IPSJ Journal Vol. 50 No. 9 1942–1955 (Sep. 2009)

Regular Paper

Efficient and Strongly Secure Password-based Server Aided Key Exchange

Kazuki Yoneyama[†1]

In ACNS'06, Cliff, et al. proposed the password-based server aided key exchange (PSAKE) as one of password-based authenticated key exchanges in the three-party setting (3-party PAKE) in which two clients with different passwords exchange a session key with the help of their corresponding server. Though they also studied a strong security definition of the 3-party PAKE, their security model is not strong enough because there are desirable security properties which cannot be captured. In this paper, we define a new formal security model of the 3-party PAKE which is stronger than the previous model. Our model captures all known desirable security requirements of the 3-party PAKE, like the resistance to key-compromise impersonation, to the leakage of ephemeral private keys of servers and to the undetectable on-line dictionary attack. Also, we propose a new scheme as an improvement of PSAKE with the optimal number of rounds for a client, which is secure in the sense of our model.

1. Introduction

Recently, password-based authenticated key exchange (PAKE) protocols have received much attention as practical schemes in order to share a mutual session key secretly and reliably. Basic PAKE schemes enable two entities to authenticate each other and agree on a large session key from a human memorable password. Thus, PAKE schemes are regarded as practical key exchange schemes because entities do not have any pre-shared cryptographic symmetric key, certificate or support from a trusted third party. Such basic schemes where two entities pre-share a *common* password are classified into a model called same password-authentication (SPA) model. The SPA model is the most studied PAKE model in previous papers and is usually used for *client-to-server* key exchanges. The concept of PAKE was first introduced by Bellovin and Merritt [4] in 1992 known as encrypted key exchange (EKE). The first construction of password-only PAKE in the SPA model was proposed by Jablon [12] in 1996 known as simple password exponential key exchange (SPEKE). Formal definitions for this setting were first given by Bellare, et al. [3] and Boyko, et al. [5], and a concrete construction was also given in the random oracle (RO) model. In addition, various protocols have been proposed to achieve a secure PAKE scheme in the SPA model.

1.1 Password-based Key Exchange in the 3-party Setting

Within a variety of communication environments such as a mobile network, one of the main concerns is to establish a secure channel between clients with *different* passwords. Several schemes have been presented to provide PAKE between two entities with different passwords, called different password-authentication (DPA) models. In practice, clients prefer to remember very few passwords. Consequently, PAKE in the DPA model is useful to solve this problem. In the DPA model, entities carry out a key exchange with the assistance of an intermediate server because entities have no secret common information. So, PAKE in the DPA model is usually called password-based authenticated key exchanges in the three-party setting (3-party PAKE) and is usually used for *client-to-client* key exchanges.

Basic security requirements of the 3-party PAKE are known-key security (KS) (i.e., the session key is not compromised in the face of adversaries who have learned some other session keys), basic impersonation (BI) (i.e., the adversary cannot impersonate any honest client to the other client of the session without the client's password), and the resistance to off-line dictionary attacks (offDA). The resistance to offDA means that there is no successful adversary as follows: The adversary guesses a password and verifies his guess off-line. No participation of the server is required, so the server does not notice the attack. If his guess is wrong the adversary tries again with another password, until he finds the proper one.

Though the 3-party PAKE has been considered in previous papers [16),17)], these schemes assume a trusted intermediate server because the server can know the session key of clients. Several works [7),15),18)] considered *key privacy* against a passive server (KP) (i.e., a semi-honest server cannot know any information about the session key of clients). However, none of their schemes warrants a provable

†1 University of Electro-Communications

security. Indeed, a scheme [18] is known to be vulnerable to an undetectable on-line dictionary attack (UDonDA) [11]. The central idea of UDonDA is that an attacker guesses a password of a client, completes some computations with it and sends the server the result as a part of his request for a session key. Then, if the server cannot tell this request from the request from honest clients, the server performs some further computations on the result using the correct password of the client and responds. This response helps the attacker verify his guess. So, the server is used as an oracle without taking notice of the attack.

The first formal security definition of the 3-party PAKE (AFP model) was proposed by Abdalla, et al. [1]. They also provided a generic method to construct provably secure 3-party PAKE protocol from the 2-party PAKE. To reduce the complexity of the generic construction, the first concrete protocol of a provably secure 3-party PAKE protocol in the random oracle model is proposed in the paper [2]. Wang and Hu pointed out that schemes in papers [1),2)] are vulnerable to UDonDA, and provided a stronger definition of the 3-party PAKE (WH model) which captures the resistance to UDonDA.

