chosen plaintext semantically-secure. Then, the protocol from Figure 1 is a correct partitioned password based group key agreement achieving key secrecy as defined in Definition 2 and password-privacy as defined in Definition 3 under the computational Diffie–Hellman assumption in group G in the random oracle/ideal cipher model.*

Correctness. In an true implementation of the protocol, it is easy to verify that all participants in the protocol

Table 3: The temporary database in the PPP-GPAKE protocol

Two-party Session ID	Value	Two-party key
$sid_{i,t}$	$U_i \| Y_i \| U_t \| Y_t$	$sk_{i,t} = H(U_i \| U_t \| g^{x_i} \| g^{x_t} \| g^{x_i x_t})$
$sid_{i,t-1}$	$U_i \| Y_i \| U_{t-1} \| Y_{t-1}$	$sk_{i,t-1} = H(U_i \| U_{t-1} \| g^{x_i} \| g^{x_{t-1}} \| g^{x_i x_{t-1}})$
… …	… …	… …


Round 1

Broadcast. Each U_i chooses uniformly at random a value $x_i \in \{1,\ldots,q-1\}$ and broadcasts $Y_i = \varepsilon_{pw_i}(U_i \| g^{x_i})$.

Computation. For every received message (Y_t),

If $\varepsilon^{-1}_{pw_i}(Y_t) = U_t \| X_t \neq \perp$, U_i sets $sid_{i,t} = U_i \| Y_i \| U_t \| Y_t$.

$sk_{i,t} = H(U_i \| U_t \| X_i \| X_t \| X_t^{x_i})$,

Otherwise U_i selects $sk_{i,t}$ uniformly at random in the range of H.

As a result, for every U_t holding the same two-party session key as U_i, user U_i defines a two-party session key

$sk_{i,t} = H(U_i \| U_t \| g^{x_i} \| g^{x_t} \| g^{x_i x_t})$, and a matching session identifier[a].

Finally, U_i stores $sid_{i,t}$ into the temporary database

Two-party Session ID	Value	Two-party key
$sid_{i,t}$	$U_i \| Y_i \| U_t \| Y_t$	$sk_{i,t} = H(U_i \| U_t \| g^{x_i} \| g^{x_t} \| g^{x_i x_t})$
$sid_{i,t-1}$	$U_i \| Y_i \| U_{t-1} \| Y_{t-1}$	$sk_{i,t-1} = H(U_i \| U_{t-1} \| g^{x_i} \| g^{x_{t-1}} \| g^{x_i x_{t-1}})$
… …	… …	… …

Round 2

Broadcast. Each user U_i selects uniformly at random a value $r_i \in \{1,\ldots,q-1\}$, and broadcasts

$M_{it} = (sid'_{i,t}, a_{it} = ENC_{sk_{i,t}}(r_i))$, where $sid'_{i,t} = Y_i \| Y_t$.

For each $t \neq i$, receiving the message M_{it}, U_t will compare $sid'_{i,t}$ with the database's records for getting the identity information $U_i \| U_t$.

Computation. For every received message $(sid'_{t,i}, a_{ti})$, user U_i computes

$c_{it} = DEC_{sk_{i,t}}(a_{ti})$ and sets $pid = \{i\} \cup \{t : c_{it} \neq \{r_i, \perp\}\}$.

Search the temporary database, gets ID.

Further, for each $t \in pid$, $t \neq i$, it sets $r_t^* = c_{it}$ and also $r_t^* = r_i$.

Session key/session identifier definition.

User U_i sets $acc_i = \text{True}$, derives the (sub-group) key as the addition $sk_i = \sum_{l \in pid} r_l^*$, and also the session identifier[b] $sid_i = \{sid_{i,t} \| a_{i,t}\}_{t \in pid_i} \| pid_i$.

[a]assuming $i < t$, i.e., users inputs are displayed ordered in the two party session identifiers.

