Password-Authenticated Multi-Party Key Exchange with Different Passwords

Jeong Ok Kwon[†], Ik Rae Jeong[§], Kouichi Sakurai[‡], and Dong Hoon Lee[†]

[†]Center for Information Security Technologies (CIST), Korea University, Korea.
{pitapat,donghlee}@korea.ac.kr
[§] ETRI, 161 Gajeong-dong, Yuseoung-Gu, Daejeon, 305-700 Korea
jir@etri.re.kr
[‡]Department of Computer Science and Communication Engineering, Kyushu University, Japan.
sakurai@csce.kyushu-u.ac.jp

Abstract. Password-authenticated key exchange (PAKE) allows two or multiple parties to share a session key using a human-memorable password *only*. PAKE has been applied in various environments, especially in the "client-server" model of remotely accessed systems. Designing a secure PAKE scheme has been a challenging task because of the low entropy of password space and newly recognized attacks in the emerging environments. In this paper, we study PAKE for multi-party with different passwords which allows group users with different passwords to agree on a common session key by the help of a trusted server using their passwords only. In this setting, the users do not share a password between themselves but only with the server. The fundamental security goal of PAKE is security against dictionary attacks. We present the first two provably secure protocols for this problem in the *standard* model under the DDH assumption; our first protocol is designed to provide forward secrecy and to be secure against known-key attacks. The second protocol is designed to additionally provide *key secrecy* against curious servers. The protocols require a *constant* number of rounds.

Keywords. Provable security, group key exchange, password-based authentication, dictionary attacks

1 Introduction

Password-authenticated key exchange. To communicate securely over an insecure public network, it is essential that secret session keys are exchanged securely. The shared secret key may be subsequently used to achieve some cryptographic goals such as confidentiality or data integrity. In the public-key based and symmetric-key based key exchange protocols, a party has to keep long random secret keys. However, it is difficult for a human to memorize a long random string; thus, a party uses an additional storage device to keep the random string. On the other hand, password-authenticated key exchange (PAKE) protocols allow two or more specified parties to share a secret session key using *only* a human-memorable. Hence, PAKE protocols do not require that each party holds some devices such as smart cards or hardware tokens. From this point of view, PAKE provides convenience and mobility. Protocols for PAKE can be used in several environments, especially in networks where a security infrastructure like PKI (Public-Key Infrastructure) is not deployed. Because PAKE protocols provide a new and unique way to authenticate parties and derive high-quality cryptographic keys from low-grade passwords, PAKE has received significant attention.

Multi-party PAKE with different passwords. Several multi-party PAKE protocols have already been constructed so far. But most of those protocols assume that users of a group have a common pre-shared password. However, a multi-party PAKE protocol using the same password is not scalable in the sense that a user may want to communicate securely with other users who have not shared the same password. If a user has a password for a group, the number of passwords that the user has to memorize linearly increases depending on the number of groups. This is impractical since it is difficult for a user to remember many passwords. Multi-party PAKE with different passwords surmounts all the problems above. In this setting, a user shares a password only with a trusted server and the trusted server helps any set of users share a common session key. The main advantage of this solution is that it requires each user only to remember a single password with the trusted server. This seems to be a more practical scenario in the real world than multi-party PAKE with the same password. However, the server has to participate in the protocol run to help users of a group authenticate each other.

Intrinsic problems with respect to the server-aided PAKE. Compared to other security models, the most distinguishable characteristic of the PAKE security model is that the model must incorporate protection against dictionary attacks. The dictionary attacks are possible because of the low entropy of the password space. In practice, a password consists of 4 or 8 characters, such as a natural language phrase, to be easily memorized. The set of these probable passwords is small. As a consequence, there exists a relatively small dictionary. Usually dictionary attacks are classified into two classes: *on-line* and *off-line* dictionary attacks.

The on-line dictionary attacks are always possible, but these attacks cannot become a serious threat if the on-line attacks can be easily detected and thwarted by counting access failures. In the server-aided PAKE protocols, however, we must more carefully consider on-line dictionary attacks, because a malicious insider may launch on-line dictionary attacks indiscernibly using the server as a password verification oracle. If a failed guess cannot be detected and logged by the server, the attacks are called undetectable on-line dictionary attacks [18]. If this kind of attack succeeds on a PAKE protocol, an adversary is able to find the correct passwords of users and hence get everything allowed to the honest users. This, of course, breaks the overall security of the key exchange protocol. To prevent the undetectable on-line attacks, a server-aided PAKE protocol must provide a method by which the server can distinguish an honest request from a malicious one.

Even if there exists a little bit of redundancy information in the protocol messages, an adversary may perform an off-line dictionary attack by using the redundancy as a verifier for checking whether a guessed password is correct or not. We also have to consider insider attacks by a malicious user who attempts to perform an off-line dictionary attack on the other user's password using his information. The main security goal of PAKE schemes is to restrict the adversaries to attempting detectable on-line dictionary attacks only. If a PAKE scheme is secure, then an adversary cannot obtain any advantage in guessing the passwords and the session keys of users through off-line dictionary attacks.

Key secrecy with respect to the server-aided PAKE. One of the most basic security requirements of a key exchange protocol is *key secrecy* which guarantees that no computationally-bounded adversary should learn anything about the session keys shared between honest users by eavesdropping or sending messages of their choice to the users in the protocol. It is necessary that the key secrecy be also preserved against the server which behaves honestly but in the curious manner. That is, the server should not learn anything about the session keys of the users by eavesdropping, even if the server helps users of a group establish a session key between themselves. We formally define this security notion in Section 3.

Fault-tolerance in the server-aided PAKE. Secure, scalable and reliable group key exchanges have received much attention in recent years. We concentrate on the fault-tolerance with respect to users, where some users of a group can be disconnected by network failures caused by network misconfigurations or router failures. We say a protocol has fault-tolerance, even though some users of a group are disconnected by network failures, the other users of the group who execute the protocol correctly should be able to successfully share a session key without sending any additional message. We do not consider the fault of the server which makes the PAKE protocols fail.

2 Our Work in Relation to Prior Work

Our results. In this paper, we consider multi-party PAKE with different passwords. We summarize our main contributions as follows.

Constant-round PAKE protocols proven secure in the standard model. For a small device communicating over a wireless network, it is especially important to establish a session key with a small amount of computation, communication, and a small number of rounds. For these wireless applications, the multi-party key exchange protocols requiring linearly increasing round and computation complexities are not acceptable when the number of group users grows large. The best-known provably secure solution is N-party EKE-M in [13]. This protocol requires a constant number of rounds, however it does not have key secrecy with respect to curious servers. This protocol was conjectured secure when the block cipher is instantiated via an "ideal cipher" and the hash function is instantiated via an "ideal hash" (so-called the random oracle model). The idealized oracle methodology may enable the design and security proof of cryptographic protocols easier and more efficient. However a secure scheme in the idealized oracle model may not be secure in the real world if an idealized random function is instantiated with real functions [15, 30, 20, 16, 9]. Thus a scheme seems to be more reliable, if we do not use idealized random functions. Toward this goal, we present our first provably secure protocol, called $\mathcal{PAMKE}1$ in the standard model which provides forward secrecy and security against known-key attacks but not key secrecy with respect to curious servers. To improve key secrecy, we present the second protocol, called $\mathcal{PAMKE}2$, while enjoying all security features of $\mathcal{PAMKE}1$ and still requiring a constant number of rounds. We notice that designing of constant-round multi-party PAKE protocols having key secrecy with respect to curious servers is not easy at all.

Our protocols combine a two-party PAKE between a user and a server. Since we use a two-party PAKE scheme secure in the standard model, we may achieve our goal. For the two-party PAKE, we use the efficient two-party PAKE scheme suggested by Kobara and Imai [24]. As mentioned in [**?**], the security model which they used was different from the standard one and hence their result only applied to their specific model. We note that the security result in [24] may not be applied for a security analysis of our protocol. $\mathcal{PAMKE}1$ combines a two-party key distribution between a user and a server. Using a key distribution technique, we have the server distributes a session key selected by himself to each user. This, of course, does not have key secrecy with respect to servers. To solve this problem, $\mathcal{PAMKE}2$ combines a different approach with the key distribution approach, which allows all honest group users unbiasedly determine a session key together.

The building blocks that we combine are not fully independent modules, i,e., we do not consider a generic construction such as [6, 28, 21, 3]. The modular approach simplifies the design for protocols and analysis of protocols. Because of the merits using this modular approach, it has been used in the design of various key exchange protocols. However, the resulting protocols are often less efficient than tailored protocols because each protocol module is treated as a black box.

We compare the efficiency and the security of our protocols with the previous protocols. Table 1 summarizes the comparisons in which communication cost is the total number of bits that each user sends during a protocol run.

A mechanism for detection of un-/detectable on-line dictionary attacks. The proposed schemes have a mechanism to detect undetectable on-line dictionary attacks to server-side and a mechanism to detect detectable on-line dictionary attacks to user-side.

Related work. Several PAKE protocols in the multi-party setting have been studied [5, 11, 26, 13, 2]. Only [13] among them considers multi-party PAKE with different passwords (the others

Table 1. Comparisons of efficiency and security with the related protocols for multi-party with different passwords.

Scheme	N-party EKE-M	$\mathcal{PAMKE}1$	$\mathcal{PAMKE}2$
Round	2	3	5
Exponentiation (per user)	2	3	6
Communication (per user)	$\lvert p\rvert$	$\lvert p\rvert + \lvert\tau\rvert$	$2\lvert p\rvert + 2\lvert\tau\rvert$
Security	KK	KK&FS&DOD&UDOD	KSS&KK&FS&DOD&UDOD
Assumption	I.C.&I.H.	Standard	Standard

We use a group $\mathbb{Z}_p^*$ where p is a prime and $|\tau|$ is the length of an MAC tag. An FS protocol is a forward-secure key exchange protocol, a KK protocol is a secure key exchange protocol against known-key attacks, a KSS protocol has key secrecy with respect to servers. A DOD protocol and a UDOD protocol have a mechanism for detection of detectable and undetectable on-line dictionary attacks, respectively. I.C. denotes the ideal cipher model and I.H. denotes the ideal hash model.

consider multi-party PAKE with same passwords). In [13], Byun *et al.* have proposed two protocols, called N-party EKE-U in unicast networks which requires $O(n)$ rounds, where n is the number of users of a group, and N-party EKE-M in multicast networks which requires a constant rounds. Unfortunately, the N-party EKE-U protocol is vulnerable to off-line dictionary attacks (an explicit attack is known [33]) and the strengthened variant of this in [14] to counter the attack in [33] is still vulnerable an off-line dictionary attack (an explicit attack is known [31]). N-party EKE-M has been proven secure in the ideal cipher model and the ideal hash model without forward secrecy. N-party EKE-M does not provide key secrecy with respect to curious servers.

3 Security Model

The model defined in this section is based on Abdalla *et al.*'s model in [3, 4] and Katz *et al.*'s model in [23] which follow closely the model established by Bellare and Rogaway [7, 8] which has been extensively since then. We explicitly define notions of security which will be necessary for proving the security about our schemes in later sections.

In this section, we give a definition of security, called key secrecy with respect to server. The new definition captures the security level that the session key shared between honest group users should be only known to these group users and no one else, including the trusted server. The security notion can be viewed an extension of the one in the three-party setting in [3].

Networks. In the paper, we assume that there are two kinds of channel, a broadcast channel and a peer-to-peer channel. Since the peer-to-peer channel is a duplex channel, parties can simultaneously send messages to each other. In the both channels, it allows that an adversary intercept the messages and substitute their own messages for some of them. However, the broadcast network guarantees that all users receive the identical messages.

Participants. We fix nonempty sets, $\mathcal{G}$ of potential users of a group and $\mathcal{S}$ of potential servers. We assume the set $\mathcal{G}$ contains N users and the set $\mathcal{S}$ contains a single server. We consider a password-authenticated multi-party key exchange protocol in which any nonempty subset of $\mathcal{G}$, $\mathcal{G}_u$ wants to exchange a session key and a server $S \in \mathcal{S}$ helps the users with different passwords shares a common session key. We do not assume that the subsets are always include the same participant or always the same size. A participant P may have many instances of the protocol, which is either a user or a server. An instance of P is represented by an oracle P^s, for any $s \in \mathbb{N}$.

Long-term secret keys. Each user $U_i \in \mathcal{G}$ holds a password pw_i obtained at the start of the protocol using a password generation algorithm $\mathcal{PG}(1^\kappa)$ which on input a security parameter 1^κ outputs a password pw_i uniformly distributed in a password space $\mathsf{Password}$ of size $\mathcal{PW}$. The server $S \in \mathcal{S}$ holds $pw_S = [pw_i]_{U_i \in \mathcal{G}}$ with an entry for each user.

Adversaries. In this model, we assume two types of malicious inside attackers: a malicious group user[1] who can deviate from the protocol to perform an off-line dictionary attack against the other users, and the legal (trusted) server that behaves honestly but in curious manner to learn information about a session key shared between honest users. We also assume outside attackers who are neither a user nor the server. They are all probabilistic polynomial time Turing machines.

QUERIES FOR OUTSIDE ATTACKERS AND MALICIOUS USERS. An adversary $\mathcal{A}$ including a malicious user controls all the communications and makes queries to any oracle. The queries that $\mathcal{A}$ can use are as follows:

- $\mathsf{Execute}(\mathcal{G}_u, S^t)$: This query models passive attacks, where the adversary $\mathcal{A}$ gets the instances of honest executions of a protocol by $\mathcal{G}_u$ and S.
- $\mathsf{SendUser}(U_i^s, m)$: This query is used to send message m to user U_i^s and to get the response from U_i^s. Where the adversary $\mathcal{A}$ can intercept a message and then modify it, create a new message, or simply send it without any manipulation. This query models an active attack against a user.
- $\mathsf{SendServer}(S^t, m)$: This query is used to send message m to server S^t and to get the response from S^t. This query models an active attack against a server.
- $\mathsf{Reveal}(U_i^s)$: This query allows the adversary $\mathcal{A}$ to learn a particular session key established by a user. If a session key sk_i^s has previously been constructed by U_i^s, it is returned to the adversary. That is, this models *known-key attacks* in the real system. A PAKE protocol said to be secure against known-key attacks if compromise of multiple session keys for sessions other than the one does not affect its key secrecy. This notion of security means that session keys are computationally independent from each other. A bit more formally, this security protects against "Denning-Sacco" attacks [19] involving compromise of multiple session keys (for sessions other than the one whose secrecy must be guaranteed). Security against known-key attacks implies that an adversary cannot gain the ability to perform the off-line dictionary attacks on the passwords from the compromised session keys which are successfully established between honest parties.
- $\mathsf{Corrupt}(P)$: This query allows the adversary $\mathcal{A}$ to learn the long-term keys of parties. That is, this models *forward secrecy.* The adversary is assumed to be able to obtain long-term keys of parties, but cannot control the behavior of these parties directly (because once the adversary has asked a query $\mathsf{Corrupt}(P)$, the adversary may impersonate P in subsequent Send queries.) We restrict that on $\mathsf{Corrupt}(P)$ the adversary only can get the long-term keys, but cannot obtain any internal data of P.
- $\mathsf{Test}(U_i^s)$: This query is used to define the advantage of the adversary $\mathcal{A}$. This query is allowed only once by $\mathcal{A}$, and only to *fresh* oracles, which is defined later as freshness for Test-query. If the intended partners of U_i^s are part of the malicious set, the invalid symbol $\perp$ is returned. Otherwise, a coin b is flipped. If $b = 1$, the session key $sk_{U_i}^s$ held by U_i^s is returned, and if $b = 0$ a string randomly drawn from a session key distribution is returned.

A *passive adversary* can use the $\mathsf{Execute}$, Reveal, $\mathsf{Corrupt}$ and Test queries while an *active adversary* additionally can use the $\mathsf{SendUser}$/$\mathsf{SendServer}$ query. Even though the $\mathsf{Execute}$ query may seem to be useless since it can be simulated by repeatedly using the Send queries. Yet the $\mathsf{Execute}$ query is essential to distinguish on-line dictionary attacks from off-line dictionary attacks.

[1] In this paper, we do not assume the malicious insider such that a malicious user misleads other honest group users to compute different conference keys so that the honest users can not confer correctly [25, 34, 35, 22].

The Send queries are directly asked by the adversary and the number of those dose not take into account the number of Execute queries. That is, the number of undetectable on-line dictionary attacks and detectable on-line dictionary attacks can be bounded by the number of SendUser queries and SendServer queries, respectively.

Queries for malicious servers. A server S is allowed to access to the Execute, SendUser and Reveal oracles but not to a SendServer oracle (because the server knows the passwords for all users, one can easily simulate this oracle using the passwords). To emulate the server's advantage in learning information about a session key shared between honest group users, we define an additional oracle, TestGroup as follows.

- $\mathsf{TestGroup}(\mathcal{G}_u)$: This query is allowed only once by S, and only to *fresh* oracles, which is defined later as freshness for TestGroup-query. On this query a simulator flips a coin b. If b is 1, then the session key $sk_{\mathcal{G}_u}$ of a group $\mathcal{G}_u$ is returned. Otherwise a string randomly drawn from a session key distribution is returned.