Cliff, et al. [9] proposed another security definition of the 3-party PAKE (CTB model) which is an extension of the Canetti-Krawczyk model [6] for the 2-party AKE. Also, they proposed a variant of the 3-party PAKE, called the password-based server aided key exchange (PSAKE), which has the similar setting of the 3-party PAKE except the server uses password and encryption based authenticators. The encryption based authenticator in PSAKE means that a client has the server's public-key as well as the password to access the server, and the server has his private-key as well as clients' passwords. They also prove the security of PSAKE in the standard model, i.e., without any random oracle. Owing to the encryption based authenticator, PSAKE has strong security which cannot be achieved in password-only setting 3-party PAKE schemes. Indeed, PSAKE seems to be secure from the leakage of ephemeral private keys of servers (LEP) (i.e., even if the session specific ephemeral private key of *the server* in a session is compromised, the secrecy of the session key is not compromised) ⋆1 and from key-compromise impersonation (KCI) (i.e., when a client's password is compromised, this event does not enable an outside adversary to impersonate other entities to the client). LEP represents the security under the situation whereby local and temporary computations (ephemeral key) of servers may be leaked to adversaries. These leakages may also occur by sloppy usages or implementations. For example, an ephemeral key may remain in the memory for the reuse of computations in order to reduce computational costs or because of the failure to release the temporary memory area. Then, contents of the memory may be revealed by various attacks, e.g., malicious Trojan Horse programs. Thus, even if we successfully developed an exceedingly secure 3-party PAKE scheme, such a leakage might be possible. So, it is desirable that 3-party PAKE schemes satisfy LEP.

1.2 Need for New Security Models

The AFP model, the CTB model and the WH model formalize the indistinguishability of session keys against outside adversaries. However, each model has some uncaptured security requirement. For example, the AFP model and the WH model cannot grasp the notion of forward secrecy (FS) (i.e., secrecy of the past session keys after the leakage of passwords). Also, the CTB model and the WH model cannot grasp KP. Furthermore, in the AFP and the CTB model, the resistance to UDonDA is out of scope. In addition, there are some security requirements which are not captured in these models (see Section 2.2) ⋆2. Indeed, schemes in papers [1),19)] are insecure against LEP because they include the 2-party PAKE between a client and a server. In the 2-party PAKE, if an ephemeral private key of either party is leaked, the password of the party is easily derived by offDA because the session key deterministically depends on the client's ephemeral key, its static password and the communication received from the other party. Thus, the secrecy of the session key is not guaranteed. Therefore, by LEP the temporary session key is revealed and schemes in papers [1),19)] are clearly insecure against BI. Similarly, the scheme in the paper [2] is also insecure against LEP

⋆1 This property is not guaranteed when the ephemeral private key of a client of the session is leaked. In this case, the password of the client is easily derived by off-line dictionary attacks because the session key deterministically depends on the client's ephemeral key, its static password, and the communication received from other parties. Thus, the secrecy of the session key is not guaranteed. So, we only consider leakage with respect to the server.

⋆2 Indeed, the scheme in the paper [9] may be secure against UDonDA and satisfies other desirable security requirements. However, the CTB model itself does not support these requirements.

Table 1 Comparison between previous schemes and our scheme.

	setting of setup	# of rounds for a client	UDonDA	LEP
[AFP05]	password-only	2 + P	insecure	insecure
[AP05]	password-only	2	insecure	insecure
[WH06]	password-only	2 + P	secure	insecure
[CTB06]	password and public-key crypto	3	unproven	unproven
Our scheme	password and public-key crypto	2	secure	secure

Where P denotes the number of moves of a secure 2-party PAKE.

because the ephemeral private key of the server passwords can be revealed by offDAs.

1.3 Our Contribution

We define a new security model of 3-party PAKE stronger than previous models. Our model is based on the recent formal model of authenticated key exchange by LaMacchia, et al. [14]. The major difference between our model and previous models consists in adversary's available oracle queries. Specifically, revealing of static secret or ephemeral secret separately, and in adversary's capability in the target session, i.e., the adversary can obtain static secrets of all entities and ephemeral secrets of the server in the target session are different. Therefore, our model can afford resistance to complicated attacks which cannot be captured in previous models.