Figure 1: An efficient partitioned GPAKE with privacy protection (PPP-GPAKE)

will terminate through accepting and computing the same session identifier and session key as participants with the same password.

Key secrecy. Evidence from several games, where a challenger interacts with the opponent confronting him with a counterfeit Test-challenge in the spirit of Definition 2. From game to game, the Challenger behaves differently from the previous one, with the corresponding effect on A's success probability. Following standard notation, we denote by $Adv(A, G_i)$ the opponent's advantage when confronted with Game i. The security parameter is denoted by ℓ.

In the sequel, we denote by q_{exe} the number of Execute calls made by the adversary. Also q will indicate the number of instances to which the opponent has launched a Send query, therefore, it is the number of instances that have suffered on-line attacks. In like wise,q_{ro} will express the number of queries A makes to the hash function H.

Game 0. This first game corresponds to a real attack, in which all the parameters are selected according to the actual scheme. By definition,

$$Adv(A, G_i) = Adv(A).$$

Game 1. We assume that the hash function H is simulated as a Random Oracle. Namely, each time a new query α is requested, the simulator selects u.a.r a value h_a from the range of H and stores the pair α, h_a in a table (from now on, the H-list). Should the value α be queried again, the simulator will look in the H-list and forward h_a as answer.

In addition, we explicit the ideal cipher simulation here. For a given password pw, the simulator will maintain an IC$_{pw}$-list in which for every query (pw,g) he stores a different value $\hat{g}$ which is selected uniformly at random in $\hat{G}$. Similarly, he also maintains a list capturing the decryption calls done to the Ideal Cipher (Called IC$^{-1}_{pw}$-list). Thus, a bijection $\sigma_{pw} : G \mapsto \hat{G}$ and list inverse are actually explicited by the two lists IC$_{pw}$-list, IC$^{-1}_{pw}$-list. The Random Oracle and the Ideal Cipher assumptions are made explicit by assuming

$$Adv(A, G_1) = Adv(A, G_0).$$

Game 2. In this game, we exclude certain conflicts of values chosen uniformly at random in different conversations. Namely, this game aborts in case the same exponent in Round 1 or the same random contribution in Round 2 is selected in different conversations by two (non-necessarily distinct) honest users. Similarly, we exclude the event that an Hcollision occurs at the time of extracting different two-party keys or session identifiers at the end of Round 1 in different protocol executions.

It is not hard to see that the difference between the two games

$$|Adv(A, G_2) - Adv(A, G_1)|,$$

the probability of 'partial collisions' on independent transcripts is bounded, which is in turn bounded by

$$\frac{q_{ro}^2}{2^d} + N^2 \left[\frac{1}{|G|} + \frac{1}{|\mathrm{P}|}\right],$$

where P is the plaintext space for $\prod$, from where the nonces are selected in Round 1.

Game 3. Consider the event that A queries the random oracle on the 5-tuple

$$(U_i \parallel U_t \parallel X_i \parallel X_t \parallel Z),$$

such that both the values X_i and X_t were generated by the simulator during the game and $Z{=}X_t^{x_i}$(essentially, if A queries the oracle on a valid CDH tuple). If such event (that we call Bad) happens, the simulation is aborted. Clearly,

$$|Adv(A, G_3) - Adv(A, G_2)| \leq P(Bad).$$

It is easy to see that for any adversary A that cause the Bad event to happen it is possible to construct another adversary B against the CDH assumption. The reduction is rather straightforward B, for inputting g^x, g^y, chooses a random index $q^* \leftarrow \{1, \ldots, q_{exe}\}$ and two user indices i, $t \leftarrow \{1, \ldots, N\}$ also at random. Then, in the q^*th protocol execution requested by the adversary B sets $X_i = g^x$ and $X_t = g^y$ for the users i and t, respectively. Finally, in the end of the game, it selects one random entry from H-list such that among the ones with $X_i = g^x$ and $X_t = g^y$, and returns the last value Z of the tuple. Clearly, if the event Bad occurs, and B guessed correctly the indices i, t, and q^*, then B found a solution for the CDH problem. Otherwise, if any of the guess was wrong, B aborts. It is not hard to see that

$$p(Bad) \times \frac{1}{q_{exe}} \frac{1}{N^2} \frac{1}{q_{ro}} \leq Adv_{G,g}^{CDH}(\ell),$$

where $Adv_{G,g}^{CDH}(\ell)$ is the probability that B has of winning a computational Diffie–Hellman challenge over G with generator g.