Partnering. We define *partnering* for broadcast networks. We do not assume a synchronous network, and a round number is appended to a broadcast message. We assume that a sender's identity is also appended to the message to indicate the sender of the message. Let the session identifier sid_i^s be the concatenation of all broadcast messages that oracle U_i^s has sent or received. For the concatenation we assume that the messages are lexically ordered according to the sender's identity. Let a partner identifier pid_i^s for instance U_i^s be a set of the identities of the users with whom U_i^s intends to establish a session key. pid_i^s includes U_i itself. The oracles U_i^s and U_j^t are *partnered* if $\mathsf{pid}_i^s = \mathsf{pid}_j^t$ and $\mathsf{sid}_i^s = \mathsf{sid}_j^t$.

Freshness for the Test-query. We define a notion of freshness considering forward secrecy which means that an adversary does not learn any information about *previously* established session keys when making a Corrupt query. We say an oracle U_i^s is *fresh* if the following conditions hold:

- U_i^s has computed a session key $sk_i^s \neq \mathrm{NULL}$ and neither U_i^s nor one of its partners has been not asked for a Reveal query.
- Neither any party in pid_i^s nor S^t has been not asked for a Corrupt query by the adversary before a query of the form $\mathsf{SendUser}(U_i^s, *)$ or $\mathsf{SendServer}(S^t, *)$.

Freshness for the TestGroup-query. We define a notion of freshness for TestGroup. An oracle is said to be *fresh*, if no Reveal query is asked later to the oracle or one of its partners.

PAKE Security. Consider a game between an adversary $\mathcal{A}$ and a set of oracles.

Key secrecy with respect to outside/inside attackers. $\mathcal{A}$ is allowed to ask queries to the Execute, Reveal, Corrupt and SendUser/Server oracles in order to defeat the security of a protocol $\mathcal{P}$, and receives the responses. At some point during the game a Test query is asked to a *fresh* oracle, and the adversary may continue to make other queries. Finally the adversary outputs its guess b' for the bit b used by the Test oracle, and terminates. We define CG to be an event that $\mathcal{A}$ correctly guesses the bit b. The advantage of adversary $\mathcal{A}$ must be measured in terms of the security parameter k and is defined as follows:

$$\mathsf{Adv}_{\mathcal{P},\mathcal{A}}(k) = 2 \cdot \Pr[\mathsf{CG}] - 1.$$

The advantage function is defined as $\mathsf{Adv}_{\mathcal{P}}(k,t) = \max_{\mathcal{A}}\{\mathsf{Adv}_{\mathcal{P},\mathcal{A}}(k)\}$, where $\mathcal{A}$ is any adversary with time complexity t which is polynomial in k.

Key secrecy with respect to servers. We assume $\mathcal{A}$ is the malicious server. $\mathcal{A}$ is allowed to ask multiple queries to the Execute, SendUser and Reveal oracles. At some point during the game a $\mathsf{TestGroup}$ query is asked to a *fresh* oracle. We define the advantage of $\mathcal{A}$ in breaking the key privacy of the key exchange protocol $\mathcal{P}$ as $\mathsf{Adv}^{\mathsf{kss}}_{\mathcal{P},\mathcal{A}}(k) = 2 \cdot \Pr[\mathsf{CG}] - 1$ and the advantage function as $\mathsf{Adv}^{\mathsf{kss}}_{\mathcal{P}}(k,t) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathsf{kss}}_{\mathcal{P},\mathcal{A}}(k)\}$ as in the above definition.

Definition 1. We say a protocol $\mathcal{P}$ is a *secure multi-party PAKE protocol without guarantee of key secrecy with respect to servers* if the following two properties are satisfied:

- Validity: if all oracles in a session are partnered, the session keys of all oracles are same.
- Key secrecy: $\mathsf{Adv}_{\mathcal{P}}(k,t)$ is bounded by $q_{se}/\mathcal{PW} + \epsilon(k)$, where $\epsilon(k)$ is negligible, $q_{se} = q^U_{se} + q^S_{se}$, q^U_{se} is the number of $\mathsf{SendUser}$ queries, q^S_{se} is the number of $\mathsf{SendServer}$ queries, and $\mathcal{PW}$ is the size of the password space.

(1) We say a protocol $\mathcal{P}$ is a secure pake protocol for multi-party without guarantee of key secrecy against server if validity and key secrecy are satisfied when no Reveal and $\mathsf{Corrupt}$ queries are allowed.
(2) We say a protocol $\mathcal{P}$ is a secure $\mathsf{pake\text{-}kk}$ protocol for multi-party without guarantee of key secrecy against server for multi-party if validity and key secrecy are satisfied when no $\mathsf{Corrupt}$ query is allowed.
(3) We say a protocol $\mathcal{P}$ is a secure $\mathsf{pake\text{-}fs}$ protocol for multi-party without guarantee of key secrecy against server if validity and key secrecy are satisfied when no Reveal query is allowed.
(4) We say a protocol $\mathcal{P}$ is a secure $\mathsf{pake\text{-}kk\&fs}$ protocol without guarantee of key secrecy against server for multi-party if validity and key secrecy are satisfied.

Definition 2. We say a protocol $\mathcal{P}$ is a *secure multi-party PAKE protocol guaranteeing key secrecy with respect to servers* if the validity and the following property is satisfied:

- Key secrecy: (1) $\mathsf{Adv}_{\mathcal{P}}(k,t)$ is bounded by $q_{se}/\mathcal{PW} + \epsilon(k)$; and (2) $\mathsf{Adv}^{\mathsf{kss}}_{\mathcal{P}}(k,t)$ is bounded by $\epsilon(k)$.

(1) We say a protocol $\mathcal{P}$ is a secure $\mathsf{pake\text{-}kss}$ protocol for multi-party guaranteeing key secrecy against server if validity and key secrecy are satisfied when no Reveal and $\mathsf{Corrupt}$ queries are allowed.
(2) We say a protocol $\mathcal{P}$ is a secure $\mathsf{pake\text{-}kss\&kk}$ protocol for multi-party guaranteeing key secrecy against server if validity and key secrecy are satisfied when no $\mathsf{Corrupt}$ query is allowed.
(3) We say a protocol $\mathcal{P}$ is a secure $\mathsf{pake\text{-}kss\&fs}$ protocol for multi-party guaranteeing key secrecy against server if validity and key secrecy are satisfied when no Reveal query is allowed.
(4) We say a protocol $\mathcal{P}$ is a secure $\mathsf{pake\text{-}kss\&kk\&fs}$ protocol for multi-party guaranteeing key secrecy against server if validity and key secrecy are satisfied.

4 Multi-Party PAKE Protocols

In this section, we present two protocols with implicit authentication, $\mathcal{PAMKE}1$ and $\mathcal{PAMKE}2$ in the multi-party setting; A key exchange protocol is said to provide implicit key authentication if users are assured that no other users aside from partners can possibly learn the value of a particular secret key. The protocols require only a constant number of rounds, achieve forward secrecy, and are secure against known-key attacks. $\mathcal{PAMKE}1$ and $\mathcal{PAMKE}2$ are designed without using any ideal function and their security are proved under the DDH assumption. $\mathcal{PAMKE}1$ does not provide key secrecy with respect to servers, whereas $\mathcal{PAMKE}2$ provides the key secrecy.

4.1 The $\mathcal{PAMKE}1$ Protocol

$\mathcal{PAMKE}1$ consists of three building blocks; two-party PAKE in which each user of a group and the server exchange a secret key, detection of detectable/undetectable on-line dictionary attacks in which each user and the server check whether there are malicious attempts or not to make use of they as an oracle for on-line dictionary attacks, and key distribution in which the server distributes randomly selected a secret key to each user using the secret key resulted in the two-party PAKE. An example of an execution of $\mathcal{PAMKE}1$ is shown in Fig. 1 in Appendix D.

Public information. Let $\mathbb{G}$ be a finite cyclic group of order q in $\mathbb{Z}_p^*$. Two primes p, q such that $p = 2q + 1$, where p is a safe prime such that the DDH problem is hard to solve in $\mathbb{G}$. The terms, g_1 and g_2, are generators of $\mathbb{G}$ having order q, where g_1 and g_2 must be generated so that their discrete logarithmic relation is unknown. Let H be a hash function satisfying the collision-free property from $\{0,1\}^*$ to $\mathbb{Z}_q^*$. We note that H is not modeled as a random oracle but is just used for binding a user id and the password. Let $\mathsf{M} = (\mathsf{KEY.G}, \mathsf{MAC.G}, \mathsf{MAC.V})$ be a message authentication code (MAC) algorithm. $\mathsf{Mac.K}$ generates a key k_{mac}. Given k_{mac}, $\mathsf{MAC.G}$ computes a tag $\tau = \mathsf{Mac.gen}_{k_{mac}}(M)$ for a message M. $\mathsf{MAC.V}$ verifies a message-tag pair using key k_{mac}, and returns 1 if the tag is valid or 0 otherwise. Let $\mathcal{F}$ be a pseudo random function family.

Initialization. We assume that each user $U_i \in \mathcal{G}$ and the server S have shared a password pw_i, the public information and the set of user identities $\mathcal{G}_u$ that wants to exchange a session key.

Two-party PAKE.

1. Each user $U_i \in \mathcal{G}_u$ chooses a random number $x_i \in \mathbb{Z}_q^*$, computes $X_i = g_1^{x_i} \cdot g_2^{H(U_i \| S \| pw_i)} \ mod\ p$.
2. Each user U_i sends $(U_i || X_i)$ to S.
3. For each $i \in \mathcal{G}_u$, S chooses a random number $y_i \in \mathbb{Z}_q^*$, computes $Y_i = g_1^{y_i} \cdot g_2^{H(U_i \| S \| pw_i)} \ mod\ p$.
4. S sends $(S || Y_i)$ to each user U_i.
5. Upon receiving $(S || Y_i)$, each user U_i computes $k_i = (Y_i / g_2^{H(U_i \| S \| pw_i)})^{x_i} \ mod\ p$.
6. Upon receiving $(U_i || X_i)$, S analogously computes k_i.

Detection of undetectable/detectable on-line dictionary attacks.

1. Each user U_i computes $\tau_{i,S} = \mathsf{MAC.G}_{k_i}(U_i \| S \| X_i \| Y_i)$ and sends $(U_i || \tau_{i,S})$ to S.
2. For each $i \in \mathcal{G}_u$, S computes $\tau_{S,i} = \mathsf{MAC.G}_{k_i}(S \| U_i \| X_i \| Y_i)$ and sends $(S || \tau_{S,i})$ to U_i.
3. Upon receiving $(S || \tau_{S,i})$, each user U_i computes $\mathsf{MAC.V}_{k_i}(\tau_{S,i})$.
4. Each user U_i halts if $\mathsf{MAC.V}$ returns 0, or moves the next step otherwise.
5. Upon receiving $(U_i || \tau_{i,S})$, for each $i \in \mathcal{G}_u$, S checks the validity of $\tau_{i,S}$ using k_i.
6. S sets a set of user identities $\mathcal{G}_u^1$ that passes the MAC verification. Let $\mathcal{G}_u^1 = (U_1, \ldots, U_{|\mathcal{G}_u^1|})$.

Key distribution.

1. S chooses randomly a key K from $\{0,1\}^l$.
2. For each $i \in \mathcal{G}_u^1$, S computes $K_i = K \oplus H(\mathcal{G}_u^1 || k_i)$.
3. S broadcasts $(\mathcal{G}_u^1 || U_1 || K_1 || ... || U_{|\mathcal{G}_u^1|} || K_{|\mathcal{G}_u^1|})$.

Key computation. Each user U_i computes the session key $sk = \mathcal{F}_K(\mathcal{G}_u^1 || \mathsf{sid})$, where $\mathsf{sid} = (K_1 || \ldots || K_{|\mathcal{G}_u^1|})$.

Undetectable and detectable on-line dictionary attacks. $\mathcal{PAMKE}1$ is designated to be secure against undetectable and detectable on-line dictionary attacks in addition to off-line dictionary attacks. If a failed guess can be detected and logged by the server or the users, the attacks are not possible anymore. Our simple and efficient mechanism to detect the *undetectable* on-line dictionary attacks requires from each user to prove to the server that it knows the knowledge of password pre-shared with the server before getting the necessary information for key exchange from the server. Upon receiving the messages for proof of knowledge, the server verifies whether the knowledge proof is valid or not before responding according to the request of the user. If it is valid, the server gives information to the user to complete the key exchange; this may be viewed as a type of "challenge-response" mechanism. For realizing the proof of knowledge for a password, we use a mechanism that authenticates an acknowledgment message using an MAC keyed by an ephemeral Diffie-Hellman key generated by each user and the server. If the MAC verification is failed, the server will notice that whose password isbeing a target of undetectable on-line dictionary attacks and it be at a crisis. If the number of failing tries exceeds a predefined threshold, the server reacts and informs the target user to stop any further use of the password and to change the password into a new one. To prevent the *detectable* on-line dictionary attacks, $\mathcal{PAMKE}1$ uses the similar challenge-response mechanism; this can be viewed as a type of mechanisms for key confirmation. If the MAC verification is failed, a user will notice that his/her password is being a target of on-line dictionary attacks and it be at a crisis. After a small amount of detection of failures the user stops any further use of the password and changes the password into a new one.

To generate a valid message-tag pair, there are only three ways: an adversary guesses successfully a correct password at once or after a small number of guess (but it is generally very low according to the size of password space), solves the DDH problem or breaks the MAC algorithm.

Fault-tolerance. $\mathcal{PAMKE}1$ has fault- tolerance. If some users of a group are disconnected by network failures, the other users who execute the protocol correctly can successfully share a session key without any additional message sending and delay. If after sending the acknowledgement $\tau_{i,s}$, the user U_i is still connected to the network, one can receive the message K_i from the server and thus can derive the session key from the message. Of course, if the broadcast message of the server is disappeared from the network, the server needs to resend the message.

4.2 Security Result

The following theorem shows that $\mathcal{PAMKE}1$ is secure against off-line dictionary attacks since an adversary's capability to perform the off-line attacks is limited by its computational power, while the adversary can only test one password per a message by himself.

Theorem 1. Let $\mathbb{G}$ be a group in which the DDH assumption holds and $\mathcal{F}$ be a secure pseudo random function family. Then $\mathcal{PAMKE}1$ is a secure pake-kk&fs protocol under the DDH assumption. Concretely,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs}}_{\mathcal{PAMKE}1}(k,t,q_{ex},q^U_{se},q^S_{se}) \leq\, & 2(N_s+2q_{ex}+q^U_{se}+q^S_{se})\cdot|\mathcal{G}|\cdot\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)\\ & +2(q^U_{se}+q^S_{se})\cdot|\mathcal{G}|\cdot\mathsf{Adv}^{\mathsf{suf}}_{\mathsf{M}}(k,q^U_{se},q^S_{se})+2\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa,T,q,h)\\ & +\frac{2(q^U_{se}+q^S_{se})}{\mathcal{PW}}+\frac{|\mathcal{G}|(q_{ex}+q^U_{se}+q^S_{se})^2}{q}.\end{aligned}$$

where t is the maximum total game time including an adversary's running time, and an adversary makes q_{ex} Execute queries, q^U_{se} SendUser queries, and q^S_{se} SendServer queries. N_s is the upper bound of the number of sessions that an adversary makes, and $\mathcal{PW}$ is the size of the password space.

Proof of Theorem 1. Because of the limitation of space, the proof of this theorem is given in Appendix B.1. The proof of security for $\mathcal{PAMKE}1$ defines a sequence of hybrid experiments, starting with the real attack and ending up with an experiment where the adversary to break key secrecy has no advantage. Each experiment having some modifications represents a different security aspect.

4.3 The $\mathcal{PAMKE}2$ Protocol

$\mathcal{PAMKE}2$ is designed to resistant to curious servers. To achieve this goal, we use another approach called MAC key distribution and MAC-authenticated multi-party key exchange, instead of the key distribution approach in $\mathcal{PAMKE}1$. Through this mechanism, the session key is determined not by the server but unbiasedly by all honest group users together. $\mathcal{PAMKE}2$ is still requires a constant number of rounds. $\mathcal{PAMKE}2$ uses the mechanisms in $\mathcal{PAMKE}1$ for two-party PAKE and detection of undetectable/detectable on-line dictionary attacks, and the unauthenticated Burmester and Desmedt's group key exchange protocol (shortly, $\mathcal{BD}$) in [12] for the MAC-authenticated multi-party key exchange. An example of an execution of $\mathcal{PAMKE}2$ is shown in Fig. 2 in Appendix D. $\mathcal{PAMKE}2$ is the same as in $\mathcal{PAMKE}1$ except for the following points:

Detection of undetectable/detectable on-line dictionary attacks.

1. S computes $\tau_{S,i} = \mathsf{MAC.G}_{k_i}(U_i \| S \| X_i \| Y_i)$ and sends $(S || \tau_{S,i})$ to U_i.
2. Upon receiving $(S || \tau_{S,i})$, each user U_i computes $\mathsf{MAC.V}_{k_i}(\tau_{S,i})$.
3. Each user U_i halts if $\mathsf{MAC.V}$ returns 0, or moves the next step.
4. Each user U_i chooses a random number $r_i \in \mathbb{Z}_q^*$ and computes $\tau_{i,S} = \mathsf{MAC.G}_{k_i}(U_i \| S \| X_i \| Y_i \| g_1^{r_i})$.
5. Each user U_i sends $(U_i || \tau_{i,S} || g_1^{r_i})$ to S.
6. Upon receiving $(U_i || \tau_{i,S} || g_1^{r_i})$, for each $i \in \mathcal{G}_u$, S checks the validity of $\tau_{i,S}$ using k_i.
7. S sets $\mathcal{G}_u^1$ as the set that passes the MAC verification by arranging the identities in lexical order. Let $|\mathcal{G}_u^1| = n$ and $\mathcal{G}_u^1 = \{U_1, ..., U_n\}$.