Also, we construct a new 3-party PAKE scheme based on the Abdalla-Pointcheval scheme in the paper [2]. Our scheme has the same setting as PSAKE (i.e., use of public-key crypto). Also, our scheme only needs the optimal number of rounds, i.e., 2-rounds between a client and the server, as does Abdalla-Pointcheval scheme. Thus, our scheme is more efficient than general constructions in papers [1),19)] and PSAKE. Furthermore, we show that our scheme is secure in the sense of our security model and the random oracle model. While public-key encryption schemes are time-consuming, as same as PSAKE, with the help of the server's public-key crypto, our scheme can provide a strong security like the resistance to LEP and to KCI. To our knowledge, our scheme is the first 3-party PAKE scheme for which the resistance to LEP is proved.

The comparison between previous schemes and ours is shown in **Table 1**.

2. Preliminaries

2.1 3-party PAKE

3-party PAKE schemes contain three parties (two clients and a server) who will engage in the protocol. We denote the set of clients by $\mathcal{U}$ and the server by S. Let each password be pre-shared between a client and the server and be uniformly and independently chosen from a fixed low-entropy dictionary $\mathcal{D}$ of size $|\mathcal{D}|$. Note that clients do not need to share passwords with other clients. In addition, in PSAKE and our scheme, the server pre-establishes his public-key and private-key pair and releases the public-key. We denote with U^l the l^{th} instance which clients $U \in \mathcal{U}$ run. Also, we denote with S^l the l^{th} instance which the server S runs. All instances finally output the *accept* symbol and halt if their specified execution is correctly finished. The session identifier $\mathsf{sid}_P^{l_i}$ of an instance P^{l_i} is represented via matching conversations, i.e., concatenations of messages which are sent and received between clients in the session, along with their identity strings, (initialized as *null*). Note that, we say that two instances $P_i^{l_i}$ and $P_j^{l_j}$ are partnered if both $P_i^{l_i}$ and $P_j^{l_j}$ output *accept*, both $P_i^{l_i}$ and $P_j^{l_j}$ share the same sid but not *null*, and the partner identification set for $P_i^{l_i}$ coincides with the one for $P_j^{l_j}$.

2.2 Security Requirement

It is desirable for 3-party PAKE protocols to possess all the following security properties:

- **Known-key security (KS):** The session key is not compromised in the face of adversaries who have learned some other session keys.
- **Forward secrecy (FS):** If static secrets (including passwords) of a client and the server is compromised, secrecy of past session keys is not compromised.
- **Key privacy against passive server (KP):** The session key cannot be distinguished from a random number by the passive server [⋆1].
- **Resistance to basic impersonation (BI):** It may be desirable that even

⋆1 Since the server knows all passwords of its clients, a malicious server is always able to impersonate one of its members and exchange a session key with another client. So, we cannot require that a malicious server cannot learn the session key.

if an adversary reveals the ephemeral private key of the server in a session, the adversary cannot impersonate any honest client to the other client of the session without the client's password.

- **Resistance to key-compromise impersonation (KCI):** When a client's password is compromised, it may be desirable that this event does not enable an outside adversary to impersonate other entities to the client.
- **Resistance to unknown-key share (UKS):** Any client C including a malicious client insider cannot interfere with the session establishment between two honest clients A and B such that at the end of the attack both parties compute the same session key (which C is not allowed to learn it), yet while A is convinced that the key is shared with B, B believes that the peer to the session has been C (definition by Diffie, et al. [10]). In addition, client A should not be able to coerce into sharing a key with any client C including a malicious client insider when in fact it thinks that it is sharing the key with the other client B (definition by Choo, et al. [8]).
- **Resistance to leakage of ephemeral private key of server (LEP):** Even if all the session specific ephemeral private keys of *the server* in a session are compromised, secrecy of the session key should not be compromised [⋆1].
- **Resistance to undetectable on-line dictionary attacks (UDonDA):** There is no successful adversary as follows: The adversary attempts to use a guessed password in an on-line transaction. He verifies the correctness of his guess by using responses from the server. If his guess fails he must start a new transaction with the server using another guessed password. By computing requests to the server which a failed guess can not be detected and logged by the server, the adversary makes the server be not able to depart an honest request from a malicious request. So, the adversary can obtain enough information to guess the password and eventually find the right password.
- **Resistance to off-line dictionary attacks (offDA):** There is no successful adversary as follows: The adversary guesses a password and verifies his guess off-line. No participation of the server is required, so the server do not notice the attack. If his guess fails the adversary tries again with another password, until he finds the proper one.

