社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－CSEC－34 (47)
2006／7／21

Password-Authenticated Key Exchange for Multi-Party with Different Passwords Using a Constant Number of Rounds

Jeong Ok KWON†‡ Kouichi SAKURAI‡ and Dong Hoon LEE†

† Graduate School of Information Security CIST, Korea university
1, 5-Ga, Anam-dong Sungbuk-gu, Seoul, 136-701 Korea

‡ Department of Computer Science and Communication Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-0053 Japan

E-mail: †{pitapat, donghlee}@korea.ac.kr, ‡sakurai@csce.kyushu-u.ac.jp

Abstract Multi-party password-authenticated key exchange (PAKE) with different passwords allows the users of a group to agree on a common session key with their *different* passwords by the help of a server. In this setting, a user shares a password only with the server, but not between the users. In this paper, we present two multi-party PAKE protocols. The suggested protocols are provably-secure in the *standard* model. Our first protocol is designed to provide forward secrecy and security against known-key attacks. The second protocol is designed to additionally provide *key secrecy against the server* which means that even the server can not know the session keys of the users of a group. The suggested protocols require a *constant* number of rounds.

Keyword Password-authenticated key exchange, key exchange with different passwords, key secrecy against the server

1. Introduction

1.1. 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 for two or more specified parties to share a secret key using *only* an easily memorable password. Hence, PAKE protocols do not require that each party holds some devices like smart cards or hardware tokens. In this point of view, PAKE provides convenience and mobility to people. Protocols for PAKE can be used in several environments, especially in networks where a security infrastructure like PKI (Public-Key Infrastructure) is not deployed. This implies that a PAKE protocol makes it possible for parties who have no public keys to establish a session key. Because PAKE protocols provide a novel way to authenticate parties and derive high-quality cryptographic keys from low-grade passwords, PAKE has received significant attention.

1.2. Multi-party PAKE with different passwords

Several multi-party PAKE protocols have already been constructed so far. However most of them assume a group of users share the same 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 to share a password for a group, the number of passwords that the user has to memorize linearly increases. This is impractical since it is difficult for a user to remember many passwords. In order to solve this problem, we consider multi-party PAKE with different passwords, where a user shares a password only with a trusted server. In this setting, 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. The disadvantage is that the server has to participate in the protocol run to help a group of users authenticate each other.

1.3. Dictionary attacks in server-aided PAKE

Compared to other security models, the most distinguishable characteristic of the PAKE security model is that the model must incorporate dictionary attacks. The dictionary attacks are possible due to the low entropy of the password space. In practice, a password consists of 4 or 8 characters such as natural language phrase to be easily memorized. The set of these probable passwords is small, so 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 can not become a serious threat because the on-line attacks can be easily detected and thwarted by counting access failures.

In the server-aided PAKE protocols, we more carefully consider the online dictionary attacks, because an adversary would get some redundancy information using the server as a password verification oracle. If a failed guess can not be detected and logged by the server, the attacks are called *undetectable* on-line dictionary attacks. 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 such that using the server as the password verification oracle are prohibited. Even if there exists a little bit of redundancy information in the protocol messages, an adversary will 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 users' passwords using its own information. The main security goal of PAKE schemes is to restrict the adversaries to the on-line dictionary attacks only. If a PAKE scheme is secure then an adversary can not obtain any advantage in guessing the passwords and the session keys of users through the off-line dictionary attacks.

1.4. Key secrecy with respect to 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 its choice to the users in the protocol. It can be necessary that the key secrecy is 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 a group of users establish a session key between the users.

1.5. Fault-tolerance in 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. 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 server-aided PAKE protocols fail.

2. Our Work in Relation to Prior Work

2.1. Our contributions

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

New security model for key secrecy with respect to the server. In addition to key secrecy of the session key, we define the notion of key secrecy with respect to the server. This means that the session keys shared between the users of a group should not be known to the server. This notion is an extended notion of the three-party setting in [3].