MAC key distribution and MAC-authenticated multi-party key exchange.

1. S chooses an MAC key k_{mac} using $\mathsf{KEY.G}$.
2. For each $i \in \mathcal{G}_u^1$, S computes $K_i = k_{mac} \oplus H(\mathcal{G}_u^1 || k_i)$ and $\sigma_{1,i} = \mathsf{MAC.G}_{k_{mac}}(U_i || \alpha_i)$ where $\alpha_i = g_1^{r_i}$ *mod* p.
3. S broadcasts $(\mathcal{G}_u^1 || U_1 || K_1 || ... || U_n || K_n)$ and $(U_{j+1} || U_{j+(n-1)} || \alpha_j || \sigma_{j,1})$ for $1 \leq j \leq n$, where $j = j$ *mod* n.
4. Upon receiving the broadcast message of S, each user U_i computes k_{mac} from K_i using $\mathcal{G}_u^1$.
5. Each user U_i computes $\mathsf{MAC.V}_{k_{mac}}(U_{i-1} || \alpha_{i-1})$ and $\mathsf{MAC.V}_{k_{mac}}(U_{i+1} || \alpha_{i+1})$.
6. Each user U_i moves the next step if both MAC verifications are correct, or halts otherwise.
7. Each user U_i computes $\beta_i = (\alpha_{i+1}/\alpha_{i-1})^{r_i}$ *mod* p and broadcasts $(U_i \| \beta_i \| \sigma_{2,i} = \mathsf{MAC.G}_{k_{mac}}(U_i \| \beta_i))$.
8. Upon receiving the broadcast messages form users, S checks if one receives all broadcast messages of users in $\mathcal{G}_u^1$. If not, S requests owners of missing messages to resend the message.
9. Upon receiving $(U_j \| \beta_j \| \sigma_{2,j})$, for each $U_j \in \mathcal{G}_u^1$ $(j \neq i)$, each user U_i checks if the validity of $\sigma_{2,j}$ using k_{mac}.
10. Each user U_i computes $\gamma_i = (\alpha_{i-1})^{nr_i} \cdot \beta_i^{n-1} \cdot \beta_{i+1}^{n-2} \cdots \beta_{i-2}$ *mod* p if all the MAC verifications are correct or halts otherwise.

Key computation. Each user U_i computes the session key $sk_i = F_{\gamma_i}(\mathcal{G}_u^1 || \mathsf{sid})$, where $\mathcal{G}_u^1 = (U_1, \ldots, U_n)$, $\mathsf{sid} = (\mathsf{K} || \alpha || \sigma_1 || \beta || \sigma_2)$, $\mathsf{K} = (K_1 || \ldots || K_n)$, $\alpha = (\alpha_1 || \ldots || \alpha_n)$, $\sigma_1 = (\sigma_{1,1} || \ldots || \sigma_{1,n})$, $\beta = (\beta_1 || \ldots || \beta_n)$, and $\sigma_2 = (\sigma_{2,1} || \ldots || \sigma_{2,n})$.

Completeness. If everything works correctly in $\mathcal{PAGKE}2$, the session key computed by U_i is $sk_i = F_{\gamma_i}(\mathcal{G}_u^1 || \mathsf{sid})$, where $\gamma_i = g_1^{r_1 r_2 + r_2 r_3 + \ldots + r_n r_1}$ *mod* p.

Fault-tolerance. $\mathcal{PAMKE}2$ is not fully fault-tolerant. If someone among users of a group receiving the broadcast messages from S is disconnected by network failures, the session key computation would be failed. Because the session key is correctly shared between users, if and only if the users involved in the MAC key distribution and MAC-authenticated multi-party key exchange phase are linked in a cyclic. Until the cyclic structure are completed, the multi-party key exchange may be delayed.

4.4 Security Result

Theorem 2. Let $\mathbb{G}$ be a group in which the DDH assumption holds, $\mathcal{F}$ be a secure pseudo random function family and M be an unforgeable MAC algorithm. Then $\mathcal{PAMKE}2$ is a secure

$\mathsf{pake\text{-}kss\&kk\&fs}$ protocol. Concretely,

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{pake\text{-}kss\&kk\&fs}}_{\mathcal{PAMKE}2}(k,t,q_{ex},q^U_{se},q^S_{se})\\ &\le 2(2N_s+2q_{ex}+q^U_{se}+q^S_{se})\cdot|\mathcal{G}|\cdot\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)+4\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa,T,q,h)+\\ &\quad 2(q^U_{se}+q^S_{se}+2)\cdot|\mathcal{G}|\cdot\mathsf{Adv}^{\mathsf{suf}}_{\mathsf{M}}(k,q^U_{se})+\frac{2(q^U_{se}+q^S_{se})}{\mathcal{PW}}+\frac{|\mathcal{G}|(q_{ex}+q^U_{se})^2}{q}.\end{aligned}$$

where the parameters are defined as in Theorem 1.

Proof of Theorem 2. The proof of this theorem is given in Appendix B.2. The proof of security for $\mathcal{PAMKE}2$ follows the same path as in Theorem 1 defining a sequence of hybrid experiments.

5 Concluding Remarks

This paper considers multi-party PAKE with different passwords and provides the first provably secure two constant-round protocols without using any ideal function. The protocols introduced here are the best solution since the security of the protocols is based on weaker and more reasonable assumptions, and the protocols achieve constant-round complexity, yet much work remains to be done to improve the computational efficiency and fault-tolerance of the protocols having secrecy with respect to key secrecy, while preserving constant-round complexity.

References

1. M. Abdalla, E. Bresson, O. Chevassut, A. Essiari, B. M öller, and D. Pointcheval, *Provably Secure Password-Based Authentication in TLS*, In Proc. of ASIACCS'06, ACM Press, pages 35-45, ACM Press, 2006.
2. M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. *Password-based Group Key Exchange in a Constant Number of Rounds*, In Proc. of PKC '06, LNCS 3958, pages 427 - 442 , 2006.
3. M. Abdalla, P.-A. Fouque, D. Pointcheval. *Password-Based Authenticated Key Exchange in the Three-Party Setting*, In PKC05, LNCS 3386, pages 65-84, 2005.
4. M. Abdalla and D. Pointcheval. Interactive Diffie-Hellman Assumptions With Applications to Password-Based Authentication, In *FC05*, LNCS 3570, pages 341-356, 2005.
5. N. Asokan and P. Ginzboorg. *Key Agreement in Ad-hoc Networks*, Journal of Computer Communications 23(17), pages 1627-1637, 2000.
6. M. Bellare, R. Canetti, and H. Krawczyk. *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols*, In Proc. of 30th Annual ACM Symposium on Theory of Computing, ACM, pages 419-428, 1998.
7. M. Bellare and P. Rogaway. *Entity authentication and key distribution*, In Proc. of CRYPTO '93, LNCS 773, pages 232-249, Springer-Verlag, 1993.
8. M. Bellare and P. Rogaway. *Provably secure session key distribution-the three party case*, In Proc. of the 27th ACM Symposium on the Theory of Computing, 1995.
9. M. Bellare, A. Boldyreva and A. Palacio. *An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem*, In Proc. of EUROCRYPT '04, LNCS 3027, pages 171-188, 2004.
10. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. *Provably Authenticated Group Diffie-Hellman Key Exchange*, In Proc. of the 8th ACM conference on Computer and Communications Security, pages 255-264, 2001.
11. E. Bresson, O. Chevassut, and D. Pointcheval. *Group Diffie-Hellman Key Exchange Secure Against Dictionary Attacks*, In Proc. of ASIACRYPT 2002, LNCS 2501, pages 497-514, Springer-Verlag, 2002.
12. M. Burmester and Y. Desmedt. *A Secure and Efficient Conference Key Distribution System*, In Proc. of EUROCRYPT '94, LNCS 950, pages 275-286, Springer-Verlag, 1995.
13. J. W. Byun and D. H. Lee. *Password-Authenticated Key Exchange between Clients with Different Passwords*, In Proc. of ACNS '05, LNCS 3531, pages 75-90, 2005.
14. J. W. Byun and D.H. Lee. *Comments on Weaknesses in Two Group Diffie-Hellman Key Exchange Protocols*, IACR ePrint Archive, 2005/209, 2005.
15. R. Canetti, O. Goldreich, and S. Halevi. *The random oracle methodology, revisited*, In Pro. of the 32nd Annual ACM Symposium on Theory of Computing, pages 209-218, 1998.

16. R. Canetti, O. Goldreich and S. Halevi. *On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes*, In Pro. of 1st Theory of Cryptography Conference (TCC), LNCS 2951 , pages 40-57, 2004.
17. R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *Eurocrypt '02*. Full version available at http://eprint.iacr.org/2002/059.
18. Y. Ding, P. Horster. Undetectable on-line password guessing attacks, *ACM Operating Systems Review* 29 (4) (1995) 77-86.
19. D. Denning and G. M. Sacco. *Timestamps in Key Distribution Protocols*, Communications of the ACM, 24(8), pages 533-536, 1981.
20. S. Goldwasser and Y. Taumen. *On the (in)security of the Fiat-Shamir Paradigm*, In Proc. of STOC '03, pages 102-115, IEEE Computer Society, 2003.
21. J. Katz, R. Ostrovsky, and M. Yung. *Forward secrecy in Password-only Key Exchange Protocols*, In Proc. of SCN '02, LNCS 2576, pages 29-44,Springer-Verlag, 2002.
22. J. Katz and J. S. Shin, *Modeling Insider Attacks on Group Key-Exchange Protocols*, In Proc. of CCS '05, pages 180-189, 2005.
23. J. Katz and M. Yung. *Scalable Protocol for Authenticated Group Key Exchange*, In Proc. of CRYPTO '03, LNCS 2729, pages 110-125, Springer-Verlag, 2003.
24. K. Kobara and H. Imai. *Pretty-simple password-authenticated key-exchange under standard assumptions*, IEICE Transactions, E85-A(10): 2229-2237, Oct. 2002. Also available at `http://eprint.iacr.org/2003/038/`.
25. B. Klein, M. Otten, T. Beth, *Conference Key Distribution Protocols in Distributed Systems, In Proc. of Codes and Ciphers-Cryptography and Coding IV*, IMA, page 225-242, 1995.
26. S. M. Lee, J. Y. Hwang and D. H. Lee. *Efficient Password-Based Group Key Exchange*, In Proc. of TrustBus '04, LNCS 3184, pages 191-199, Springer-Verlag, 2004.
27. P. MacKenzie. *More Efficient Password Authenticated Key Exchange*, In Proc. of the RSA Data Security Conference, Cryptographer's Track (RSA CT '01), LNCS 2020, pages 361-377, Springer-Verlag, 2001.
28. A. Mayer and M. Yung. *Secure Protocol Transformation via "Expansion": From Two- Party to Groups*, In Proc. of 6th ACM Conference on Computer and Communication Security, ACM, pages 83-92, 1999.
29. M. Naor and O. Reingold. *Number-Theoretic Constructions of Efficient Pseudo-Random Functions*, In Proc. of the 38th IEEE Symposium on Foundations of Computer Science, pages 458-467, IEEE Computer Society, 2004.
30. J. B. Nielsen. *Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case*, In Proc. of CRYPTO '02, LNCS 2442, pages 111-126, 2002.
31. Raphael C.-W. Phan and B.-M Goi. *Cryptanalysis of the N-Party Encrypted Diffie-Hellman Key Exchange Using Different Passwords*, In Proc. of ACNS '06, LNCS ??, pages ??-??, Springer-Verlag, 2006.
32. V. Shoup. *On Formal Models for Secure Key Exchange*, Draft, 1999. Available at `http://eprint.iacr.org/1999/012`.
33. Q. Tang and L. Chen. *Weaknesses in Two Group Diffie-Hellman Key Exchange Protocols*, IACR ePrint Archive, 2005/197, 2005.
34. Wen-Guey Tzeng, *A Practical and Secure-Fault-Tolerant Conferenc-Key Agreement Protocol*, In Porc. of PKC 2000, volume 1751 of Lecture Notes in Computer Science, page 1-13, Springer Verlag, 2000.
35. Wen-Guey Tzeng and Zhi-Jia Tzeng, *Round-Efficient Conference Key Agreement Protocols with Provable Security*, Asiacrypt 2000, LNCS 1976, pages 614-627, 2004.

A Primitives

Decisional Diffie-Hellman Assumption. Let $\mathbb{G} = \langle g \rangle$ be any finite cyclic group of prime order q. The DDH problem is defined as follows: given a triple (U, V, W), determine that the triple is a Diffie-Hellman triple (g^a, g^b, g^{ab}) or a random triple (g^a, g^b, g^r). The advantage of an algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G},\mathcal{A}}(t)$, running in time t is ϵ, if

$$|\Pr[a, b \leftarrow \mathbb{Z}_q : \mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[a, b, r \leftarrow \mathbb{Z}_q : \mathcal{A}(g, g^a, g^b, g^r) = 1]| \geq \epsilon.$$

We say the DDH assumption holds in $\mathbb{G}$ if no probabilistic polynomial time algorithm $\mathcal{A}$ can solve the DDH problem with non-negligible advantage. We let $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)$ denote the maximum advantage which is over all adversaries $\mathcal{A}$s running in time at most t.

Diffie-Hellman Random Self-Reducibility [29]. We use a multiplicative subgroup $\mathbb{G}$ of $\mathbb{Z}_p^*$, where p is a prime, q is a prime divisor of $p-1$, and g is a generator of $\mathbb{G}$. We say that a probabilistic polynomial-time algorithm $\mathcal{R}$ is a random self-reducing algorithm, if $\mathcal{R}$ on any input $\langle p, q, g, g^a, g^b, g^c\rangle$ outputs $\langle p, q, g, g^{a'},\ g^{b'},\ g^{c'}\rangle$ such that if $c = a \cdot b \ mod\ q$, then $c' = a' \cdot b'\ mod\ q$, otherwise a', b', and c' are all random elements in $\mathbb{Z}_q$, where a, b and c are randomly selected from $\mathbb{Z}_q$. Such $\mathcal{R}$ can be constructed as follows:

$\mathcal{R}$ chooses s_1, s_2 and σ uniformly in $\mathbb{Z}_q$, and computes

$$g^{a'} = (g^a)^\sigma \cdot g^{s_1}, \quad g^{b'} = g^b \cdot g^{s_2},$$
$$g^{c'} = (g^c)^\sigma \cdot (g^a)^{\sigma \cdot s_2} \cdot (g^b)^{s_1} g^{s_1 \cdot s_2}$$

and outputs

$$\langle p, q, g, g^{a'}, g^{b'}, g^{c'}\rangle.$$

Pseudorandom Functions. Let $\mathcal{F} : Keys(\mathcal{F}) \times D \to R$ be a family of functions, and $g : D \to R$ a random function. $\mathcal{A}$ is an algorithm that takes an oracle access to a function and returns a bit. We consider two experiments:

$\mathbf{Exp}^{\mathsf{prf\text{-}1}}_{\mathcal{F},\mathcal{A}}$	$\mathbf{Exp}^{\mathsf{prf\text{-}0}}_{\mathcal{F},\mathcal{A}}$
$K \overset{R}{\leftarrow} Keys(\mathcal{F})$	$g \overset{R}{\leftarrow} Rand^{D \to R}$
$d \leftarrow \mathcal{A}^{\mathcal{F}_K(\cdot)}$	$d \leftarrow \mathcal{A}^{g(\cdot)}$
return d	return d

The advantage of an adversary $\mathcal{A}$ is defined as follows:

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F},\mathcal{A}} = \Pr[\mathbf{Exp}^{\mathsf{prf\text{-}1}}_{\mathcal{F},\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{\mathsf{prf\text{-}0}}_{\mathcal{F},\mathcal{A}} = 1].$$

The advantage function is defined as follows:

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, \mu) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{A}_{\mathcal{F},\mathcal{A}}}\},$$

where $\mathcal{A}$ is any adversary with time complexity T making at most q oracle queries and the sum of the length of these queries being at most μ bits. The scheme $\mathcal{F}$ is a secure pseudo random function family if the advantage of any adversary $\mathcal{A}$ with time complexity polynomial in κ is negligible.

Message Authentication Codes. A message authentication code (MAC) algorithm consists of three algorithm, $\mathsf{M} = (\mathsf{KEY.G}, \mathsf{MAC.G}, \mathsf{MAC.V})$. $\mathsf{Mac.K}$ generates a key k_{mac}. Given k_{mac}, $\mathsf{MAC.G}$ computes a tag $\tau = \mathsf{MAC.G}_{k_{mac}}(M)$ for a message M. $\mathsf{MAC.V}$ verifies a message-tag pair using key k_{mac}, and returns 1 if the tag is valid or 0 otherwise.