⋆1 This property is not guaranteed when the ephemeral private key of a client of the session is leaked. In this case, the password of the client is easily derived by off-line dictionary attacks because the session key deterministically depends on the client's ephemeral key, the static password, and the communication received from other parties. Thus, secrecy of the session key is not guaranteed. So, we only consider resistance to LEP of the server.

In the AFP model, FS, and resistance to KCI, LEP and UDonDA cannot be captured. First, the resistance to KCI and LEP, and FS cannot be represented because adversary capabilities do not include any query for corruption of parties in the test session. Therefore, the conditions of KCI, LEP and FS cannot be represented. Also, the resistance to UDonDA is out of scope in the AFP model. They count UDonDA in the number of queries for message modifications which are limited to certain numbers. Hence, in the AFP model, UDonDA is not discriminated from detectable on-line dictionary attacks.

Since the CTB model is the extension for the 3-party PAKE from the Canetti-Krawczyk model [6] for the 2-party AKE, the CTB model inherits uncaptured security properties from the Canetti-Krawczyk model. More specifically, in the CTB model, KP, and resistance to KCI, LEP and UDonDA cannot be captured. First, the resistance to KCI cannot be represented because adversary capabilities do not include any query for corruption of parties in the test session before completing the session. The resistance to LEP cannot be represented because adversary capabilities do not include any query to access ephemeral keys of parties in the test session. Therefore, the conditions of KCI and LEP cannot be represented. Also, KP and the resistance to UDonDA are out of scope in the CTB model. Though KP requires that a passive server (i.e., passwords of clients can be known), cannot distinguish between the real session key in a session and a random key, there is no definition which captures such a situation. Thus, KP is not guaranteed even if the security in the CTB model is satisfied. In the AFP model, they count UDonDA in the number of queries for message modifications which are limited to certain numbers. Hence, in the CTB model, UDonDA is not discriminated from detectable on-line dictionary attacks.

The WH model can be regarded as the AFP model plus resistance to UDonDA. Thus, FS, and the resistance to KCI and LEP cannot be captured from the same reason as the AFP model.

3. New Model: Strong 3-party Hybrid AKE Security

3.1 Adversary Capabilities

An outside adversary or a malicious insider can obtain and modify messages on unauthenticated-links channels. Furthermore, the adversary is given oracle access to client and server instances. We remark that unlike the standard notion of an "oracle", in this model instances maintain a state which is updated as the protocol progresses.

- $\mathsf{Execute}(U_1^{l_1}, U_2^{l_2}, S^{l_3})$: This query models passive attacks. The output of this query consists of the messages that were exchanged during the honest execution of the protocol among $U_1^{l_1}$, $U_2^{l_2}$ and S^{l_3}.
- $\mathsf{SendClient}(U^l, m)$: This query models active attacks against a client. The output of this query consists of the message that the client instance U^l would generate on receipt of message m.
- $\mathsf{SendServer}(S^l, m)$: This query models active attacks against the server. The output of this query consists of the message that the server instance S^l would generate on receipt of message m.
- $\mathsf{SessionKeyReveal}(U^l)$: This query models misuses of session keys. The output of this query consists of the session key held by the client instance U^l if the session is completed for U^l. Otherwise, return $\perp$.
- $\mathsf{StaticKeyReveal}(P)$: This query models leakage of the static secret of P (i.e., the password between the client and the server, or the private information for the server). The output of this query consists of the static secret of P. Note that, the adversary is neither given full control of P not provided with any ephemeral secret information.
- $\mathsf{EphemeralKeyReveal}(P^l)$: This query models the leakage of all session-specific information (ephemeral key) used by the instance P^l. The output of this query consists of the ephemeral key of the instance P^l.
- $\mathsf{EstablishParty}(U, S, pw_U)$: This query models the adversary to register a static secret pw_U on behalf of a client. In this way the adversary totally controls that client. Clients against whom the adversary did not issue this query are called *honest*.
- $\mathsf{Test}(U^l)$: This query doesn't model the adversarial ability, but the indistinguishability of the session key. At the beginning, a hidden bit b is chosen. If no session key for the client instance U^l is defined, then return the undefined symbol $\perp$. Otherwise, return the session key for the client instance U^l if $b = 1$ or a random key from the same space if $b = 0$. Note that, the adversary can make an only Test query at any time during the experiment. The target session is called the test session.
- $\mathsf{TestPassword}(U, pw')$: This query doesn't model the adversarial ability, but no leakage of the password. If the guess password pw' is just the same as the client U's password pw, then return 1. Otherwise, return 0. Note that, the adversary can ask an only $\mathsf{TestPassword}$ query at any time during the experiment.