PAKE protocols proven secure in the standard model. In a wireless network, it is important to establish a session key efficiently due to the computational and communicational restrictions of the mobile devices. So non-constant-round multi-party key exchange protocols are not acceptable in the wireless networks. There is a constant-round provably-secure multi-party PAKE protocol, called "N-party EKE-M" [11]. However N-party EKE-M does not provide key secrecy with respect to the server. N-party EKE-M was conjectured secure when the block cipher is instantiated via an "ideal cipher" and the hash function is instantiated via an "ideal hash" (the so-called the random oracle model). The idealized oracle methodology may enable the design and security proof of cryptographic schemes easier and more efficient. However a secure scheme in the idealized oracle model

TABLE I

COMPARISONS OF EFFICIENCY AND SECURITY WITH THE RELATED PROTOCOLS FOR MULTI-PARTY WITH DIFFERENT PASSWORDS

Scheme	N-party EKE-M [11]	PAMKE1	PAMKE2
Round	2	3	5
Exponentiation (per user)	2	3	6
Message length (per user)	$\|p\|$	$\|p\|+\|\tau\|$	$2\|p\|+2\|\tau\|$
Security	KK	KK&FS&DOD&UDOD	KSS&KK&FS&DOD&UDOD
Assumption	IC&IH	Standard	Standard

We use a group $\square_p^*$ 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, an SC protocol is a secure protocol against server compromise, and a DOD protocol and a UDOD protocol have a mechanism for detection of detectable and undetectable on-line dictionary attacks, respectively. IC denotes the ideal cipher model and IH denotes the ideal hash model.

may be not secure in the real world if an idealized random function is instantiated with real functions [13, 25, 17, 14, 7]. 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 PAMKE1 in the standard model which provides forward secrecy but not key secrecy with respect to the server. To provide key secrecy with respect to the server, we present the second protocol, called PAMKE2.

PAMKE1 and PAMKE2 use the efficient two-party PAKE scheme suggested by Kobara et al. [20] which is secure in the standard model. In PAMKE1, the server distributes a session key selected by itself to each user of a group and hence these schemes do not provide key secrecy with respect to the server. To provide key secrecy with respect to the server, PAMKE2 makes all users of a group to contribute to the value of a session key. We compare the efficiency and the security of our protocols with the previous protocols in Table 1 where the message length is the total number of bits that each user sends during a protocol run.

2.2. Related works

Several multi-party PAKE protocols have been suggested so far [4, 9, 22, 11, 2]. But only [11] considers multi-party PAKE with different passwords (the others consider multiparty PAKE with same passwords). In [11], 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 constant rounds. Unfortunately, the N-party EKE-U protocol is vulnerable to the off-line dictionary attacks (an explicit attack is known in [27]) and the variant of N-party EKE-U in [12] to counter the attack of [27] is still vulnerable to the off-line dictionary attacks (an explicit attack is known in [26]). N-party EKE-M has been proven secure in the ideal cipher/hash model without forward secrecy, but does not provide key secrecy with respect to the server.

3. Multi-Party PAKE Protocols

In this section, we present two protocols, PAMKE1 and PAMKE2 in the multi-party setting which require only a constant number of rounds, achieve forward secrecy, and are secure against known-key attacks. PAMKE1 and PAMKE2 are designed without using any ideal function and their security are proved under the DDH assumption. PAMKE1 does not provide key secrecy with respect to servers, whereas PAMKE2 provides the key secrecy.

Networks. In the paper, we assume that there are two kinds of channels, 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. The broadcast network guarantees that all

users receive identical messages, that is, no private channel exists in the networks.

Participants. We fix nonempty sets, G of potential users of a group and S of potential servers. We assume the set G contains N users and the set S contains a single server. We consider a password-authenticated multi-party key exchange protocol in which any nonempty subset of G, G_u wants to exchange a session key and a server $S \in S$ 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.

3.1. The PAMKE1 Protocol

PAMKE1 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. For the two-party PAKE, PAMKE1 uses the two-party PAKE scheme in [20]. The scheme in [20] does not need any ideal function. Unfortunately, the security model which they use is different from the standard one and hence their result only applies to their specific model. Thus it seems that the security result can not be applied for the security analysis of our protocols. An example of an execution of PAMKE1 is shown in Fig. 1.