In defining the security of an MAC we use the standard definition of strong unforgeability under adaptive chosen-message attack. Namely, let M be an MAC scheme and $\mathcal{A}$ be an adversary, and consider the following experiment:

```
$\mathbf{Exp}^{\mathsf{suf}}_{\mathcal{A},\mathsf{M}}(k)$
$k_{mac} \leftarrow \{0,1\}^k$
$(M, \tau) \leftarrow \mathcal{A}^{\mathsf{M}_{k_{mac}}(\cdot)}(1^k)$
if $\mathsf{MAC.V}_{k_{mac}}(M, \tau) = 1$ and oracle $\mathsf{MAC.G}_{k_{mac}}(\cdot)$
  never returned $\tau$ on input $M$ then return 1
else return 0
```

The advantage of an adversary $\mathcal{A}$ is defined as: $\mathsf{Adv}^{\mathsf{suf}}_{\mathcal{A},\mathsf{M}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{suf}}_{\mathcal{A},\mathsf{M}}(k) = 1\right]$. We say that M is strongly unforgeable (SUF-secure) if $\mathsf{Adv}^{\mathsf{suf}}_{\mathcal{A},\mathsf{M}}(k)$ is negligible for all PPT algorithms $\mathcal{A}$. When we are interested in a concrete security analysis, we drop the dependence on k and say that M is (t, q, ϵ)-*SUF-secure* if $\mathsf{Adv}^{\mathsf{suf}}_{\mathcal{A},\mathsf{M}} \leq \epsilon$ for all $\mathcal{A}$ running in time t and making at most q queries to its M oracle. (We remark that allowing N queries to an oracle $\mathsf{MAC.V}_{k_{mac}}(\cdot,\cdot)$ cannot increase the advantage of an adversary by more than a factor of N.)

B Proofs of Theorems

B.1 Proof of Theorem 1

Consider an adversary $\mathcal{A}$ attacking $\mathcal{PAMKE}1$ in the sense of forward secrecy and security against known-key attacks. In this proof, we prove that the best strategy the adversary can take is to eliminate one password from one password dictionary per initiated session. Assume that $\mathcal{A}$ breaks $\mathcal{PAMKE}1$ with a non-negligible probability. The advantage of $\mathcal{A}$ is from the following two cases:

(Case 1) For the Test oracle U_i^s, all parties in pid_i^s have a partner oracle.
(Case 2) For the Test oracle U_i^s, there exists at least one party U_j $(j \neq i \wedge U_j \in \mathsf{pid}_i^s)$ such that U_j does not have a partner oracle.

For $i \in \{1, 2\}$, let $\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ i}}_{\mathcal{PAMKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se})$ be the advantage of an adversary from Case i. Then we have

$$\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs}}_{\mathcal{PAMKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se}) = \mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case1}}_{\mathcal{PAMKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se}) + \mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 2}}_{\mathcal{PAMKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se}).$$

Case 1 measures forward secrecy of $\mathcal{PAMKE}1$, which means that even with the long-term keys of parties any adversary does not learn any information about session keys which are successfully established between honest parties without any interruption. Consider the advantage from Case 1.

Claim 1. $\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 1}}_{\mathcal{PAMKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se}) \leq 2q_{ex}|\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + \frac{|\mathcal{G}|(q_{ex}+q^U_{se}+q^S_{se})^2}{q}$.

Proof of Claim 1. If the advantage of an adversary is from Case 1, the passwords of the parties may be revealed by Corrupt queries. Although Corrupt queries are allowed for the Test oracle Π_i^s, all instances in pid_i^s are executed by Execute queries by the definition of freshness. This case can be viewed that there are no passwords in the protocol, and thus we may ignore Corrupt queries.

A user may get information about the session key of a particular session that he did not participate, if a random number used by he is used twice by other users or the server in other sessions. The other case allows us to solve the DDH problem.

The advantage from Case 1 is as the follow:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 1}}_{\mathcal{PAMKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se}) = {}& \mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 1\text{-}Col}}_{\mathcal{PAMKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se}) + \\ & \mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 1\text{-}\overline{Col}}}_{\mathcal{PAMKE}1}(k, t, q_{ex}).\end{aligned}$$

Let Col be the event that there exists a user U_i in session s such that the random number x_i used by instance U_i^s is equal to the random number used by instance U_j $(i \neq j)$ or S in session s'. The probability that this event occurs $\Pr[\mathsf{Col}]$ is bounded by the birthday paradox as $\frac{|\mathcal{G}|(q_{ex}+q^U_{se}+q^S_{se})^2}{2q}$; this immediately implies that

$$\mathsf{Adv}^{\mathsf{pagke\text{-}kk\&fs\text{-}Case\ 1\text{-}Col}}_{\mathcal{PAGKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se}) = 2\Pr[\mathsf{CG} \wedge \mathsf{Col}] - 1 \leq 2\Pr[\mathsf{Col}] \leq \frac{|\mathcal{G}|(q_{ex} + q^U_{se} + q^S_{se})^2}{q}, \tag{1}$$

where q is the size of the group $\mathbb{G}$.

We consider the advantage in the case without event Col. We assume an adversary $\mathcal{A}$ making only a single $\mathsf{Execute}(\mathcal{G}_u,S^t)$ query (notice that this is sufficient for supporting Theorem 1). The set $\mathcal{G}_u$ and the number of parties are chosen by $\mathcal{A}$. Let $\mathcal{G}_u = \mathcal{G}_u^1 = \{U_1,...,U_n\}$. The distribution of the transcript $\mathcal{T}$ and the resulting session key sk of $\mathcal{PAMKE}1$ is given by:

$$\mathbf{Real} \overset{\text{def}}{=} \left\{ \begin{array}{l} x_1,\cdots,x_n,y_1,\cdots,y_n \leftarrow \mathbb{Z}_q^*;\ g_1^{x_1},\cdots,g_1^{x_n};\ g_1^{y_1},\cdots,g_1^{y_n};\ K \leftarrow \{0,1\}^l \\ \tau_{1,S} = \mathsf{MAC.G}_{g_1^{x_1y_1}}(U_1\|S\|g_1^{x_1}\|g_1^{y_1}), \tau_{2,S} = \mathsf{MAC.G}_{g_1^{x_2y_2}}(U_2\|S\|g_1^{x_2}\|g_1^{y_2}),\cdots, \\ \tau_{n,S} = \mathsf{MAC.G}_{g_1^{x_ny_n}}(U_n\|S\|g_1^{x_n}\|g_1^{y_n});\ \tau_{S,1} = \mathsf{MAC.G}_{g_1^{x_1y_1}}(S\|U_1\|g_1^{x_1}\|g_1^{y_1}), \\ \tau_{S,2} = \mathsf{MAC.G}_{g_1^{x_2y_2}}(S\|U_2\|g_1^{x_2}\|g_1^{y_2}),\cdots,\tau_{S,n} = \mathsf{MAC.G}_{g_1^{x_ny_n}}(S\|U_n\|g_1^{x_n}\|g_1^{y_n}) \\ K_1 = K \oplus H(\mathcal{G}_u^1||g_1^{x_1y_1}), K_2 = K \oplus H(\mathcal{G}_u^1||g_1^{x_2y_2}),\cdots,K_n = K \oplus H(\mathcal{G}_u^1||g_1^{x_ny_n}) \\ \mathcal{T} = (g_1^{x_1},\cdots,g_1^{x_n},g_1^{y_1},\cdots,g_1^{y_n},\tau_{1,S},\cdots,\tau_{n,S},\tau_{S,1},\cdots,\tau_{S,n},K_1,\cdots,K_n) \\ sk = \mathcal{F}_K(\mathcal{G}_u^1||K_1||\ldots||K_n) \end{array} \right\}.$$

Consider the following randomized distribution:

$$\mathbf{Exp_1} \overset{\text{def}}{=} \left\{ \begin{array}{l} w_1,x_1,\cdots,x_n,y_1,\cdots,y_n \leftarrow \mathbb{Z}_q^*;\ g_1^{x_1},\cdots,g_1^{x_n};\ g_1^{y_1},\cdots,g_1^{y_n};\ K \leftarrow \{0,1\}^l \\ \tau_{1,S} = \mathsf{MAC.G}_{g_1^{w_1}}(U_1\|S\|g_1^{x_1}\|g_1^{y_1}), \tau_{2,S} = \mathsf{MAC.G}_{g_1^{x_2y_2}}(U_2\|S\|g_1^{x_2}\|g_1^{y_2}),\cdots, \\ \tau_{n,S} = \mathsf{MAC.G}_{g_1^{x_ny_n}}(U_n\|S\|g_1^{x_n}\|g_1^{y_n});\ \tau_{S,1} = \mathsf{MAC.G}_{g_1^{w_1}}(S\|U_1\|g_1^{x_1}\|g_1^{y_1}), \\ \tau_{S,2} = \mathsf{MAC.G}_{g_1^{x_2y_2}}(S\|U_2\|g_1^{x_2}\|g_1^{y_2}),\cdots,\tau_{S,n} = \mathsf{MAC.G}_{g_1^{x_ny_n}}(S\|U_n\|g_1^{x_n}\|g_1^{y_n}) \\ K_1 = K \oplus H(\mathcal{G}_u^1||g_1^{w_1}), K_2 = K \oplus H(\mathcal{G}_u^1||g_1^{x_2y_2}),\cdots,K_n = K \oplus H(\mathcal{G}_u^1||g_1^{x_ny_n}) \\ \mathcal{T} = (g_1^{x_1},\cdots,g_1^{x_n},g_1^{y_1},\cdots,g_1^{y_n},\tau_{1,S},\cdots,\tau_{n,S},\tau_{S,1},\cdots,\tau_{S,n},K_1,\cdots,K_n) \\ sk = \mathcal{F}_K(\mathcal{G}_u^1||K_1||\ldots||K_n) \end{array} \right\}.$$

A standard argument shows that for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ running in time t:

$$|\mathsf{Pr}[(\mathcal{T},sk) \leftarrow \mathbf{Real} : \mathcal{A}(\mathcal{T},sk) = 1] - \mathsf{Pr}[(\mathcal{T},sk) \leftarrow \mathbf{Exp_1} : \mathcal{A}(\mathcal{T},sk) = 1]| \leq \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t).$$

Consider the following additionally randomized distribution:

$$\mathbf{Exp_2} \overset{\text{def}}{=} \left\{ \begin{array}{l} w_1,w_2,x_1,\cdots,x_n,y_1,\cdots,y_n \leftarrow \mathbb{Z}_q^*;\ g_1^{x_1},\cdots,g_1^{x_n};\ g_1^{y_1},\cdots,g_1^{y_n};\ K \leftarrow \{0,1\}^l \\ \tau_{1,S} = \mathsf{MAC.G}_{g_1^{w_1}}(U_1\|S\|g_1^{x_1}\|g_1^{y_1}), \tau_{2,S} = \mathsf{MAC.G}_{g_1^{w_2}}(U_2\|S\|g_1^{x_2}\|g_1^{y_2}),\cdots, \\ \tau_{n,S} = \mathsf{MAC.G}_{g_1^{x_ny_n}}(U_n\|S\|g_1^{x_n}\|g_1^{y_n});\ \tau_{S,1} = \mathsf{MAC.G}_{g_1^{w_1}}(S\|U_1\|g_1^{x_1}\|g_1^{y_1}), \\ \tau_{S,2} = \mathsf{MAC.G}_{g_1^{w_2}}(S\|U_2\|g_1^{x_2}\|g_1^{y_2}),\cdots,\tau_{S,n} = \mathsf{MAC.G}_{g_1^{x_ny_n}}(S\|U_n\|g_1^{x_n}\|g_1^{y_n}) \\ K_1 = K \oplus H(\mathcal{G}_u^1||g_1^{w_1}), K_2 = K \oplus H(\mathcal{G}_u^1||g_1^{w_2}),\cdots,K_n = K \oplus H(\mathcal{G}_u^1||g_1^{x_ny_n}) \\ \mathcal{T} = (g_1^{x_1},\cdots,g_1^{x_n},g_1^{y_1},\cdots,g_1^{y_n},\tau_{1,S},\cdots,\tau_{n,S},\tau_{S,1},\cdots,\tau_{S,n},K_1,\cdots,K_n) \\ sk = \mathcal{F}_K(\mathcal{G}_u^1||K_1||\ldots||K_n) \end{array} \right\}.$$

A standard argument shows that for any PPT algorithm $\mathcal{A}$ running in time t:

$$|\mathsf{Pr}[(\mathcal{T},sk) \leftarrow \mathbf{Exp_1} : \mathcal{A}(\mathcal{T},sk) = 1] - \mathsf{Pr}[(\mathcal{T},sk) \leftarrow \mathbf{Exp_2} : \mathcal{A}(\mathcal{T},sk) = 1]| \leq \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t).$$

Continuing in this way, we obtain the following distribution:

$$\mathbf{Exp_n} \overset{\text{def}}{=} \left\{ \begin{array}{l} w_1,\cdots,w_n,x_1,\cdots,x_n,y_1,\cdots,y_n \leftarrow \mathbb{Z}_q^*;\ g_1^{x_1},\cdots,g_1^{x_n};\ g_1^{y_1},\cdots,g_1^{y_n};\ K \leftarrow \{0,1\}^l \\ \tau_{1,S} = \mathsf{MAC.G}_{g_1^{w_1}}(U_1\|S\|g_1^{x_1}\|g_1^{y_1}), \tau_{2,S} = \mathsf{MAC.G}_{g_1^{w_2}}(U_2\|S\|g_1^{x_2}\|g_1^{y_2}),\cdots, \\ \tau_{n,S} = \mathsf{MAC.G}_{g_1^{w_n}}(U_n\|S\|g_1^{x_n}\|g_1^{y_n});\ \tau_{S,1} = \mathsf{MAC.G}_{g_1^{w_1}}(S\|U_1\|g_1^{x_1}\|g_1^{y_1}), \\ \tau_{S,2} = \mathsf{MAC.G}_{g_1^{w_2}}(S\|U_2\|g_1^{x_2}\|g_1^{y_2}),\cdots,\tau_{S,n} = \mathsf{MAC.G}_{g_1^{w_n}}(S\|U_n\|g_1^{x_n}\|g_1^{y_n}) \\ K_1 = K \oplus H(\mathcal{G}_u^1||g_1^{w_1}), K_2 = K \oplus H(\mathcal{G}_u^1||g_1^{w_2}),\cdots,K_n = K \oplus H(\mathcal{G}_u^1||g_1^{w_n}) \\ \mathcal{T} = (g_1^{x_1},\cdots,g_1^{x_n},g_1^{y_1},\cdots,g_1^{y_n},\tau_{1,S},\cdots,\tau_{n,S},\tau_{S,1},\cdots,\tau_{S,n},K_1,\cdots,K_n) \\ sk = \mathcal{F}_K(\mathcal{G}_u^1||K_1||\ldots||K_n) \end{array} \right\}.$$

For any PPT algorithm $\mathcal{A}$ running in time t, we have via standard hybrid argument:

$$|\mathsf{Pr}[(\mathcal{T}, sk) \leftarrow \mathbf{Real} : \mathcal{A}(\mathcal{T}, sk) = 1] - \mathsf{Pr}[(\mathcal{T}, sk) \leftarrow \mathbf{Exp_n} : \mathcal{A}(\mathcal{T}, sk) = 1]| \leq n \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t). \quad (2)$$

Consider the values $K, H(\mathcal{G}_u^1||g_1^{w_1}), \cdots, H(\mathcal{G}_u^1||g_1^{w_n})$ in experiment $\mathbf{Exp_n}$. They are constrained according to the following n equations:

$$\begin{aligned} K_1 &= K \oplus H(\mathcal{G}_u^1||g_1^{w_1}) \\ &\vdots \\ K_n &= K \oplus H(\mathcal{G}_u^1||g_1^{w_n}) \end{aligned}$$

Since only $n-1$ of them are linearly independent, the value of K is independent of the set of equations, and thus about the value of sk. This implies that, for any $\mathcal{A}$:

$$\mathsf{Pr}[(\mathcal{T}, sk_0) \leftarrow \mathbf{Exp_n}; sk_1 \overset{R}{\leftarrow} \{0,1\}^l; b \overset{R}{\leftarrow} \{0,1\} : \mathcal{A}(\mathcal{T}, sk_b) = b] = 1/2. \quad (3)$$

Combining with Equations (2), (3) and the fact that $n \leq |\mathcal{G}|$, we get $\mathsf{Adv}_{\mathcal{PAMKE}1}^{\text{pake-kk\&fs-Case 1-}\overline{\text{Col}}}(k, t, 1) \leq 2|\mathcal{G}| \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)$. We note that this immediately yields via a standard hybrid argument that

$$\mathsf{Adv}_{\mathcal{PAMKE}1}^{\text{pake-kk\&fs-Case 1-}\overline{\text{Col}}}(k, t, q_{ex}) \leq 2q_{ex}|\mathcal{G}| \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t). \quad (4)$$

This result stated in the theorem can be obtained using random self-reducibility properties of the DDH problem following [32] and will appear in the full version.

Equations (1) and (4) yield the the desired result.

Claim 2. The advantage from Case 2 is following:

$$\begin{aligned} \mathsf{Adv}_{\mathcal{PAMKE}1}^{\text{pake-kk\&fs-Case 2}}(k, t, q_{ex}, q_{se}^U, q_{se}^S) \leq\ & 2(N_s + q_{se}^U + q_{se}^S) \cdot |\mathcal{G}| \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t) \\ & + 2(q_{se}^U + q_{se}^S) \cdot |\mathcal{G}| \cdot \mathsf{Adv}_{\mathsf{M}}^{\mathsf{suf}}(k, q_{se}^U, q_{se}^S) \\ & + 2\mathsf{Adv}_{\mathcal{F}}^{\mathsf{prf}}(\kappa, T, q, h) + \frac{2(q_{se}^U + q_{se}^S)}{\mathcal{PW}}. \end{aligned}$$

Proof of Claim 2. To compute the upper bound of the advantage from Case 2, we assume that an adversary $\mathcal{A}$ gets the advantage from Case 2. In this case, the passwords of the parties are not revealed by freshness conditions. Informally, there are only two ways an adversary can get information about a particular session key; either the adversary successfully breaks the authentication part, which means that the adversary correctly guesses the passwords, or correctly guesses the bit b involved in the Test query.