3.2 Definition of Indistinguishability

Firstly, we consider the notion of indistinguishability. This notion provides security properties with respect to session keys, i.e., KS, FS, KP, resistance to BI, resistance to KCI and resistance to LEP. Note that, to capture notions of FS and resistance to KCI, an adversary can obtain static keys in the test session.

The adversary is considered successful if it guesses whether the challenge is the true session key or a random key. The adversary is allowed to make $\mathsf{Execute}$, $\mathsf{SendClient}$, $\mathsf{SendServer}$, $\mathsf{SessionKeyReveal}$, $\mathsf{StaticKeyReveal}$, $\mathsf{EphemeralKeyReveal}$, $\mathsf{EstablishParty}$ and Test queries, and outputs a guess bit b'. Let Succ^{ind} denote the event that $b' = b$ where b is the random bit chosen in the $\mathsf{Test}(U^l)$ query. Note that, we restrict the adversary such that U^l and the partnered client $\bar{U}^{l'}$ of the session are honest, and none of the following conditions hold:

(1) The adversary reveals the session key of sid_U^l or of $\mathsf{sid}_{\bar{U}}^{l'}$.

(2) The adversary asks no $\mathsf{SendClient}(U^l, m)$ or $\mathsf{SendClient}(\bar{U}^{l'}, m')$ query. Then the adversary either makes queries:
- $\mathsf{EphemeralKeyReveal}(U^l)$ or
- $\mathsf{EphemeralKeyReveal}(\bar{U}^{l'})$.

(3) The adversary asks $\mathsf{SendClient}(\bar{U}^{l'}, m)$ query. Then the adversary either makes queries:
- $\mathsf{StaticKeyReveal}(U)$,
- $\mathsf{StaticKeyReveal}(S)$,

- $\mathsf{EphemeralKeyReveal}(U^i)$ for any session i or
- $\mathsf{EphemeralKeyReveal}(\bar{U}^{l'})$.

(4) The adversary asks $\mathsf{SendClient}(U^l, m)$ query. Then the adversary either makes queries:

- $\mathsf{StaticKeyReveal}(\bar{U})$,
- $\mathsf{StaticKeyReveal}(S)$,
- $\mathsf{EphemeralKeyReveal}(U^l)$ or
- $\mathsf{EphemeralKeyReveal}(\bar{U}^i)$ for any session i.

Now, the adversary $\mathcal{A}$'s advantage is formally defined by:

$$\mathsf{Adv}^{ind}(\mathcal{A}) = |2 \cdot \Pr[\mathsf{Succ}^{ind}] - 1| \quad \text{and} \quad \mathsf{Adv}^{ind}(t, R) = \max_{\mathcal{A}}\{\mathsf{Adv}^{ind}(\mathcal{A})\},$$

where the maximum is over all $\mathcal{A}$ with a time-complexity at most t and using the number of queries to its oracle at most R.

We say that a 3-party PAKE satisfies the indistinguishability of the session key if the advantage Adv^{ind} is only negligibly larger than $n \cdot q_{send}/|\mathcal{D}|$, where n is a constant and q_{send} is the number of send queries, and parties who complete matching sessions compute the same session key.