Public information. A finite cyclic group G of order q in $\square_p^*$. 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 G. g_1 and g_2 are generators of G both having order q, where g_1 and g_2 must be generated so that their discrete logarithmic relation is unknown. A hash function $H : \{0,1\}^* \rightarrow \square_q^*$. A message authentication code (MAC) algorithm, $M=$ *(KEY.G,MAC.G,MAC.V)*. *Mac.K* generates a key k_{mac}. Given k_{mac}, *MAC.G* computes a $\tau = MAC.G_{k_{mac}}(M)$ for a message M. *MAC.V* verifies a message-tag pair using key k_{mac}, and returns 1 if the tag is valid or 0 otherwise. F is a pseudo random function family.

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

Two-party PAKE. Each user $U_i \in G_u$ chooses a random number $x_i \in \square_q^*$, computes $X_i = g_1^{x_i} \cdot g_2^{H(U_i \| S \| pw_i)} \bmod p$ *mod p*. Each user U_i sends $(U_i \| X_i)$ to S. For each $i \in G_u$, S chooses a random number $y_i \in \square_q^*$, computes $Y_i = g_1^{y_i} \cdot g_2^{H(U_i \| S \| pw_i)} \bmod p$ S sends $(S_i \| Y_i)$ to each user Ui. Upon receiving $(S_i \| Y_i)$, each user U_i computes $k_i = (Y_i / g_2^{H(U_i \| S \| pw_i)})^{x_i} \bmod p$. Upon receiving $(U_i \| X_i)$, S analogously computes k_i.

Detection of undetectable/detectable on-line dictionary attacks. Each user U_i computes $\tau_{i,S} = MAC.G_{k_i}(U_i \| S \| X_i \| Y_i)$ and sends $(U_i \| \tau_{i,S})$ to S. For each $i \in G_u$, S computes $\tau_{S,i} = MAC.G_{k_i}(S \| U_i \| X_i \| Y_i)$ and sends $(S \| \tau_{S,i})$ to U_i. Upon receiving $(S \| \tau_{S,i})$, each user U_i computes $MAC.V_{k_i}(\tau_{S,i})$. Each user U_i halts if *MAC.V* returns 0, or moves the next step otherwise. Upon receiving $(U_i \| \tau_{i,S})$, for each $i \in G_u$, S checks the validity of $\tau_{i,S}$ using k_i. S sets a set of user identities G_u^1 that passes the MAC verification. Let $G_u^1 = \{U_1, ..., U_{|G_u^1|}\}$.

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

Key computation. Each user U_i computes the session key $sk = F_K(G_u^1 \| sid)$, where $sid = (K_1 \| ... \| K_{|G_u^1|})$.

Undetectable and detectable on-line dictionary attacks. PAMKE1 is designated to 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 is being 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, PAMKE1 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. PAMKE1 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.

3.2. Security Result

The following theorem shows that PAMKE1 is secure against off-line dictionary attacks since an adversary's capability to mount 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 G be a group in which the DDH assumption holds and F be a secure pseudo random function family. Then PAMKE1 is a secure PAKE-KK&FS protocol (A PAKE-KK&FS protocol is said to provide security against known-key attacks and forward secrecy) under the DDH assumption. Concretely,

$$\mathrm{Adv}^{\text{PAKE-KK\&FS}}_{PAMKE1}(k,t) = (4|G|+2|G|\cdot N_s)\cdot \mathrm{Adv}^{\text{DDH}}_{G}(t) + 2\mathrm{Adv}^{\text{PRF}}_{F}(k,t,q,h) + \frac{2(q_{se}^U + q_{se}^S)}{PW} + \frac{|G|(q_{ex}+q_{se}^U+q_{se}^S)^2}{q},$$

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

Proof of Theorem 1. The proof of this theorem appears in the full version of this paper.