To evaluate this advantage, we incrementally define a sequence of hybrid experiments having some modifications per each experiment, starting at the real experiment $\mathbf{Exp_0}$ and ending up with $\mathbf{Exp_5}$. We simulate the experiments and then consider the adversary attacking the simulated protocol. We denote the probability that an event E occurs in $\mathbf{Exp_i}$ as $\mathrm{Pr}_i[\mathsf{E}]$.

Experiment $\mathbf{Exp_0}$. This is the real attack, in which the server and each user are given pw_i for $U_i \in \mathcal{G}$. In this experiment, all the oracles in the game defining the advantage of an adversary in Section 3, are allowed to the adversary. The instances of parties answer to each SendUser query and each SendServer query with independent random exponents, respectively and the Execute query is proceeded similarly. Thus, the instances can easily answer to the Reveal, Corrupt, and Test queries. By definition,

$$\mathrm{Pr}_0[\mathsf{CG}] = (\mathsf{Adv}_{\mathcal{PAMKE}1}^{\text{pake-kk\&fs}}(k, t, q_{ex}, q_{se}^U, q_{se}^S) + 1)/2. \quad (5)$$

Experiment $\mathbf{Exp_1}$. We define the event $\mathsf{AskSend\text{-}WithPW}$ as the event that a flow m is generated by the adversary under pw and a $\mathsf{SendUser}(U_i^s, m)$ query or a $\mathsf{SendServer}(S^t, m)$ query is asked. In this experiment, we delete the executions wherein event $\mathsf{AskSend\text{-}WithPW}$ occurs. In these executions, we stop choosing b' at random:

$$|\Pr_0[\mathsf{CG}] - \Pr_1[\mathsf{CG}]| \leq \Pr_1[\mathsf{AskSend\text{-}WithPW}]. \tag{6}$$

Experiment $\mathbf{Exp_2}$. In this experiment, we replace each of the MAC key used as input in the detection phase of undetectable/detectable on-line dictionary attacks with a random key, using standard hybrid arguments. We can use a standard hybrid argument because the MAC keys are all independent ones. We show that the difference between the success probability of an adversary in the previous and current experiments is negligible, if the DDH assumption holds and as soon as $\mathsf{AskSend\text{-}WithPW}$ does not occur. Formally,

$$|\Pr_1[\mathsf{CG}] - \Pr_2[\mathsf{CG}]| \leq N_s \cdot |\mathcal{G}| \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t).$$

To measure the above probability, we define a series of $|\mathcal{G}_u| + 1$ games, $\mathbf{G_2^0}$,...,$\mathbf{G_2^{|\mathcal{G}_u|}}$. $\mathbf{G_2^0}$ is the same with experiment $\mathbf{Exp_1}$. Let $|\mathcal{G}_u| = n$. $\mathbf{G_2^t}$ (for $1 \leq t \leq |\mathcal{G}_u|$) is defined as the follow:

$\underline{\mathbf{G_2^t}}$

1. Select the passwords for all users in $\mathcal{G}$ executing $\mathcal{PG}(1^\kappa)$.
2. For all $1 \leq l \leq t$, select a random number w_l, and use $g_1^{w_l}$ (i.e., at random) instead of $k_l = g_1^{x_l y_l}$ as the MAC key.
3. For all $t < l \leq n$, use $k_l = g_1^{x_l y_l}$ (i.e., according to the protocol) as the MAC key.
4. Compute the session key as in $\mathbf{Exp_1}$.
5. For all oracle queries of an adversary, answer to them as in $\mathbf{Exp_1}$.

We construct a distinguisher $\mathcal{D}_2^{t-1,t}$ which solves the DDH problem using a difference of the advantages of an adversary $\mathcal{A}$ in $\mathbf{G_2^{t-1}}$ and $\mathbf{G_2^t}$. $\mathcal{D}_2^{t-1,t}$ is given (g_1, U, V, W) that an instance of the DDH problem whose base is g_1. $\mathcal{D}_2^{t-1,t}$ uses the instance by the random self-reducibility of the DDH problem. $\mathcal{D}_2^{t-1,t}$ perfectly simulates $\mathbf{G_{t\text{-}1}}$ or $\mathbf{G_t}$ depending on whether or not (g_1, U, V, W) is a DDH triple. That is, if (g_1, U, V, W) is the DDH triple, $\mathcal{D}_2^{t-1,t}$ simulates $\mathbf{G_{t\text{-}1}}$ or $\mathbf{G_t}$ otherwise. We assume $\mathcal{A}$ making only a single $\mathsf{Execute}$ query and a single $\mathsf{SendUser}$ or $\mathsf{SendServer}$ query. Let $\mathsf{SendUser}_r(U_i^s, m)$ be the r-th round's $\mathsf{SendUser}$ query to the instance ($\mathsf{SendServer}_r(S^t, m)$ is analogously defined). The concrete description of $\mathcal{D}_2^{t-1,t}$ is as the follow:

$\underline{\mathcal{D}_2^{0,1}(g_1, U, V, W)}$

1. Select the passwords for all users in $\mathcal{G}$ executing $\mathcal{PG}(1^\kappa)$.
2. Give $\mathcal{A}$ the passwords for all malicious users in $\mathcal{G}$.
3. For $\mathsf{Execute}(\mathcal{G}_u, S^t)$, $\mathsf{SendUser}_1(U_i^s, S^t)$ and $\mathsf{SendServer}_1(S^t, U_i^s)$, randomly select σ, s_1, s_2, and compute $\mathsf{U} = U^\sigma \cdot g_1^{s_1}$ and $\mathsf{V} = V \cdot g_1^{s_2}$. Compute $X_1 = \mathsf{U} \cdot g_2^{H(U_1 \| S \| pw_1)}$,$X_2 = g_1^{x_2} \cdot g_2^{H(U_2 \| S \| pw_2)}$,...,$X_n = g_1^{x_n} \cdot g_2^{H(U_n \| S \| pw_n)}$, $Y_1 = \mathsf{V} \cdot g_2^{H(U_1 \| S \| pw_1)}$,$Y_2 = g_1^{y_2} \cdot g_2^{H(U_2 \| S \| pw_2)}$,...,$Y_n = g_1^{y_n} \cdot g_2^{H(U_n \| S \| pw_n)}$.
4. For $\mathsf{Execute}(\mathcal{G}_u, S^t)$, $\mathsf{SendUser}_2(U_i^s, m)$ and $\mathsf{SendServer}_2(S^t, m)$, compute $\mathsf{W} = W^\sigma \cdot U^{\sigma s_2} \cdot V^{s_1} \cdot g_1^{s_1 s_2}$. Use W as the MAC key, $\tau_{1,S}$ ($\tau_{S,1}$), and use $g_1^{x_2 y_2}$,...,$g_1^{x_n y_n}$ as the MAC key $\tau_{2,S}$ ($\tau_{S,2}$),...,$\tau_{n,S}$ ($\tau_{S,n}$), respectively.

5. Compute the session key as in $\mathbf{Exp_1}$.
6. For $\mathsf{Test}(U_i^s)$, check whether all users in $\mathcal{G}_u$ have the same session key. If this check fails, return the invalid symbol $\perp$. If this check is correct and if U_i^s is *fresh*, return sk without coin flipping.
7. For all oracle queries of $\mathcal{A}$, answer to them following the protocol under the above restriction.
8. If $\mathcal{A}$ terminates with b', $\mathcal{D}_2^{0,1}$ returns b' and halts.

$\underline{\mathcal{D}_2^{1,2}(g_1, U, V, W)}$: Same as $\mathcal{D}_2^{0,1}$ except following points.

1. For $\mathsf{Execute}(\mathcal{G}_u, S^t)$, $\mathsf{SendUser}_1(U_i^s, S^t)$ and $\mathsf{SendServer}_1(S^t, U_i^s)$, randomly select σ, s_1, s_2, w_1, and compute $\mathsf{U} = U^\sigma \cdot g_1^{s_1}$ and $\mathsf{V} = V \cdot g_1^{s_2}$. Compute $X_1 = g_1^{x_1} \cdot g_2^{H(U_1\|S\|pw_1)}$,$X_2 = \mathsf{U} \cdot g_2^{H(U_2\|S\|pw_2)}$,$X_3 = g_1^{x_3} \cdot g_2^{H(U_1\|S\|pw_3)}$,..., $X_n = g_1^{x_n} \cdot g_2^{H(U_n\|S\|pw_n)}$, $Y_1 = g_1^{y_1} \cdot g_2^{H(U_1\|S\|pw_1)}$,$Y_2 = \mathsf{V} \cdot g_2^{H(U_2\|S\|pw_2)}$,$Y_3 = g_1^{y_3} \cdot g_2^{H(U_3\|S\|pw_3)}$,...,$Y_n = g_1^{y_n} \cdot g_2^{H(U_n\|S\|pw_n)}$.
2. For $\mathsf{Execute}(\mathcal{G}_u, S^t)$, $\mathsf{SendUser}_2(U_i^s, m)$ and $\mathsf{SendServer}_2(S^t, m)$, compute $\mathsf{W} = W^\sigma \cdot U^{\sigma s_2} \cdot V^{s_1} \cdot g_1^{s_1 s_2}$. Use $g_1^{w_1}$ as the MAC key, $\tau_{1,S}$ $(\tau_{S,1})$, and use W as the MAC key, $\tau_{2,S}$ $(\tau_{S,2})$. Use $g_1^{x_3 y_3}$,...,$g_1^{x_n y_n}$ as the MAC key, $\tau_{3,S}$ $(\tau_{3,n})$,...,$\tau_{n,S}$ $(\tau_{S,n})$, respectively.
3. If $\mathcal{A}$ terminates with b', $\mathcal{D}_2^{1,2}$ returns b' and halts.

Continuing in this way, we get the following distinguisher.

$\underline{\mathcal{D}_2^{n-1,n}(g_1, U, V, W)}$: Same as $\mathcal{D}_2^{n-2,n-1}$ except following points.

1. For $\mathsf{Execute}(\mathcal{G}_u, S^t)$, $\mathsf{SendUser}_1(U_i^s, S^t)$ and $\mathsf{SendServer}_1(S^t, U_i^s)$, randomly select $\sigma, s_1, s_2, w_1, ...,$ w_n, and compute $\mathsf{U} = U^\sigma \cdot g_1^{s_1}$ and $\mathsf{V} = V \cdot g_1^{s_2}$. Compute $X_1 = g_1^{x_1} \cdot g_2^{H(U_1\|S\|pw_1)}$,..., $X_{n-1} = g_1^{x_{n-1}} \cdot g_2^{H(U_{n-1}\|S\|pw_{n-1})}$, $X_n = \mathsf{U} \cdot g_2^{H(U_n\|S\|pw_n)}$, $Y_1 = g_1^{y_1} \cdot g_2^{H(U_1\|S\|pw_1)}$,...,$Y_{n-1} = g_1^{y_{n-1}} \cdot g_2^{H(U_{n-1}\|S\|pw_{n-1})}$,$Y_n = \mathsf{V} \cdot g_2^{H(U_n\|S\|pw_n)}$.
2. For $\mathsf{Execute}(\mathcal{G}_u, S^t)$, $\mathsf{SendUser}_2(U_i^s, m)$ and $\mathsf{SendServer}_2(S^t, m)$. Use $g_1^{w_1}$,...,$g_1^{w_n}$ as the MAC key, $\tau_{1,S}$ $(\tau_{S,1})$,...,$\tau_{n,S}$ $(\tau_{S,n})$, respectively.
3. If $\mathcal{A}$ terminates with b', $\mathcal{D}_2^{n-1,n}$ returns b' and halts.

Consider the case that the message X_i or Y_i of $\mathsf{SendServer}_1(S^t, X_i)$ and $\mathsf{SendUser}_1(U_i^s, Y_i)$ has not been previously computed by $\mathcal{D}_2^{t-1,t}$. Even though the instance involved in the $\mathsf{SendServer}$ or $\mathsf{SendUser}_1(U_i^s, Y_i)$ accepts itself, its partners may not be oracle instances. Thus a Test query involving this instance may always return the invalid symbol $\perp$. Therefore the advantage of $\mathcal{D}_2^{t,t-1}$ is as the follow (for $1 \leq t \leq n$):

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G},\mathcal{D}_2^{t-1,t}} &= |\Pr[u, v \xleftarrow{R} \mathbb{Z}_q^*; (g_1, U, V, W) \leftarrow (g_1, g_1^u, g_1^v, g_1^{uv}) : \mathcal{D}_2^{0,1}(g_1, U, V, W) = 1] - \\ &\quad \Pr[u, v, w \xleftarrow{R} \mathbb{Z}_q^*; (g_1, U, V, W) \leftarrow (g_1, g_1^u, g_1^v, g_1^w) : \mathcal{D}_2^{0,1}(g_1, U, V, W) = 1]| \\ &= |\Pr_2^{t-1}[\mathsf{CG}] - \Pr_2^t[\mathsf{CG}]|,\end{aligned}$$

which leads $|\Pr_2^0[\mathsf{CG}] - \Pr_2^n[\mathsf{CG}]| \leq |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)$. We note that the hybrid allow us to define N_s different experiments where N_s is the upper bound of the number of sessions that an adversary makes, and also immediately yields that $|\Pr_2^0[\mathsf{CG}] - \Pr_2^n[\mathsf{CG}]| \leq N_s \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)$.

We have the desire result between $\mathbf{Exp_1}$ and $\mathbf{Exp_2}$:

$$|\Pr_1[\mathsf{CG}] - \Pr_2[\mathsf{CG}]| \le N_s \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t). \tag{7}$$

Experiment $\mathbf{Exp_3}$. In this experiment, we replace the pseudo random function family $\mathcal{F}$ used to derive a session key with a random function. That is, in $\mathbf{Exp_2}$, a session key is computed by using a pseudo random function family $\mathcal{F}$, while in $\mathbf{Exp_3}$, a session key is computed by using a random function g. We show that the difference between the success probability of an adversary in the previous and current experiments is negligible, if $\mathcal{F}$ is a secure pseudo random function family. Formally,

$$|\Pr_2[\mathsf{CG}] - \Pr_3[\mathsf{CG}]| = \mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h). \tag{8}$$

Consider a distinguisher $\mathcal{D}$ to break pseudo randomness of a pseudo random function family $\mathcal{F}$ using the difference of the advantages of an adversary $\mathcal{A}$ in $\mathbf{Exp_2}$ and $\mathbf{Exp_3}$. $\mathcal{D}$ is given an oracle function $f(\cdot)$ in the experiment of pseudo randomness of the function family $\mathcal{F}$. $\mathcal{D}$ simulates Exp_2 or $\mathbf{Exp_3}$ depending on whether $f(\cdot)$ is a function from $\mathcal{F}$ or not. The more concrete description of $\mathcal{D}$ is as the follow:

$\underline{\mathcal{D}^{f(\cdot)}}$

1. For all oracle queries of $\mathcal{A}$, $\mathcal{D}$ answers to them as in $\mathbf{Exp_2}$ by using an oracle function $f(\cdot)$ instead of $\mathcal{F}$ to make a session key.
2. If $\mathcal{A}$ terminates with b', $\mathcal{D}$ returns b' and halts.

The advantage of $\mathcal{D}$ to break a pseudo random function family (with probability related to $\mathcal{A}$'s success probability) is as the follow.

$$\begin{aligned}\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h) &= |\Pr[K \overset{R}{\leftarrow} Keys(\mathcal{F}) : \mathcal{D}^{\mathcal{F}_K(\cdot)} = 1] - \Pr[g \overset{R}{\leftarrow} Rand^{D \rightarrow R} : \mathcal{D}^{g(\cdot)} = 1]| \\ &= |\Pr_2[\mathsf{CG}] - \Pr_3[\mathsf{CG}]|.\end{aligned}$$

In $\mathbf{Exp_3}$, answers to the Test query are purely random. That is, this implies the the bit b used by the Test oracle cannot be guessed by the adversary better than at random:

$$\Pr_3[\mathsf{CG}] = \frac{1}{2}. \tag{9}$$

From Equations (5), (6), (7), (8) and (9), the advantage from Case 2 is bounded as the follow:

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 2}}_{\mathcal{PAMKE}1}(k, t, q_{ex}, q^U_{se}, q^S_{se}) \\ &\le 2(\Pr_1[\mathsf{AskSend\text{-}WithPW}] + N_s \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + \mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h)).\end{aligned}$$

Probability of Event $\mathsf{AskSend\text{-}WithPW}$. The security against detectable and undetectable on-line dictionary attacks are measured by the probability that $\mathsf{AskSend\text{-}WithPW}$ occurs. To measure $\Pr_1[\mathsf{AskSend\text{-}WithPW}]$, we define two auxiliary experiments $\mathbf{Exp_{3'}}$ and $\mathbf{Exp_{3^\star}}$ similar to $\mathbf{Exp_3}$.