3.2.1 Capturing Security Properties.

The condition of KS is represented as the adversary can obtain session keys except one of the test session by SessionKeyReveal query. The condition of KP against passive server is represented as the freshness condition 2, that is, the adversary can obtain static and ephemeral private key of the server by StaticKeyReveal and EphemeralKeyReveal query but no SendClient query for the test session. BI is represented as the freshness condition 3, that is, the adversary can freely eavesdrop messages, obtain ephemeral private key of the server, and send any message to honest clients except the target client by Execute and SendClient queries but no StaticKeyReveal query to the target client and the server. KCI and the condition of FS are also represented as the freshness condition 4, that is, the adversary can obtain static secret of the target client by StaticKeyReveal query but cannot ask StaticKeyReveal query to the partnered client and the server. LEP is represented as the adversary can obtain the ephemeral key of the server during the test session by EphemeralKeyReveal query. Also, our model captures resistance to unknown-key share (UKS) (i.e., any client C including a malicious client insider cannot interfere with the session establishment between two honest clients A and B such that at the end of the attack both parties compute the same session key which C should not be able to learn it. Yet while A is convinced that the key is shared with B, B believes that the peer to the session has been C). UKS is represented as the adversary can establish a malicious insider by EstablishParty query and try to make a honest client which thinks that he shares the session key with the insider share the session key with an another honest client by choosing these two honest clients for the test session.

By the definition of indistinguishability, if adversaries carry out the above malicious behaviors according to KS, KP, BI, UKS, LEP, KCI and FS, information of the session key is not leaked when a scheme satisfies the indistinguishability. Thus, a scheme which satisfies the indistinguishability satisfies KS, KP, resistance to BI, resistance to UKS, resistance to LEP, resistance to KCI and FS.

3.3 Definition of Password Protection

Next, we consider the notion of password protection. This notion provides security properties with respect to passwords, i.e., resistance to UDonDA and to offDA. Beyond the notion of indistinguishability, the notion of password protection is needed because we have to consider security for passwords against attacks by malicious client insiders which can trivially know the session key. Thus, just the notion of indistinguishability cannot capture insider attacks. Also, we cannot allow the adversary to reveal ephemeral private keys of the target client. Given the ephemeral key, the target password is easily derived by offDA because the session key in a session deterministically depends on the client's ephemeral key, the password, and the communication received from the other party.

The adversary is considered successful if it guesses a password of a client. The adversary is allowed to make Execute, SendClient, SendServer, SessionKeyReveal, StaticKeyReveal, EphemeralKeyReveal and TestPassword queries. Let Succ^{pw} denote the event that $\mathsf{TestPassword}(U)$ outputs 1. Note that, we restrict the adversary such that U and the server of the session are honest, and none of the following conditions hold:

- We suppose that S is the corresponding server of U. Then the adversary either makes queries:
  - $\mathsf{StaticKeyReveal}(U)$,

– $\mathsf{StaticKeyReveal}(S)$ or
– $\mathsf{EphemeralKeyReveal}(U^i)$ for any session i.

Now, the adversary $\mathcal{A}$'s advantage is formally defined by:

$$\mathsf{Adv}^{pw}(\mathcal{A}) = \Pr[\mathsf{Succ}^{pw}] \quad \text{and} \quad \mathsf{Adv}^{pw}(t,R) = \max_{\mathcal{A}}\{\mathsf{Adv}^{pw}(\mathcal{A})\},$$

where the maximum is over all $\mathcal{A}$ with time-complexity at most t and using the number of queries to its oracle at most R.

We say that a 3-party PAKE satisfies password protection if the advantage Adv^{pw} is only negligibly larger than $n \cdot q_{send}/|\mathcal{D}|$, where n is a constant and q_{send} is the number of send queries for which messages are found as "invalid" by the party. An "invalid" message means a message which is not derived according to the protocol description.

3.3.1 Capturing Security Properties.

UDonDA is represented as the adversary can unlimitedly use $\mathsf{SendClient}$ and $\mathsf{SendServer}$ queries as far as the party does not find that the query is "invalid". offDA is represented as the adversary can be the insider by $\mathsf{SessionKeyReveal}$, $\mathsf{StaticKeyReveal}$ and $\mathsf{EphemeralKeyReveal}$ queries except the target client and its corresponding server.

By the definition of password protection, if adversaries carry out above malicious behaviors according to UDonDA and offDA, information of the password is not leaked when a scheme satisfies password protection. Thus, a scheme which satisfies password protection satisfies resistance to UDonDA and resistance to offDA.

4. Proposed Scheme

In this section, we show our 3-party PAKE scheme in the same setting as PSAKE.