4. The PAMKE2 Protocol

PAMKE2 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 PAMKE1, while preserving constant-round. Through this mechanism, the session key is determined not by the

server but unbiasedly by all honest group users together. PAMKE2 uses the mechanisms in PAMKE1 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, *BD*) in [11] for the MAC-authenticated multi-party key exchange. PAMKE2 is the same as in PAMKE1 except for the following points:

Detection of undetectable/detectable on-line dictionary attacks. S computes $\tau_{S,i} = MAC.G_{k_i}(U_i \| S \| X_i \| Y_i)$ and sends $(S \| \tau_{S,i})$ to U_i. Upon receiving $(S \| \tau_{S,i})$, each user U_i computes $MAC.V_{k_i}(\tau_{S,i})$. Each user U_i halts if $MAC.V$ returns 0, or moves the next step. Each user U_i chooses a random number $r_i \in \square_q^*$ and $\tau_{i,S} = MAC.G_{k_i}(U_i \| S \| X_i \| Y_i \| g^{\eta})$ and sends $(U_i \| \tau_{i,S} \| g_1^{\eta})$ to S. Upon receiving $(U_i \| \tau_{i,S} \| g_1^{\eta})$, for each $i \in G_u^1$, S checks the validity of $\tau_{i,S}$ using k_i. S sets G_u^1 as the set that passes the MAC verification by arranging the identities in lexical order. Let $|G_u^1| = n$ and $G_u^1 = \{U_1, ..., U_n\}$.

MAC key distribution and MAC-authenticated multi-party key exchange. S chooses an MAC key k_{mac} using $KEY.G$. For each $i \in G_u^1$, S computes $K_i = K \oplus H(G_u^1 \| k_i)$ and $\sigma_{1,i} = MAC.G_{k_{mac}}(U_i \| \alpha_i)$ where $\alpha_i = g_1^{\eta} \bmod p$. S broadcasts $(G_u^1 \| U_1 \| K_1 \| ... \| U_n \| K_n)$ and $(U_{j+1} \| U_{j+(n-1)} \| \alpha_j \| \sigma_{j,1})$ for $1 \le j \le n$, where $j = j \bmod n$. Upon receiving the broadcast message of S, each user U_i computes k_{mac} from K_i using G_u^1 and computes $MAC.V_{k_{mac}}(U_{i-1} \| \alpha_{i-1})$ and $MAC.V_{k_{mac}}(U_{i+1} \| \alpha_{i+1})$. Each user U_i moves the next step if both MAC verifications are correct, or halts otherwise. Each user U_i computes $\beta_i = (\alpha_{i+1} / \alpha_{i-1})^{\eta} \bmod p$ and broadcasts $(U_i \| \beta_i \| \sigma_{2,1} = MAC.G_{k_{mac}}(U_i \| \beta_i))$. Upon receiving the broadcast messages form users, S checks if one receives all broadcast messages of users in G_u^1. If not, S requests owners of missing messages to resend the message. . Upon receiving $(U_i \| \beta_i \| \sigma_{2,1})$, for each $U_i \in G_u^1 (j \ne i)$, each user U_i checks if the validity of $\sigma_{2,1}$ using k_{mac}. Each user U_i computes $\gamma_i = (\alpha_{i-1})^{n\eta} \cdot \beta_i^{n-1} \cdot \beta_{i+1}^{n-1} \cdots \beta_{i-2} \bmod p$ if all the MAC verifications are correct or halts otherwise.

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

Completeness. If everything works correctly in PAMKE2, the session key computed by U_i is $sk = F_{\gamma_i}(G_u^1 \| sid)$, where $\gamma_i = g_1^{\eta^2 + \eta^2 + \cdots + \eta^n} \bmod p$.

Fault-tolerance. PAMKE2 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.

Theorem 2. Let G be a group in which the DDH assumption holds, F be a secure pseudo random function family and M is an unforgeable MAC algorithm. Then PAMKE2 is a secure PAKE-KSS&KK&FS protocol (A PAKE-KSS&KK&FS protocol is said to provide security agianst key secrecy with respect to servers, known-key attacks and forward secrecy. Concretely,

$$\mathrm{Adv}_{PAMKE1}^{\text{PAKE-KSS\&KK\&PS}}(k,t) = (6|G|+4|G|\cdot N_s)\cdot \mathrm{Adv}_{G}^{\text{DDH}}(t) + 4\mathrm{Adv}_{F}^{\text{PRF}}(k,t,q,h) + 2|G|\cdot \mathrm{Adv}_{M}^{\text{SUF}}(k,q_{se}^{U}) + \frac{2(q_{se}^{U}+q_{se}^{S})}{PW} + \frac{|G|(q_{ex}+q_{se}^{U}+q_{se}^{S})^2}{q},$$

where the parameters are defined as in Theorem 1.