Game 4. Consider the event that A queries the random oracle on the 5-tuple

$$(U_i \parallel U_t \parallel X_i \parallel X_t \parallel Z),$$

such that the value X_t was generated by the simulator during the game, pw^* is the (random) password held by U_t whereas X_i is such that A made a query to the ideal cipher on input (pw^*, X_i) to get Y_i, and a Send query with the input (Y_i), because our scheme has privacy protection and for any adversary A that they just input the Y_i without the Identity of any user. If such event (that we call Bad^*) happens, the simulation is aborted. Clearly, $|Adv(A, G_4) - Adv(A, G_3)| \leq P(Bad^*)$. Now, the probability of the event Bad^* is bounded by the probability of a password guess, which is $O(q/|D|)$. We remark that from this point on in the simulation all H queries of the form $(U_i \parallel U_t \parallel X_i \parallel X_t \parallel X_i^{x_t})$, where users U_i and U_t sharing a password pw^* are either fully generated by the adversary or fully generated by the challenger. This means that for any two users sharing a password pw^* which has not been guessed by the adversary the corresponding two-party keys will be indistinguishable from random.

Game 5. This game deals with adversaries that only modify messages in Round 2, for the tested instance $\prod_i^j$. Precisely, consider any pair of users (U_i, U_t) for which the adversary had not made a random oracle query as the ones 'excluded' in the previous two games. Then if in the second part of the protocol execution the adversary A sends a valid message M_{it}^2 that decrypts correctly and is not a replay, then the game aborts.

With a simple reduction to the unforgeability of the encryption scheme M_{it}^2, it is possible to show that

$$|Adv(A, G_5) - Adv(A, G_4)| \leq Adv_{A\prod}^{exist}(\ell).$$

Where $Adv_{A\prod}^{exist}(\ell) \leq \delta(\ell)$, for some negligible function δ.

Game 6. Now, we modify the Execute and Send simulation in that we construct messages a_{it} as encryptions of 0, *i.e.* $a_{it} := ENC_{sk_{it}}(0)$. Precisely, this change is for all pairs of users (U_i, U_t) for which the adversary had not made a random oracle query as the ones excluded in the previous games. By relying on the CCA-security of $\prod$, one can argue that

$$|Adv(A, G_6) - Adv(A, G_5)| \leq Adv_{A\prod}^{CCA}(\ell).$$

After making this last change, the session key of a fresh session is completely random and independent from the simulated protocol transcript. Therefore, $Succ(\ell) = 1/2$, and the proof follows by putting together the bounds between the games.

Password privacy. The security proof for password privacy proceeds very similar to the one of key secrecy given above. The main idea is that after applying similar game changes as for key secrecy, the protocol messages become independent of the users passwords.

The games are defined as follows.

Game 0. This first game corresponds to a real attack, in which all the parameters are chosen as in the actual scheme. By definition, $Adv(A, G_0) = Adv(A)$.

Game 1. This is the same as Game 1 in the proof of Theorem 1. It simply makes explicit the simulation of the random oracle and the ideal cipher.

$$Adv(A, G_1) = Adv(A, G_0).$$

Game 2. This is the same as Game 2 in the proof of Theorem 1, and thus

$$|Adv(A,G_2) - Adv(A,G_1)| \le \frac{q_{ro}^2}{2^d} + N^2\left[\frac{1}{|G|} + \frac{1}{|P|}\right]$$

where P is the plaintext space for $\prod$, from where the nonces are selected in Round 1.