4.1 Notation

Let p be a prime and let g be a generator of a finite cyclic group G of order p. $A, B \in \mathcal{U}$ are identities of two clients, and S is the identity of their corresponding server. $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a public-key encryption scheme, where $\mathsf{Gen}(1^k)$ is key generation algorithm, $\mathsf{Enc}_{pk}(m;\omega)$ is an encryption algorithm of a message m using a public key pk and the randomness ω. $\mathsf{Dec}_{sk}(c)$ is a decryption algorithm

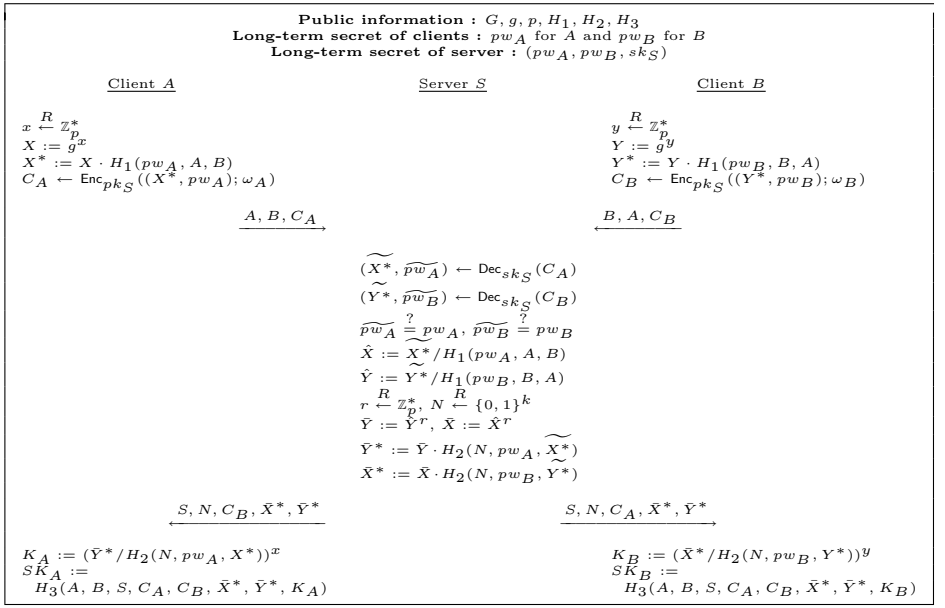

Fig. 1 A high-level overview of our protocol.

of a cipher-text c using a private key sk. A and S (resp. B and S) share a common secret password pw_A (resp. pw_B), and S has pre-established his private key sk_S with his public key pk_S. $H_1 : \mathcal{D} \times \mathcal{U}^2 \to G$, $H_2 : \mathcal{D} \times \{0,1\}^k \times G \to G$ and $H_3 : \mathcal{U}^2 \times \mathcal{S} \times Cspace^2 \times G^3 \to \{0,1\}^k$ are hash functions modeled as random oracles, where $Cspace$ is the space of a cipher-text for $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and k is a sufficiently large security parameter.

For simplicity, we omit "(mod p)" in this paper when computing the modular exponentiation. "$v \xleftarrow{R} V$" means randomly choosing an element v of a set V.

4.2 Protocol Description

Here, we show the construction of our scheme. To guarantee resistance to UDonDA, we apply public-key encryption for servers like PSAKE and the 3-party PAKE scheme in the paper [15]. A high-level overview of our protocol appears in **Fig. 1**.

Then, our protocol is described as follows:

Step 1. Clients A and B choose $x, y \in \mathbb{Z}_p^*$ randomly, compute $X = g^x$ and $Y = g^y$, and blind them as $X^* = X \cdot H_1(pw_A, A, B)$ and $Y^* = Y \cdot H_1(pw_B, B, A)$ respectively. Next, they generate $C_A \leftarrow \mathsf{Enc}_{pk_S}((X^*,\ pw_A\ );\omega_A)$ and $C_B \leftarrow \mathsf{Enc}_{pk_S}((Y^*, pw_B\ );\omega_B)$ by using their corresponding server's public-key pk_S with randomness ω_A and ω_B respectively. Finally, A sends (A, B, C_A) to the server S and B sends (B, A, C_B) to the server S. So, ephemeral private-keys of A and B are (x, X, X^*, ω_A) and (y, Y, Y^*, ω_B) respectively.