Proof of Theorem 2. The proof of this theorem appears in the full version of this paper.

5. Concluding Remarks

This paper considers multi-party PAKE with different passwords and provides the first provably-secure two constant-round protocols. 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 "oller, and D. Pointcheval, "Provably Secure Password-Based Authentication in TLS," 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," 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," Proc.of PKC05, LNCS 3386, pages 65-84, 2005.

[4] N. Asokan and P. Ginzboorg. "Key Agreement in Ad-hoc Networks," Journal of Computer Communications 23(17), pages 1627-1637, 2000.

[5] M. Bellare, R. Canetti, and H. Krawczyk. "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols," Proc. of 30th Annual ACM Symposium on Theory of Computing, ACM, pages 419-428, 1998.

[6] M. Bellare and P. Rogaway. "Entity authentication and key distribution," Proc. of CRYPTO '93, LNCS 773, pages 232-249, Springer-Verlag, 1993.

[7] M. Bellare and P. Rogaway. "Provably secure session key distribution-the three party case," Proc. of the 27th ACM Symposium on the Theory of Computing, 1995.

[8] M. Bellare, A. Boldyreva and A. Palacio. "An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem," Proc. of EUROCRYPT '04, LNCS 3027, pages 171-188, 2004.

[9] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. "Provably Authenticated Group Diffie-Hellman Key Exchange," Proc. of the 8th ACM conference on Computer and Communications Security, pages 255-264, 2001.

[10] E. Bresson, O. Chevassut, and D. Pointcheval. "Group Diffie-Hellman Key Exchange Secure Against Dictionary Attacks," Proc. of ASIACRYPT 2002, LNCS 2501, pages 497-514, Springer-Verlag, 2002.

[11] M. Burmester and Y. Desmedt. "A Secure and Efficient Conference Key Distribution System," Proc. Of EUROCRYPT '94, LNCS 950, pages 275-286, Springer-Verlag, 1995.

[12] J. W. Byun and D. H. Lee. "Password-Authenticated Key Exchange between Clients with Different Passwords," Proc. of ACNS '05, LNCS 3531, pages 75-90, 2005.

[13] J. W. Byun and D.H. Lee. "Comments on Weaknesses in Two Group Diffie-Hellman Key Exchange Protocols," IACR ePrint Archive, 2005/209, 2005.

[14] R. Canetti, O. Goldreich, and S. Halevi. "The random oracle methodology, revisited," Pro. of the 32nd Annual ACM Symposium on Theory of Computing, pages 209-218, 1998.

[15] R. Canetti, O. Goldreich and S. Halevi. "On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes," Pro. of 1st Theory of Cryptography Conference (TCC), LNCS 2951 , pages 40-57, 2004.

[16] R. Canetti and H. Krawczyk. "Universally Composable Notions of Key Exchange and Secure Channels," Proc. of Eurocrypt '02. Full version available at http://eprint.iacr.org/2002/059.

[17] D. Denning and G. M. Sacco. "Timestamps in Key Distribution Protocols," Communications of the ACM, 24(8), pages 533-536, 1981.

[18] S. Goldwasser and Y. Taumen. "On the (in)security of the Fiat-Shamir Paradigm," Proc. of STOC '03, pages 102-115, IEEE Computer Society, 2003.

[19] J. Katz, R. Ostrovsky, and M. Yung. "Forward secrecy in Password-only Key Exchange Protocols," Proc. of SCN '02, LNCS 2576, pages 29-44,Springer-Verlag, 2002.

[20] J. Katz and J. S. Shin, "Modeling Insider Attacks on Group Key-Exchange Protocols," Proc. of CCS '05, pages ??-??, 2005.

[21] J. Katz and M. Yung. "Scalable Protocol for Authenticated Group Key Exchange," Proc. of CRYPTO '03, LNCS 2729, pages 110-125, Springer-Verlag, 2003.