In experiment $\mathbf{Exp_{3'}}$, we change the computation of the MAC key computaions. We replace each of the MAC key with a random key using standard hybrid arguments. After this modification, the probability for the adversary to see the difference between previous and current experiments is to forge a new, valid message-tag pair, after having seen at most q^U_{se} or q^S_{se} valid message-tag pairs. Where the message was not previously sent by a user or the server. We have via a standard hybrid argument:

$$|\mathrm{Pr}_3[\mathsf{AskSend\text{-}WithPW}] - \mathrm{Pr}_{3'}[\mathsf{AskSend\text{-}WithPW}]| \le (q_{se}^U + q_{se}^S) \cdot |\mathcal{G}| \cdot \mathsf{Adv}_{\mathsf{M}}^{\mathsf{suf}}(k, q_{se}^U, q_{se}^S).$$

In experiment $\mathbf{Exp_{3^\star}}$, the view of the adversary is perfectly independent from the password of user. The difference between the experiments is in the way the Execute, SendUser and SendServer queries are answered. In this experiment, on $\mathsf{Send}_0(U_i, S^t)$ or $\mathsf{Send}_0(S^t, U_i)$, U_i and S randomly choose X_i and Y_i from $\mathbb{G}$, and send them, respectively. Notice that in $\mathbf{Exp_{3'}}$, U_i and S compute $X_i = g_1^{x_i} \cdot g_2^{H(U_i\|S\|pw_i)}$ *mod* p and $Y_i = g_1^{y_i} \cdot g_2^{H(U_i\|S\|pw_i)}$ *mod* p, and send them, respectively. For any an adversary, we have via a standard hybrid argument:

$$|\mathrm{Pr}_{3'}[\mathsf{AskSend\text{-}WithPW}] - \mathrm{Pr}_{3^\star}[\mathsf{AskSend\text{-}WithPW}]| \le (q_{se}^U + q_{se}^S) \cdot |\mathcal{G}| \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t).$$

In $\mathbf{Exp_{3^\star}}$, the adversary can not get information of the passwords with instances of the protocol by off-line manner from an information-theoretical viewpoint since the transcripts are indistinguishable from a random transcript in $\mathbb{G}$. Thus the transcripts are completely independent from the passwords. One remarks that the passwords cannot be correctly guessed by the adversary better than sending a message generated under a guessed password:

$$\mathrm{Pr}_{3^\star}[\mathsf{AskSend\text{-}WithPW}] = (q_{se}^U + q_{se}^S)/\mathcal{PW}.$$

The above three equations combined with $\mathrm{Pr}_1[\mathsf{AskSend\text{-}WithPW}] = \mathrm{Pr}_2[\mathsf{AskSend\text{-}WithPW}] = \mathrm{Pr}_3[\mathsf{AskSend\text{-}WithPW}]$ lead to that

$$\mathrm{Pr}_1[\mathsf{AskSend\text{-}WithPW}] = \frac{(q_{se}^U + q_{se}^S)}{\mathcal{PW}} + (q_{se}^U + q_{se}^S) \cdot |\mathcal{G}| \cdot (\mathsf{Adv}_{\mathsf{M}}^{\mathsf{suf}}(k, q_{se}^U, q_{se}^S) + \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)). \quad (10)$$

Finally, Claim 1 and Claim 2 yield the statement of the theorem:

$$\begin{aligned}\mathsf{Adv}_{\mathcal{PAMKE}1}^{\mathsf{pake\text{-}kk\&fs}}(k, t, q_{ex}, q_{se}^U, q_{se}^S) \le{}& 2(N_s + 2q_{ex} + q_{se}^U + q_{se}^S) \cdot |\mathcal{G}| \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)\\ &+2(q_{se}^U + q_{se}^S) \cdot |\mathcal{G}| \cdot \mathsf{Adv}_{\mathsf{M}}^{\mathsf{suf}}(k, q_{se}^U, q_{se}^S) + 2\mathsf{Adv}_{\mathcal{F}}^{\mathsf{prf}}(\kappa, T, q, h)\\ &+\frac{2(q_{se}^U + q_{se}^S)}{\mathcal{PW}} + \frac{|\mathcal{G}|(q_{ex} + q_{se}^U + q_{se}^S)^2}{q}.\end{aligned}$$

B.2 Proof of Theorem 2

Consider an adversary $\mathcal{A}$ attacking $\mathcal{PAMKE}2$ in the sense of forward secrecy and security against known-key attacks. In this proof, we also consider a curious server $\mathcal{S}$ attacking key privacy. Assume that $\mathcal{A}$ breaks $\mathcal{PAMKE}2$ with a non-negligible probability.

A user may get information about the session key of a particular session that it did not participate, if a transcript α_i (in phase for MAC key distribution and MAC-authenticated multi-party key exchange) by it is used twice by other users in other sessions; i.e., there exists a user U_i in session s such that random number r_i used by instance U_i is equal to the random number used by instance U_j $(i \ne j)$ in session s'. We denote this event as Repeat. The other cases allow us to solve the DDH problem, break strong unforgeability of the underling MAC algorithm or break pseudo randomness of a pseudo random function family (with probability related to the adversary's success probability). We now proceed with a more formal proof.

The advantage with event Repeat is bounded by the birthday paradox:

$$\mathsf{Adv}_{\mathcal{PAGKE}2}^{\mathsf{pagke\text{-}kk\&fs}}(k, t, q_{ex}, q_{se}^U, q_{se}^S) = 2\mathrm{Pr}[\mathsf{CG} \wedge \mathsf{Repeat}] - 1 \le 2\mathrm{Pr}[\mathsf{Repeat}] \le \frac{|\mathcal{G}|(q_{ex} + q_{se}^U)^2}{q}. \quad (11)$$

The advantage of $\mathcal{A}$ without event Repeat is from Case 1 and Case 2 which are defined in the proof of Theorem 1.

Claim 3. $\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 1}}_{\mathcal{PAMKE}2}(k, t, q_{ex}) \leq 4q_{ex}|\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)$.

Proof of Claim 3. Proving this advantage follows the same path as proving $\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 1\text{-}}\overline{\mathsf{Col}}}_{\mathcal{PAMKE}1}$ in Claim 1 (although Col occurs in an execution of $\mathcal{PAMKE}2$, a user can not obtain information about the session key of a particular session that it did not participate). Consider the MAC-authenticated multi-party key exchange phase. This converts the unauthenticated $\mathcal{BD}$ scheme to an authenticated protocol using the shared MAC key. Since the MAC key is not used to derive a session key, it does not leak any information about a session key. Thus we may ignore the authenticator using an MAC algorithm in the MAC-authenticated multi-party key exchange phase. Therefore we can see the advantage of an adversary breaking key secrecy by attacking the MAC-authenticated multi-party key exchange phase (i.e., the $\mathcal{BD}$ scheme) as the follow:

$$\mathsf{Adv}^{\mathsf{gke\text{-}fs}}_{\mathcal{BD}}(k, t, q_{ex}) \leq 2q_{ex}|\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t), \tag{12}$$

which is proved in [23].

From Equation (12) and $\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 1\text{-}}\overline{\mathsf{Col}}}_{\mathcal{PAMKE}1}$, we can get the desired result and hence we omit the details.

Claim 4. The advantage from Case 2 is as the follow:

$$\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 2}}_{\mathcal{PAMKE}2}(k, t, q_{ex}, q^U_{se}, q^S_{se}) \leq 2(N_s + q^U_{se} + q^S_{se}) \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + 2\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h) +$$
$$2(q^U_{se} + q^S_{se} + 2) \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{suf}}_{\mathsf{M}}(k, q^U_{se}) + \frac{2(q^U_{se} + q^S_{se})}{\mathcal{PW}}.$$

Proof of Claim 4. Proving this advantage follows the same path as that in Claim 2, until the experiment $\mathbf{Exp_2}$:

Experiment $\mathbf{Exp_3}$. Let Forge be the event that an adversary $\mathcal{A}$ against key privacy outputs a new, valid message-tag pair where the message was not previously sent by a user. In other words, $\mathcal{A}$ is sending a message it has built by itself, after having seen at most q^U_{se} valid message-tag pairs. We construct $\mathcal{M}$ which breaks MAC algorithm M using $\mathcal{A}$. $\mathcal{M}$ is given an MAC generation oracle $MAC.G(\cdot)$ and an MAC verification oracle $MAC.V(\cdot)$. The concrete description of $\mathcal{M}$ is as the follow:

$\mathcal{M}^{MAC.G(\cdot),MAC.V(\cdot)}$

1. Select the passwords for all users in $\mathcal{G}$ executing $\mathcal{PG}(1^\kappa)$.
2. Select i' from $[1, |\mathcal{G}|]$ and use MAC oracles to generate and verify MAC values of $U_{i'}$.
3. For $\mathsf{Corrupt}$ queries, answer normally. If $\mathcal{A}$ queries $\mathsf{Corrupt}(U_{i'})$, terminate with a random bit.
4. For all oracle queries of $\mathcal{A}$, answer as in $\mathbf{Exp_3}$ under the above restriction.
5. If a forged message-tag pair (m, σ) for $U_{i'}$ is found during simulation such that $MAC.V(m, \sigma) = 1$, output (m, σ) and halt.

The success probability of $\mathcal{M}$ depends on whether or not $\mathcal{A}$ makes a forged massage-tag pair and $\mathcal{M}$ correctly guesses i'. If these guesses are correct (and unless $\mathsf{AskSend\text{-}WithPW}$ does not occur), the simulation is perfect. So this immediately implies that

$$|\Pr_2[\mathsf{CG}] - \Pr_3[\mathsf{CG}]| \leq \Pr[\mathsf{Forge}] \leq |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{suf}}_{\mathsf{M}}(k, q^U_{se}). \tag{13}$$

Experiment $\mathbf{Exp_4}$. We replace the pseudo random function family $\mathcal{F}$ used to derive a session key with a random function. This experiment is equal to $\mathbf{Exp_3}$ in Claim 2. Then it is straightforward to see that

$$|\Pr_3[\mathsf{CG}] - \Pr_4[\mathsf{CG}]| = \mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h). \tag{14}$$

$$\Pr_4[\mathsf{CG}] = \frac{1}{2}. \tag{15}$$

Probability of Event $\mathsf{AskSend\text{-}WithPW}$. The result of $\Pr_1[\mathsf{AskSend\text{-}WithPW}]$ in $\mathcal{PAMKE}2$ is identical to that in $\mathcal{PAMKE}1$ since the parts that the passwords are used are equal in both protocols.

From Equations (5), (6), (7), (10), (13), (14) and (15), the advantage from Case 2 is bounded as the follow:

$$\mathsf{Adv}^{\mathsf{pake\text{-}kk\&fs\text{-}Case\ 2}}_{\mathcal{PAMKE}2}(k, t, q_{ex}, q^U_{se}, q^S_{se}) \leq 2(N_s + q^U_{se} + q^S_{se}) \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + 2\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h) +$$
$$2(q^U_{se} + q^S_{se} + 2) \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{suf}}_{\mathsf{M}}(k, q^U_{se}) + \frac{2(q^U_{se} + q^S_{se})}{\mathcal{PW}}.$$

Claim 5. $\mathsf{Adv}^{\mathsf{kss}}_{\mathcal{PAMKE}2}(k, t, q_{ex}, q^U_{se}) \leq 2N_s \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + 2\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h)$.

Proof of Claim 5. This claim shows that $\mathcal{PAMKE}2$ provides key secrecy with respect to the server, if the DDH assumption holds in $\mathbb{G}$ and if $\mathcal{F}$ is a secure pseudo random function family. Assume that a curious server $\mathcal{S}$ breaks key privacy with a non-negligible probability. In order to measure the advantage of $\mathcal{S}$, we define a series of $n+1$ experiments, $\mathbf{Exp_0}$,...,$\mathbf{Exp_n}$, where $|\mathcal{G}_u| = n$. $\mathbf{G_0}$ is the real attack in which the server and each user are given pw_i for all $U_i \in \mathcal{G}$. $\mathcal{S}$ can access to the Execute, SendUser, Reveal and TestGroup oracles. Since the instances for user know the password, they can easily answer to the queries. By definition,

$$\Pr_0[\mathsf{CG}] = (\mathsf{Adv}^{\mathsf{kss}}_{\mathcal{PAMKE}2}(k, t, q_{ex}, q^U_{se}) + 1)/2.$$

Exp₁ : All oracle queries are answered as in $\mathbf{Exp_0}$ except for Execute and $\mathsf{SendUser}_5(\mathcal{G}_u, m)$ queries.

1. For $\mathsf{Execute/SendUser}_5(\mathcal{G}_u, m)$, select $w_{1,2}$ at random from $\mathbb{Z}^*_q$, compute $\beta_1 = \frac{g_1^{w_{1,2}}}{g_1^{r_n r_1}}, \beta_2 = \frac{g_1^{r_3 r_2}}{g_1^{w_{1,2}}}, \ldots, \beta_n = \frac{g_1^{r_1 r_n}}{g_1^{r_{n-1} r_n}}$, and return $(\beta_1, \sigma_{2,1})||...||(\beta_n, \sigma_{2,n})$.
2. Compute $\gamma_1 = (g_1^{r_n r_1})^n \cdot (\beta_1)^{n-1} \cdots \beta_{n-1}, \gamma_2 = (g_1^{w_{1,2}})^n \cdot (\beta_2)^{n-1} \cdots \beta_n, \ldots, \gamma_n = (g_1^{r_{n-1} r_n})^n \cdot (\beta_n)^{n-1} \cdots \beta_{n+2}$, $sk_1 = \mathcal{F}_{\gamma_1}(\mathcal{G}_u||\mathsf{sid})$,..., $sk_n = \mathcal{F}_{\gamma_1}(\mathcal{G}_u||\mathsf{sid})$.

Exp₂

1. For $\mathsf{Execute/SendUser}_5(\mathcal{G}_u, m)$, select $w_{1,2}$,$w_{2,3}$ at random from $\mathbb{Z}^*_q$, compute $\beta_1 = \frac{g_1^{w_{1,2}}}{g_1^{r_n r_1}}, \beta_2 = \frac{g_1^{w_{2,3}}}{g_1^{w_{1,2}}}, \ldots, \beta_n = \frac{g_1^{r_1 r_n}}{g_1^{r_{n-1} r_n}}$, and return $(\beta_1, \sigma_{2,1})||...||(\beta_n, \sigma_{2,n})$.
2. Compute $\gamma_1 = (g_1^{r_n r_1})^n \cdot (\beta_1)^{n-1} \cdots \beta_{n-1}, \gamma_2 = (g_1^{w_{1,2}})^n \cdot (\beta_2)^{n-1} \cdots \beta_n, \gamma_3 = (g_1^{w_{2,3}})^n \cdot (\beta_3)^{n-1} \cdots \beta_1, \ldots, \gamma_n = (g_1^{r_{n-1} r_n})^n \cdot (\beta_n)^{n-1} \cdots \beta_{n+2}$, $sk_1 = \mathcal{F}_{\gamma_1}(\mathcal{G}_u||\mathsf{sid})$,..., $sk_n = \mathcal{F}_{\gamma_1}(\mathcal{G}_u||\mathsf{sid})$.

Continuing in this way, we get the following experiment.

Exp

1. For $\mathsf{Execute/SendUser}_5(\mathcal{G}_u, m)$, select $w_{1,2}, \ldots, w_{n,1}$ at random from $\mathbb{Z}_q^*$, compute $\beta_1 = \frac{g_1^{w_{1,2}}}{g_1^{w_{n,1}}}, \beta_2 = \frac{g_1^{w_{2,3}}}{g_1^{w_{1,2}}}, \ldots, \beta_n = \frac{g_1^{w_{1,n}}}{g_1^{w_{n-1,n}}}$, and return $(\beta_1, \sigma_{2,1})||...||(\beta_n, \sigma_{2,n})$.
2. Compute $\gamma_1 = (g_1^{w_{n,1}})^n \cdot (\beta_1)^{n-1} \cdots \beta_{n-1}, \gamma_2 = (g_1^{w_{1,2}})^n \cdot (\beta_2)^{n-1} \cdots \beta_n, \ldots, \gamma_n = (g_1^{w_{n-1,n}})^n \cdot (\beta_n)^{n-1} \cdots \beta_{n+2}$, $sk_1 = \mathcal{F}_{\gamma_1}(\mathcal{G}_u||\mathsf{sid})$,..., $sk_n = \mathcal{F}_{\gamma_1}(\mathcal{G}_u||\mathsf{sid})$.