Game 3. This is the same as Game 3 in the proof of Theorem 1, and thus

$$\begin{aligned} & |Adv(A,G_3) - Adv(A,G_2)| \\ \le\ & q_{exe} \cdot N^2 \cdot q_{ro} \cdot Adv_{G,g}^{CDH}(\ell). \end{aligned}$$

Game 4. This proceeds similarly to Game 4 in the proof of Theorem 1. Let us consider the event that A queries the random oracle on the 5-tuple $(U_i \parallel U_t \parallel X_i \parallel X_t \parallel Z)$,such that the value X_t was generated by the simulator during the game, pw_t^* is the (random) password held by U_t, whereas X_i is such that A made a query to the ideal cipher on input (pw_t^*, X_i) to get Y_i, and a Send query with the input (U_i, Y_i). If such event (that we call Bad^*) happens, the simulation is aborted. The difference between this and the previous game lies in the occurrence of event Bad^* whose probability is bounded by that of a password guess. Therefore,

$$\begin{aligned} |Adv(A,G_4) - Adv(A,G_3)| &\le P(Bad^*) \\ &= O\left[\frac{q}{|D|}\right]. \end{aligned}$$

Game 5. This is the same as Game 5 in the proof of Theorem 3.2, and thus

$$|Adv(A,G_5) - Adv(A,G_4)| \le Adv_{G,g}^{CDH}(\ell),$$

where $Adv_{G,g}^{CDH}(\ell) \le \delta(\ell)$ for some negligible function δ.

Game 6. Finally, in this game, the challenger modifies the Execute and Send simulation by constructing messages a_{it} as encryptions of randomly selected values R_{it}, $i.e. a_{it} := ENC_{sk_{it}}(R_{it})$, while the session key is still computed using the randomly sampled values r_i. Based on the CCA-security of $\prod$ one can argue that

$$|Adv(A,G_6) - Adv(A,G_5)| \le Adv_{A,\prod}^{CCA}(\ell).$$

In addition, after making this last change, the protocol messages in the simulation are independent of the password selection, and, in particular, are distributed identically in both the cases when the users in U share the same password pw^* or have each a different password. So, the probability that the adversary succeeds in correctly guessing the bit b in this game is $1/2 - Succ(\ell) = 1/2$. Therefore, by putting together the various bounds of the game differences, we have that the chances of A to win the password-privacy game are only negligibly above $1/2 + O\left[q/|D|\right]$.

Privacy-preserving. The process of proof is the same as the process of Password privacy. So, we can get the opportunity of A to win the Privacy-preserving game are only negligibly above $1/2 + O\left[q/|\widetilde{ID}|\right]$.

5 Conclusion

This work presents a PPP-GPAKE protocol which firstly combines group Password-Based key exchange protocol with the security attributes of privacy and partitioned which can tolerates the user entered the wrong password. The first key idea is using a temporary database to make the privacy of our scheme possible, and the other key idea is using subgroup to compute the middle two-party session keys for tolerating the entered wrong passwords of some users. Additionally, the proposed scheme is no need pre-shared secret key which can make the proposed protocol become more practical. Moreover, the proposed protocol has been shown secure under the random oracle model. In the future, we will study the PPP-GPAKE under the standard model instead of random oracle model, and give the PPP-GPAKE more secure properties with high efficiency.

Acknowledgements

This work was supported by the Liaoning Provincial Natural Science Foundation of China (Grant No. 2019-MS-286), and Shenyang Science & Technology Innovation Talents Program for Young and Middle-aged Scientists (2019).

References

[1] M. Abdalla, J. M. Bohli, M. I. G. Vasco, R. Steinwandt, "(Password) Authenticated key establishment: From 2-party to group," in *Lecture Notes in Computer Science (TCC'07)*, pp. 499–514, 2007.