[22] 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/.

[23] 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.

[24] S. M. Lee, J. Y. Hwang and D. H. Lee. "Efficient Password-Based Group Key Exchange," Proc. of TrustBus '04, LNCS 3184, pages 191-199, Springer-Verlag, 2004.

[25] P. MacKenzie. "More Efficient Password

Authenticated Key Exchange," Proc. of the RSA Data Security Conference, Cryptographer's Track (RSA CT '01), LNCS 2020, pages 361-377, Springer-Verlag, 2001.

[26] A. Mayer and M. Yung. "Secure Protocol Transformation via "Expansion": From Two- Party to Groups," Proc. of 6th ACM Conference on Computer and Communication Security, ACM, pages 83-92, 1999.

[27] M. Naor and O. Reingold. "Number-Theoretic Constructions of Efficient Pseudo-Random Functions," Proc. of the 38th IEEE Symposium on Foundations of Computer Science, pages 458-467, IEEE Computer Society, 2004.

[28] J. B. Nielsen. "Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case," Proc. of CRYPTO '02, LNCS 2442, pages 111-126, 2002.

[29] Raphael C.-W. Phan and B.-M Goi. "Cryptanalysis of the N-Party Encrypted Diffie-Hellman Key Exchange Using Different Passwords," Proc. of ACNS '06, LNCS ??, pages ??-??, Springer-Verlag, 2006.

[30] Q. Tang and L. Chen. "Weaknesses in Two Group Diffie-Hellman Key Exchange Protocols," IACR ePrint Archive, 2005/197, 2005.

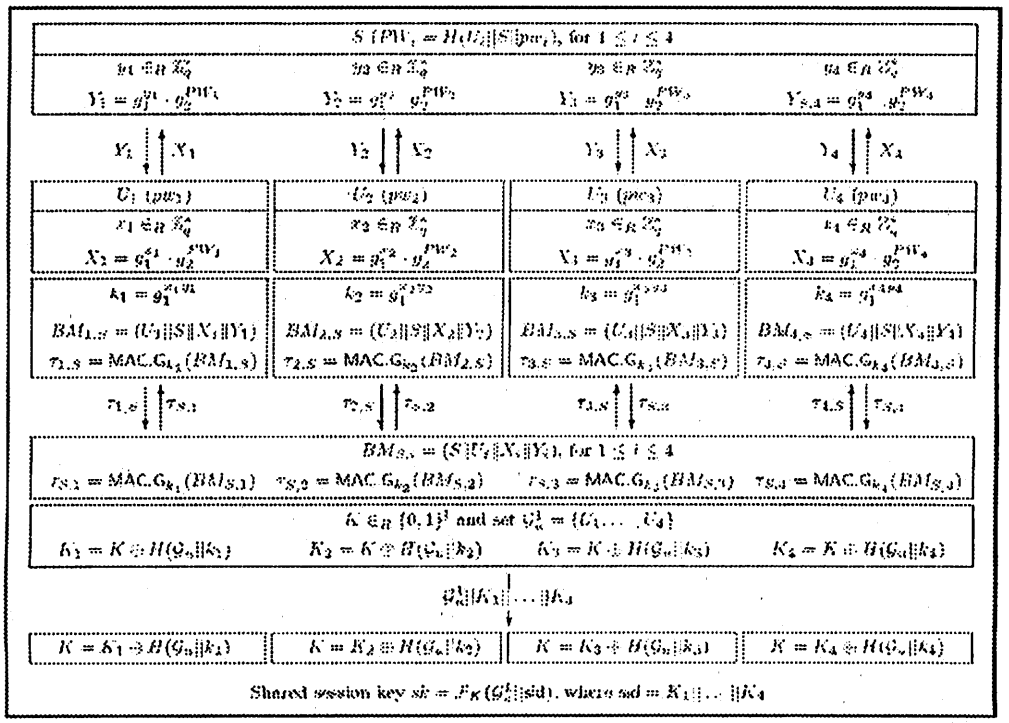

Fig. 1. An execution of PAMKE1 with $G_u = \{U_1, \ldots, U_4\}$