A standard argument shows that for any PPT algorithm running in time t (for $0 \leq i < n$):

$$|\Pr_i[\mathsf{CG}] - \Pr_{i+1}[\mathsf{CG}]| \leq \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t).$$

We construct a distinguisher $\mathcal{D}_{i,i+1}$ using the difference of the advantages of an adversary $\mathcal{S}$ in $\mathbf{Exp_i}$ and $\mathbf{Exp_{i+1}}$ such that the following equation holds:

$$\begin{aligned} \mathsf{Adv}_{\mathbb{G},\mathcal{D}_{i,i+1}}^{\mathsf{ddh}}(t) &= |\Pr[u, v \xleftarrow{R} \mathbb{Z}_q^*; (g_1, U, V, W) \leftarrow (g_1, g_1^u, g_1^v, g_1^{uv}) : \mathcal{D}_{i,i+1}(g_1, U, V, W) = 1] - \\ &\quad \Pr[u, v, w \xleftarrow{R} \mathbb{Z}_q^*; (g_1, U, V, W) \leftarrow (g_1, g_1^u, g_1^v, g_1^w) : \mathcal{D}_{i,i+1}(g_1, U, V, W) = 1]| \\ &= |\Pr_i[\mathsf{CG}] - \Pr_{i+1}[\mathsf{CG}]|. \end{aligned}$$

$\mathcal{D}_{i,i+1}$ is a distinguisher of the DDH problem on an input (g_1, U, V, W) whose base is g_1. $\mathcal{D}_{i,i+1}$ uses the instance by the random self-reducibility of the DDH problem. Since $\mathcal{D}_{i,i+1}$ knows the passwords of all users, it can easily answer queries made by $\mathcal{S}$. Hence, $\mathcal{D}_{i,i+1}$ perfectly simulates $\mathbf{Exp}_\mathsf{i}$ or $\mathbf{Exp}_{\mathsf{i+1}}$ depending on whether or not (g_1, U, V, W) is a DDH triple. That is, if (g_1, U, V, W) is the DDH triple, $\mathcal{D}_{i,i+1}$ simulates $\mathbf{Exp}_\mathsf{i}$ or $\mathbf{Exp}_{\mathsf{i+1}}$ otherwise. We assume $\mathcal{S}$ making only a single $\mathsf{Execute}$ query and a single $\mathsf{SendUser}$ query and show that $|\Pr_0[\mathsf{CG}] - \Pr_n[\mathsf{CG}]| \leq |\mathcal{G}| \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)$.

$\mathcal{D}_{0,1}(g_1, U, V, W)$

1. Select the passwords for all users in $\mathcal{G}$ executing $\mathcal{PG}(1^\kappa)$ and give $\mathcal{S}$ the passwords for all users in $\mathcal{G}$.
2. For $\mathsf{Execute/SendUser}_3(U_i^s, m)$, randomly select σ, s_1, s_2, and compute $\mathsf{U} = U^\sigma \cdot g_1^{s_1}$, $\mathsf{V} = V \cdot g_1^{s_2}$. Compute $\tau_{1,S} = \mathsf{MAC.G}_{k_1}(U_1\|S\|X_1\|Y_1\|\mathsf{U})$, $\tau_{2,S} = \mathsf{MAC.G}_{k_2}(U_2\|S\|X_2\|Y_2\|\mathsf{V})$, $\tau_{3,S} = \mathsf{MAC.G}_{k_3}(U_3\|S\|X_2\|Y_2\|g_1^{r_3})$,...,$\tau_{n,S} = \mathsf{MAC.G}_{k_n}(U_n\|S\|X_n\|Y_n\|g_1^{r_n})$.
3. For $\mathsf{Execute/SendUser}_5(\mathcal{G}_u, m)$, compute $\mathsf{W} = W^\sigma \cdot U^{\sigma s_2} \cdot V^{s_1} \cdot g_1^{s_1 s_2}$ and $\beta_1 = \frac{\mathsf{W}}{\mathsf{U}^{r_n}}, \beta_2 = \frac{\mathsf{V}^{r_3}}{\mathsf{W}}, \ldots, \beta_n = \frac{\mathsf{U}^{r_n}}{g_1^{r_{n-1} r_n}}$, and return $(\beta_1, \sigma_{2,1})||...||(\beta_n, \sigma_{2,n})$.
4. Compute $\gamma_1 = (\mathsf{U}^{r_n})^n \cdot (\beta_1)^{n-1} \cdots \beta_{n-1}, \gamma_2 = (\mathsf{W})^n \cdot (\beta_2)^{n-1} \cdots \beta_n, \gamma_3 = (\mathsf{V}^{r_3})^n \cdot (\beta_3)^{n-1} \cdots \beta_1, ...,$ $\gamma_n = (g_1^{r_{n-1} r_n})^n \cdot (\beta_n)^{n-1} \cdots \beta_{n+2}$.
5. For $\mathsf{Reveal}(U_i^s)$, if U_i^s is a terminated instance and has a session key sk_i with all other users, return sk_i.
6. For $\mathsf{TestGroup}(\mathcal{G}_u)$, check whether all users in $\mathcal{G}_u$ have the same session key. If this check fails, return $\perp$. If this check is correct and if U_i^s is *fresh*, return sk without coin flipping.
7. If $\mathcal{S}$ terminates with b', return b' and halt.

$\mathcal{D}_{1,2}(g_1, U, V, W)$: Same as $\mathcal{D}_{0,1}$ except following points.

1. For $\mathsf{Execute/SendUser}_3(U_i^s, m)$, randomly select $\sigma, s_1, s_2, w_{1,2}$, and compute $\mathsf{U} = U^\sigma \cdot g_1^{s_1}$, $\mathsf{V} = V \cdot g_1^{s_2}$. Compute $\tau_{1,S} = \mathsf{MAC.G}_{k_1}(U_1 \| S \| X_1 \| Y_1 \| g_1^{r_1})$, $\tau_{2,S} = \mathsf{MAC.G}_{k_2}(U_2 \| S \| X_2 \| Y_2 \| \mathsf{U})$, $\tau_{3,S} = \mathsf{MAC.G}_{k_3}(U_3 \| S \| X_2 \| Y_2 \| \mathsf{V})$,...,$\tau_{n,S} = \mathsf{MAC.G}_{k_n}(U_n \| S \| X_n \| Y_n \| g_1^{r_n})$.
2. For $\mathsf{Execute/SendUser}_5(\mathcal{G}_u, m)$, compute $\mathsf{W} = W^\sigma \cdot U^{\sigma s_2} \cdot V^{s_1} \cdot g_1^{s_1 s_2}$ and $\beta_1 = \frac{g_1^{w_{1,2}}}{g_1^{r_n r_1}}, \beta_2 = \frac{\mathsf{W}}{g_1^{w_{1,2}}}, \beta_3 = \frac{\mathsf{V}^{r_4}}{\mathsf{W}}, \ldots, \frac{g_1^{r_1 r_n}}{g_1^{r_{n-1} r_n}}$, and return $(\beta_1, \sigma_{2,1})||...||(\beta_n, \sigma_{2,n})$.
3. Compute $\gamma_1 = (g_1^{r_n r_1})^n \cdot (\beta_1)^{n-1} \cdots \beta_{n-1}, \gamma_2 = (g_1^{w_{1,2}})^n \cdot (\beta_2)^{n-1} \cdots \beta_n, \gamma_3 = (\mathsf{W})^n \cdot (\beta_3)^{n-1} \cdots \beta_1$,...,$\gamma_n = (g_1^{r_{n-1} r_n})^n \cdot (\beta_n)^{n-1} \cdots \beta_{n+2}$.
4. If $\mathcal{S}$ terminates with b', return b' and halt.

Continuing in this way, we get the following distinguisher.

$\underline{\mathcal{D}_{n-1,n}(g_1, U, V, W)}$: Same as $\mathcal{D}_{n-2,n-1}$ except following points.

1. For $\mathsf{Execute/SendUser}_3(U_i^s, m)$, randomly select σ, s_1, s_2, and compute $\mathsf{U} = U^\sigma \cdot g_1^{s_1}$, $\mathsf{V} = V \cdot g_1^{s_2}$. Compute $\tau_{1,S} = \mathsf{MAC.G}_{k_1}(U_1 \| S \| X_1 \| Y_1 \| \mathsf{V})$,$\tau_{2,S} = \mathsf{MAC.G}_{k_2}(U_2 \| S \| X_2 \| Y_2 \| g_1^{r_2})$,...,$\tau_{n-1,S} = \mathsf{MAC.G}_{k_{n-1}}(U_{n-1} \| S \| X_{n-1} \| Y_{n-1} \| g_1^{r_{n-1}})$,$\tau_{n,S} = \mathsf{MAC.G}_{k_n}(U_n \| S \| X_n \| Y_n \| \mathsf{U})$.
2. For $\mathsf{Execute/SendUser}_5(\mathcal{G}_u, m)$, compute $\mathsf{W} = W^\sigma \cdot U^{\sigma s_2} \cdot V^{s_1} \cdot g_1^{s_1 s_2}$ and $\beta_1 = \frac{g_1^{w_{1,2}}}{\mathsf{W}}, \beta_2 = \frac{g_1^{w_{2,3}}}{g_1^{w_{1,2}}}, \ldots, \beta_n = \frac{\mathsf{W}}{g_1^{w_{n-1,n}}}$, and return $(\beta_1, \sigma_{2,1})||...||(\beta_n, \sigma_{2,n})$.
3. Compute $\gamma_1 = (\mathsf{W})^n \cdot (\beta_1)^{n-1} \cdots \beta_{n-1}, \gamma_2 = (g_1^{w_{1,2}})^n \cdot (\beta_2)^{n-1} \cdots \beta_n, ..., \gamma_n = (g_1^{w_{1,2}})^n \cdot (\beta_n)^{n-1} \cdots \beta_{n+2}$.
4. If $\mathcal{S}$ terminates with b', return b' and halt.

Consider the case that the message α_i of $\mathsf{SendUser}_4(U_i^s, m)$ has not been previously computed by $\mathcal{D}_{i,i+1}$. Even though the instance involved in the $\mathsf{SendUser}_4$ accepts itself, its partners may not be oracle instances. Thus a $\mathsf{TestGroup}$ query involving this instance may always return the invalid symbol $\perp$. Finally, we have via a standard hybrid argument that

$$|\Pr_0[\mathsf{CG}] - \Pr_n[\mathsf{CG}]| \leq N_s \cdot |\mathcal{G}| \cdot \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t). \tag{16}$$

In game $\mathbf{Exp_n}$, the values $w_{1,2}, ..., w_{n,1}$ are expressed as n equations, $log_{g_1}\beta_1 = w_{1,2} - w_{n,1}, ...,$ $log_{g_1}\beta_n = w_{n,1} - w_{n-1,n}$ of which only $n-1$ are linearly independent. The secret key γ is equivalently expressed as $log_{g_1}\gamma = w_{1,2} + w_{2,3} + \ldots + w_{n,1}$ which is linearly independent from the n equations. Thus the value of $sk = \mathcal{F}_\gamma(\mathcal{G}_u||\mathsf{sid})$ is independent from $(\beta_1, ..., \beta_n)$, i.e., this implies that the bit b used by the Test oracle cannot be guessed by the adversary better than at random for each attempt:

$$\Pr_n[\mathsf{CG}] = \frac{1}{2}. \tag{17}$$

We define an auxiliary game $\mathbf{Exp_{n^\star}}$ similar to $\mathbf{Exp_n}$ except for the way the session keys are computed. In $\mathbf{Exp_n}$, the session keys are computed by using a pseudo random function family $\mathcal{F}$, while in $\mathbf{Exp_{n^\star}}$, the session keys are computed by using a random function g.

$\underline{\mathbf{Exp_{n^\star}}}$: Same as $\mathbf{Exp_n}$ except the following points, after having chosen a random function g.

1. Computes the session keys as $sk_1 = g(\mathcal{G}_u||\mathsf{sid}), ..., sk_n = g(\mathcal{U}||\mathsf{sid})$.

Consider a distinguisher $\mathcal{D}$ to break pseudo randomness of a pseudo random function family $\mathcal{F}$ using the difference of the advantages of an adversary $\mathcal{S}$ in $\mathbf{Exp_n}$ and $\mathbf{Exp_{n^\star}}$. $\mathcal{D}$ is given an oracle function $f(\cdot)$ in the experiment of pseudo randomness of the function family $\mathcal{F}$. $\mathcal{D}$ simulates $\mathbf{Exp_n}$ or $\mathbf{Exp_{n^\star}}$ depending on whether $f(\cdot)$ is a function from $\mathcal{F}$ or not. The more concrete description of $\mathcal{D}$ is as follows:

$\underline{\mathcal{D}^{f(\cdot)}}$

1. For all oracle queries of $\mathcal{S}$, answer them as in $\mathbf{Exp_n}$ by using an oracle function $f(\cdot)$ instead of $\mathcal{F}$ to make session keys.
2. If $\mathcal{S}$ terminates with b', return b' and halt.

The advantage of $\mathcal{D}$ to break the pseudo random function family (with probability related to $\mathcal{S}$' success probability) is as the follow:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h) \;&= |\Pr[K \overset{R}{\leftarrow} Keys(\mathcal{F}) : \quad \mathcal{F}^{\mathcal{F}_K(\cdot)} = 1] - \Pr[g \overset{R}{\leftarrow} Rand^{D\to R} : \mathcal{F}^{g(\cdot)} = 1]| \\ &= |\Pr_n[\mathsf{CG}] - \Pr_{n^\star}[\mathsf{CG}]|.\end{aligned} \tag{18}$$

From Equations (16),(17) and (18) we get that

$$\mathsf{Adv}^{\mathsf{kss}}_{\mathcal{PAMKE}2}(k, t, q_{ex}, q^U_{se}) \leq 2N_s \cdot |\mathcal{G}| \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + 2\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\kappa, T, q, h).$$

From Equation (11), Claim 3, Claim 4 and Claim 5 yield the statement of the theorem.

C Explicit Authentication

$\mathcal{PAMKE}1$ and $\mathcal{PAMKE}2$ are PAKE protocols with implicit authentication. Another notion is *explicit authentication*, which guarantees to each user that it actually shares the same session key with all the others. To convert the protocols with implicit authentication into a protocol provides explicit authentication, we use the well-known approach which generates an "authenticator" for the other users by using a message authentication code (MAC) keyed by the shared session key [17]. We now present the modified protocols, $\mathcal{PAMKE}1'$ and $\mathcal{PAMKE}2'$, providing explicit authentication. $\mathcal{PAMKE}1'$ is the protocol for $\mathcal{PAMKE}1$, and $\mathcal{PAMKE}2'$ is the protocol for $\mathcal{PAMKE}2$.

C.1 The $\mathcal{PAMKE}1'$ protocol

This protocol is executed right after the session key distribution phase of $\mathcal{PAMKE}1$.

Key computation. Upon receiving $(U_1||K_1||...||U_{|\mathcal{G}_u|}||K_{|\mathcal{G}_u|})$, each user $U_i \in \mathcal{G}_u$ computes $K = K_i \oplus H(\mathcal{G}_u||k_i)$ and broadcasts $\tau_i = \mathsf{MAC.G}_K(U_i||\mathcal{G}_u||K_1||\ldots||K_{|\mathcal{G}_u|})$.

Key confirmation: Upon receiving τ_i, each user U_i sets $\mathcal{T}_u$ which is the set of reached tags, including his tag and arranging in lexical order of the user identities. Each user U_i checks the validity of all tags in $\mathcal{T}_u$. If all in $\mathcal{T}_u$ are valid, each user U_i computes the session key as $sk_i = F_K(\mathcal{G}_u||\mathsf{sid})$, where $\mathsf{sid} = (K_1||\ldots||K_{|\mathcal{G}_u|}||\mathcal{T}_u)$.

C.2 The $\mathcal{PAMKE}2'$ protocol

This protocol is executed right after the MAC key distribution and MAC-authenticated multi-party key exchange phase of $\mathcal{PAMKE}2$.

Key computation. Each user $U_i \in \mathcal{G}_u$ broadcasts $\sigma_{i,3} = \mathsf{MAC.G}_{\gamma_i}(\mathcal{G}_u||\mathsf{K}||\alpha||\sigma_1||\beta||\sigma_2)$, where $\mathcal{G}_u = (U_1, \ldots, U_{|\mathcal{G}_u|})$, $\mathsf{K} = (K_1, \ldots, K_{|\mathcal{G}_u|})$, $\alpha = (\alpha_1, \ldots, \alpha_{|\mathcal{G}_u|})$, $\sigma_1 = (\sigma_{1,1}, \ldots, \sigma_{1,|\mathcal{G}_u|})$, $\beta = (\beta_1, \ldots, \beta_{|\mathcal{G}_u|})$, and $\sigma_2 = (\sigma_{2,1}, \ldots, \sigma_{2,|\mathcal{G}_u|})$.

Key confirmation. Upon receiving $\sigma_{j,3}$ $(j \neq i)$, each user U_i sets $\mathcal{T}_u$ as in $\mathcal{PAMKE}1'$. Each user U_i checks the validity of all tags in $\mathcal{T}_u$. If all are valid, each user U_i computes the session key as $sk_i = F_{\gamma_i}(\mathcal{G}_u||\mathsf{sid})$, where $\mathsf{sid} = (\mathsf{K}||\alpha||\sigma_1||\beta||\sigma_2||\sigma_3)$ and $\sigma_3 = (\sigma_{3,1}, ..., \sigma_{3,|\mathcal{G}_u|})$.