[2] S. M. Bellovin, M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proceedings IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 489-890, 1992.

[3] J. Black, P. Rogaway, "Ciphers with arbitrary finite domains," in *Ciphers with Arbitrary Finite Domains*, pp. 114–130, 2002.

[4] P. Chaidos, J. Groth, "Making sigma-protocols noninteractive without random oracles," in *Part of the Lecture Notes in Computer Science Book Series*, pp. 650-670, 2015.

[5] M. Chuangui, W. Fushan, G. Fengxiu, "Efficient client-to-client password authenticated key exchange based on RSA," in *Proceedings of the 5th IEEE International Conference on Intelligent Networking and Collaborative Systems*, pp. 233–238, 2013.

[6] R. Gelles, R. Ostrovsky, and K. Winoto, "Multiparty proximity testing with dishonest majority from equality testing," in *International Colloquium on Automata, Languages, and Programming*, pp. 537–548, 2012.

[7] X. Hu, Z. Zhang, "Cryptanalysis and enhancement of a chaotic maps-based three-party password authenticated key exchange protocol," *Nonlinear Dynamics*, vol. 78, no. 2, pp. 1293-1300, 2014.

[8] J. Katz, M. Yung, "Unforgeable encryption and chosen ciphertext secure modes of operation," in *Fast Software Encryption (FSE'00)*, *Lecture Notes in Computer Science*, , pp. 284–299, 2000.

[9] L. J. Liao, Z. I. Zhang, L. H. Zhu, "Computationally sound symbolic security reduction analysis of the group key exchange protocols using bilinear pairings," *Information Sciences*, vol. 20, no. 9, pp. 93–112, 2012.

[10] M. Manulis, B. Poettering and G. Tsudik, "AffiliationHiding key exchange with untrusted group authorities," in *Lecture Notes in Computer Science*, pp. 402–419, 2010.

[11] V. S. Naresh, S. Reddi, N. V. E. S. Murthy, "A provably secure cluster-based hybrid hierarchical group key agreement for large wireless ad hoc networks," *Human-centric Computing and Information Sciences,*, vol. 9, no. 1, pp. 1-32, 2019.

[12] A. Rivero-García, I. Santos-González, J. Munilla, M. Burmester, P. Caballero-Gil, "Secure lightweight password authenticated key exchange for heterogeneous wireless sensor networks," *Information Systems*, vol. 88, no. 101423, 2019. (`https://doi.org/10.1016/j.is.2019.101423`)

[13] B. Xiang, C. M. Chen, K. H. Wang, K. H. Yeh, T. Y. Wu, "Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3133-3142, 2019.

[14] X. Yang, X. P. Sheng, M. Zhang, "A certificateless signature scheme with strong unforgeability in the random oracle model," *Journal of Computational Methods in Sciences and Engineering*, vol. 18, no. 3, pp. 715-724, 2018.

Biography

Hongfeng Zhu, obtained his Ph.D. degree in Information Science and Engineering from Northeastern University. Hongfeng Zhu is a full professor of the Kexin software college at Shenyang Normal University. He is also a master's supervisor. He has research interests in wireless networks, social networks, network security and quantum cryptography. Dr. Zhu had published more than 60 international journal and international conference papers on the above research fields.

Yuanle Zhang, a postgraduate studying at Shenyang Normal University. She has researched interests in network security and quantum cryptography. Under the guidance of the teacher, she has published one article in EI journals.

Xueying Wang, obtained her Ph.D. degree in Management Science and Engineering from Wuhan University. Xueying Wang is a Dean of the Kexin software college at Shenyang Normal University. She is also a full professor and a master's supervisor. She has research interests in cloud computing, social networks, network security and E-commerce. Dr. Wang had published more than 40 international journal papers on the above research fields.

Liwei Wang, a postgraduate studying at Shenyang Normal University. She has researched interests in network security and quantum cryptography. Under the guidance of the teacher, she has published one article in EI journals.