D Figures

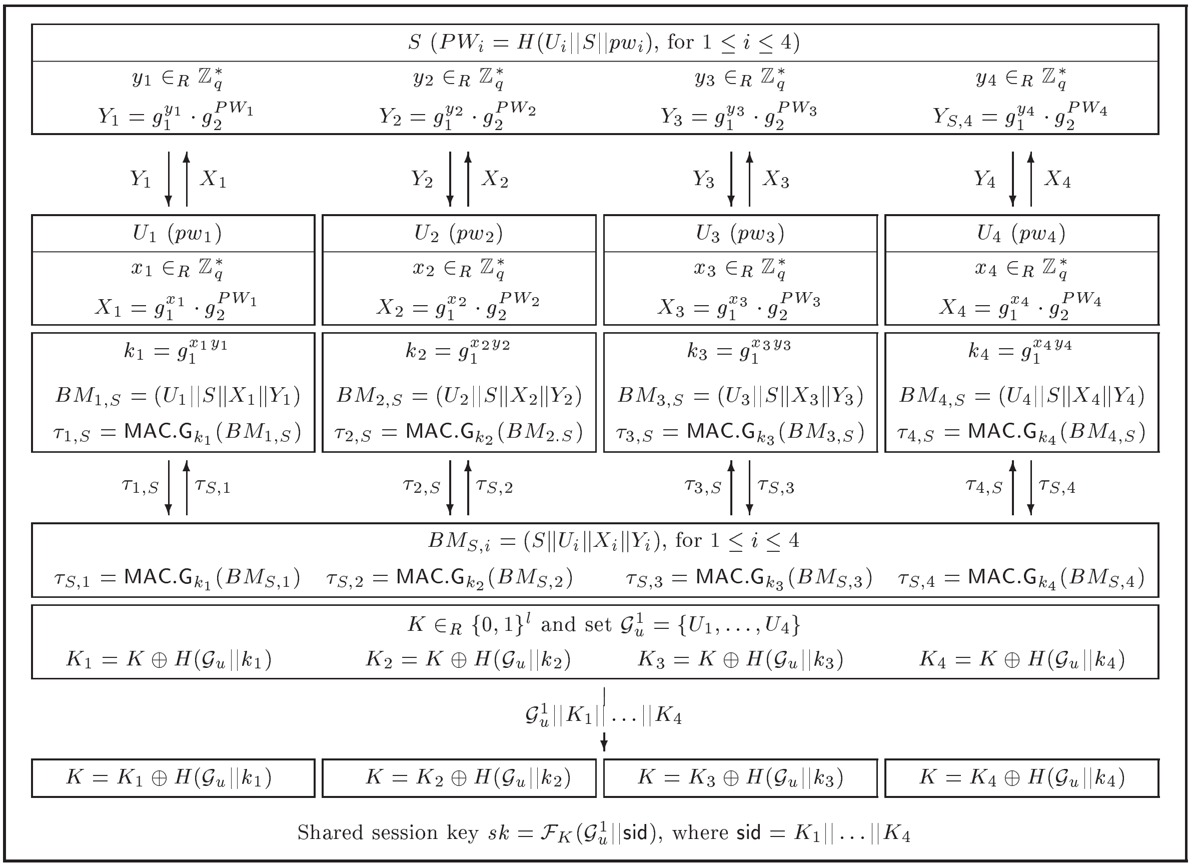
S $(PW_i = H(U_i||S||pw_i)$, for $1 \leq i \leq 4)$
$y_1 \in_R \mathbb{Z}_q^*$
$Y_1 = g_1^{y_1} \cdot g_2^{PW_1}$
$y_2 \in_R \mathbb{Z}_q^*$
$Y_2 = g_1^{y_2} \cdot g_2^{PW_2}$
$y_3 \in_R \mathbb{Z}_q^*$
$Y_3 = g_1^{y_3} \cdot g_2^{PW_3}$
$y_4 \in_R \mathbb{Z}_q^*$
$Y_{S,4} = g_1^{y_4} \cdot g_2^{PW_4}$
Y_1
X_1
Y_2
X_2
Y_3
X_3
Y_4
X_4
U_1 (pw_1)
$x_1 \in_R \mathbb{Z}_q^*$
$X_1 = g_1^{x_1} \cdot g_2^{PW_1}$
U_2 (pw_2)
$x_2 \in_R \mathbb{Z}_q^*$
$X_2 = g_1^{x_2} \cdot g_2^{PW_2}$
U_3 (pw_3)
$x_3 \in_R \mathbb{Z}_q^*$
$X_3 = g_1^{x_3} \cdot g_2^{PW_3}$
U_4 (pw_4)
$x_4 \in_R \mathbb{Z}_q^*$
$X_4 = g_1^{x_4} \cdot g_2^{PW_4}$
$k_1 = g_1^{x_1 y_1}$
$BM_{1,S} = (U_1||S\|X_1\|Y_1)$
$\tau_{1,S} = \mathsf{MAC.G}_{k_1}(BM_{1,S})$
$k_2 = g_1^{x_2 y_2}$
$BM_{2,S} = (U_2||S\|X_2\|Y_2)$
$\tau_{2,S} = \mathsf{MAC.G}_{k_2}(BM_{2.S})$
$k_3 = g_1^{x_3 y_3}$
$BM_{3,S} = (U_3||S\|X_3\|Y_3)$
$\tau_{3,S} = \mathsf{MAC.G}_{k_3}(BM_{3,S})$
$k_4 = g_1^{x_4 y_4}$
$BM_{4,S} = (U_4||S\|X_4\|Y_4)$
$\tau_{4,S} = \mathsf{MAC.G}_{k_4}(BM_{4,S})$
$\tau_{1,S}$
$\tau_{S,1}$
$\tau_{2,S}$
$\tau_{S,2}$
$\tau_{3,S}$
$\tau_{S,3}$
$\tau_{4,S}$
$\tau_{S,4}$
$BM_{S,i} = (S\|U_i\|X_i\|Y_i)$, for $1 \leq i \leq 4$
$\tau_{S,1} = \mathsf{MAC.G}_{k_1}(BM_{S,1})$
$\tau_{S,2} = \mathsf{MAC.G}_{k_2}(BM_{S,2})$
$\tau_{S,3} = \mathsf{MAC.G}_{k_3}(BM_{S,3})$
$\tau_{S,4} = \mathsf{MAC.G}_{k_4}(BM_{S,4})$
$K \in_R \{0,1\}^l$ and set $\mathcal{G}_u^1 = \{U_1, \ldots, U_4\}$
$K_1 = K \oplus H(\mathcal{G}_u||k_1)$
$K_2 = K \oplus H(\mathcal{G}_u||k_2)$
$K_3 = K \oplus H(\mathcal{G}_u||k_3)$
$K_4 = K \oplus H(\mathcal{G}_u||k_4)$
$\mathcal{G}_u^1||K_1||\ldots||K_4$
$K = K_1 \oplus H(\mathcal{G}_u||k_1)$
$K = K_2 \oplus H(\mathcal{G}_u||k_2)$
$K = K_3 \oplus H(\mathcal{G}_u||k_3)$
$K = K_4 \oplus H(\mathcal{G}_u||k_4)$
Shared session key $sk = \mathcal{F}_K(\mathcal{G}_u^1||\mathsf{sid})$, where $\mathsf{sid} = K_1||\ldots||K_4$

Fig. 1. An execution of $\mathcal{PAMKE}1$ with $\mathcal{G}_u = \{U_1, U_2, U_3, U_4\}$.

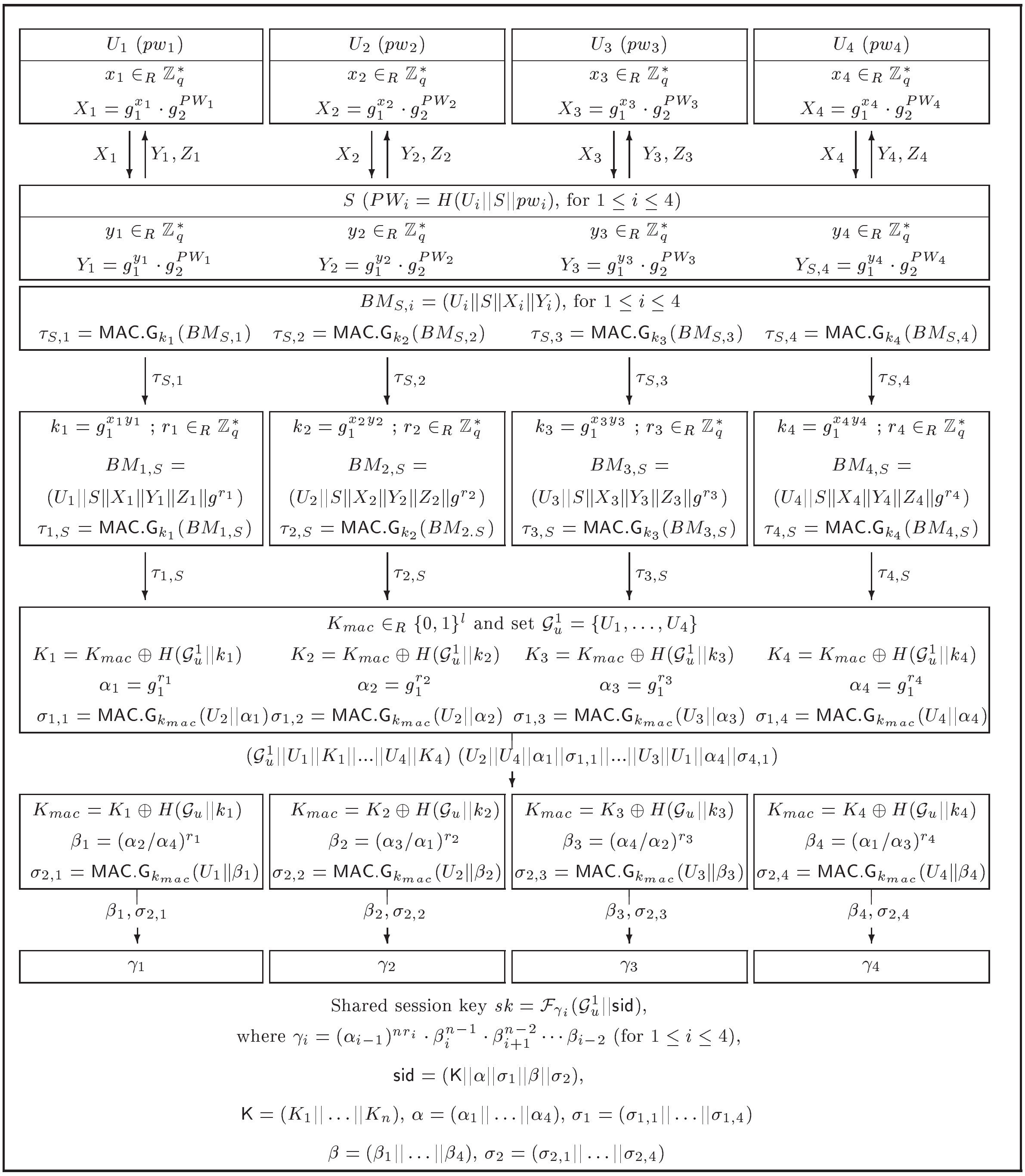
$U_1\ (pw_1)$
$U_2\ (pw_2)$
$U_3\ (pw_3)$
$U_4\ (pw_4)$
$x_1 \in_R \mathbb{Z}_q^*$
$x_2 \in_R \mathbb{Z}_q^*$
$x_3 \in_R \mathbb{Z}_q^*$
$x_4 \in_R \mathbb{Z}_q^*$
$X_1 = g_1^{x_1} \cdot g_2^{PW_1}$
$X_2 = g_1^{x_2} \cdot g_2^{PW_2}$
$X_3 = g_1^{x_3} \cdot g_2^{PW_3}$
$X_4 = g_1^{x_4} \cdot g_2^{PW_4}$
X_1
Y_1, Z_1
X_2
Y_2, Z_2
X_3
Y_3, Z_3
X_4
Y_4, Z_4
$S\ (PW_i = H(U_i||S||pw_i)$, for $1 \leq i \leq 4)$
$y_1 \in_R \mathbb{Z}_q^*$
$y_2 \in_R \mathbb{Z}_q^*$
$y_3 \in_R \mathbb{Z}_q^*$
$y_4 \in_R \mathbb{Z}_q^*$
$Y_1 = g_1^{y_1} \cdot g_2^{PW_1}$
$Y_2 = g_1^{y_2} \cdot g_2^{PW_2}$
$Y_3 = g_1^{y_3} \cdot g_2^{PW_3}$
$Y_{S,4} = g_1^{y_4} \cdot g_2^{PW_4}$
$BM_{S,i} = (U_i \| S \| X_i \| Y_i)$, for $1 \leq i \leq 4$
$\tau_{S,1} = \mathsf{MAC.G}_{k_1}(BM_{S,1})$
$\tau_{S,2} = \mathsf{MAC.G}_{k_2}(BM_{S,2})$
$\tau_{S,3} = \mathsf{MAC.G}_{k_3}(BM_{S,3})$
$\tau_{S,4} = \mathsf{MAC.G}_{k_4}(BM_{S,4})$
$\tau_{S,1}$
$\tau_{S,2}$
$\tau_{S,3}$
$\tau_{S,4}$
$k_1 = g_1^{x_1 y_1}$; $r_1 \in_R \mathbb{Z}_q^*$
$k_2 = g_1^{x_2 y_2}$; $r_2 \in_R \mathbb{Z}_q^*$
$k_3 = g_1^{x_3 y_3}$; $r_3 \in_R \mathbb{Z}_q^*$
$k_4 = g_1^{x_4 y_4}$; $r_4 \in_R \mathbb{Z}_q^*$
$BM_{1,S} =$
$BM_{2,S} =$
$BM_{3,S} =$
$BM_{4,S} =$
$(U_1||S\|X_1\|Y_1\|Z_1\|g^{r_1})$
$(U_2||S\|X_2\|Y_2\|Z_2\|g^{r_2})$
$(U_3||S\|X_3\|Y_3\|Z_3\|g^{r_3})$
$(U_4||S\|X_4\|Y_4\|Z_4\|g^{r_4})$
$\tau_{1,S} = \mathsf{MAC.G}_{k_1}(BM_{1,S})$
$\tau_{2,S} = \mathsf{MAC.G}_{k_2}(BM_{2.S})$
$\tau_{3,S} = \mathsf{MAC.G}_{k_3}(BM_{3,S})$
$\tau_{4,S} = \mathsf{MAC.G}_{k_4}(BM_{4,S})$
$\tau_{1,S}$
$\tau_{2,S}$
$\tau_{3,S}$
$\tau_{4,S}$
$K_{mac} \in_R \{0,1\}^l$ and set $\mathcal{G}_u^1 = \{U_1, \ldots, U_4\}$
$K_1 = K_{mac} \oplus H(\mathcal{G}_u^1||k_1)$
$K_2 = K_{mac} \oplus H(\mathcal{G}_u^1||k_2)$
$K_3 = K_{mac} \oplus H(\mathcal{G}_u^1||k_3)$
$K_4 = K_{mac} \oplus H(\mathcal{G}_u^1||k_4)$
$\alpha_1 = g_1^{r_1}$
$\alpha_2 = g_1^{r_2}$
$\alpha_3 = g_1^{r_3}$
$\alpha_4 = g_1^{r_4}$
$\sigma_{1,1} = \mathsf{MAC.G}_{k_{mac}}(U_2||\alpha_1)\sigma_{1,2} = \mathsf{MAC.G}_{k_{mac}}(U_2||\alpha_2)$
$\sigma_{1,3} = \mathsf{MAC.G}_{k_{mac}}(U_3||\alpha_3)$
$\sigma_{1,4} = \mathsf{MAC.G}_{k_{mac}}(U_4||\alpha_4)$
$(\mathcal{G}_u^1||U_1||K_1||...||U_4||K_4)\ (U_2||U_4||\alpha_1||\sigma_{1,1}||...||U_3||U_1||\alpha_4||\sigma_{4,1})$
$K_{mac} = K_1 \oplus H(\mathcal{G}_u||k_1)$
$K_{mac} = K_2 \oplus H(\mathcal{G}_u||k_2)$
$K_{mac} = K_3 \oplus H(\mathcal{G}_u||k_3)$
$K_{mac} = K_4 \oplus H(\mathcal{G}_u||k_4)$
$\beta_1 = (\alpha_2/\alpha_4)^{r_1}$
$\beta_2 = (\alpha_3/\alpha_1)^{r_2}$
$\beta_3 = (\alpha_4/\alpha_2)^{r_3}$
$\beta_4 = (\alpha_1/\alpha_3)^{r_4}$
$\sigma_{2,1} = \mathsf{MAC.G}_{k_{mac}}(U_1\|\beta_1)$
$\sigma_{2,2} = \mathsf{MAC.G}_{k_{mac}}(U_2\|\beta_2)$
$\sigma_{2,3} = \mathsf{MAC.G}_{k_{mac}}(U_3\|\beta_3)$
$\sigma_{2,4} = \mathsf{MAC.G}_{k_{mac}}(U_4\|\beta_4)$
$\beta_1, \sigma_{2,1}$
$\beta_2, \sigma_{2,2}$
$\beta_3, \sigma_{2,3}$
$\beta_4, \sigma_{2,4}$
γ_1
γ_2
γ_3
γ_4
Shared session key $sk = \mathcal{F}_{\gamma_i}(\mathcal{G}_u^1||\mathsf{sid})$,
where $\gamma_i = (\alpha_{i-1})^{nr_i} \cdot \beta_i^{n-1} \cdot \beta_{i+1}^{n-2} \cdots \beta_{i-2}$ (for $1 \leq i \leq 4$),
$\mathsf{sid} = (\mathsf{K}||\alpha||\sigma_1||\beta||\sigma_2)$,
$\mathsf{K} = (K_1|| \ldots ||K_n)$, $\alpha = (\alpha_1|| \ldots ||\alpha_4)$, $\sigma_1 = (\sigma_{1,1}|| \ldots ||\sigma_{1,4})$
$\beta = (\beta_1|| \ldots ||\beta_4)$, $\sigma_2 = (\sigma_{2,1}|| \ldots ||\sigma_{2,4})$

Fig. 2. An execution of $\mathcal{PAMKE}2$ with $\mathcal{G}_u = \{U_1, U_2, U_3, U_4\}$.