Step 2. The server S decrypts $(\widetilde{X}^*, \widetilde{pw_A}) \leftarrow \mathsf{Dec}_{sk_S}(C_A)$ and $(\widetilde{Y}^*, \widetilde{pw_B}) \leftarrow \mathsf{Dec}_{sk_S}(C_B)$ by using sk_S respectively. If $\widetilde{pw_A} \neq pw_A$ or $\widetilde{pw_B} \neq pw_B$, then S aborts the session. It is also crucial that the server rejects any value $\widetilde{X}^*$ or $\widetilde{Y}^*$ whose underlying value X or Y is equal to 1. Otherwise, S computes $\hat{X} = \widetilde{X}^*/H_1(pw_A, A, B)$, blinds it as $\bar{X} := \hat{X}^r$ where r is S's first random value from $\mathbb{Z}_p^*$. S also computes $\hat{Y}$ and $\bar{Y}$ similarly. Next, S computes $\bar{Y}^* = \bar{Y} \cdot H_2(N, pw_A, \widetilde{X}^*)$ where N is S's second random value from $\{0,1\}^k$. S performs similar operations and obtains $\bar{X}^*$. Finally, S sends $(S, N, C_B, \bar{X}^*, \bar{Y}^*)$ to A, sends $(S, N, C_A, \bar{X}^*, \bar{Y}^*)$ to B, and deletes session-specific information ($\widetilde{X}^*$, $\widetilde{Y}^*$, $\widetilde{pw_A}$, $\widetilde{pw_B}$, r, N, $\hat{X}$, $\hat{Y}$, $\bar{X}$, $\bar{Y}$). So, ephemeral private-keys of S are empty.

Step 3. A and B compute their Diffie-Hellman keys $K_A = (\bar{Y}^* \ /\ H_2(N,\ pw_A,\ X^*))^x$ and $K_B = (\bar{X}^* \ /\ H_2(N, pw_B, Y^*))^y$ respectively. Session keys are generated from the Diffie-Hellman key and transcripts, $SK_A = H_3(A,\ B,\ S,\ C_A,\ C_B,\ \bar{X}^*,\ \bar{Y}^*,\ K_A)$ and $SK_B = H_3(A,\ B,\ S,\ C_A,\ C_B,\ \bar{X}^*,\ \bar{Y}^*,\ K_B)$. When session keys are honestly generated, $SK_A = SK_B$ because $K_A = (g^{yr})^x$ and $K_B = (g^{xr})^y$.

4.3 Design Principles

Our protocol can be viewed as an extension of Abdalla-Pointcheval scheme [2]. The main difference consists in the use of public-key encryption.

First, upon receiving an input from a client, the corresponding server verifies the validity of the encrypted password of the client and its identity. This procedure prevents UDonDA as in the technique of Lin, et al. [15]. Applying the server's public-key may put a burden on clients because they have to verify the server's public-key in advance, and the certificate infrastructure is needed. However, we can easily resolve this problem by applying ID-based encryption for the server instead of standard public-key encryption. Since clients can encrypt messages only by matching the server's ID in an ID-based encryption, they neither need to keep nor verify the server's public-key. If we replace the use of public-key encryption by the use of ID-based encryption, the security of our scheme is not changed. Note that when using the ID-based encryption, the Key Generation Center can know all the client's passwords because it knows the server's private key.

Next, the elimination of ephemeral states except the necessary states is needed for resistance to LEP as in the technique of LaMacchia, et al. [14]. Even if EphemeralKeyReveal query is asked, information of passwords and the session key is not leaked because all the critical states are deleted immediately after being used.

Finally, when a client blinds X or Y with his password, we make the client include the identities of both clients into the computation of the password-based blinding factors. This procedure prevents KCI and UKS by a malicious client insider as in the technique of Choo, et al. [8].

We show the comparison between previous schemes [1),2),9),19)] and our scheme in **Table 2**. The computational cost is measured by the time of exponentiations because the time for hashes, MACs and symmetric key encryptions is generally ignored. The communication cost is measured by the number of elements of groups in the transcript. We suppose that the scheme of Abdalla, et al. [1] and the scheme of Wang and Hu [19] use the KOY protocol [13] as the 2-party PAKE, and our scheme uses the same encryption scheme as the scheme of Cliff, et al. [9].

Our scheme is more efficient than previous schemes [1),9),19)] in both computational and communication costs. However, our scheme is less efficient than the scheme of Abdalla and Pointcheval [2]. That scheme [2] does not have any encryption because it cannot guarantee LEP. To guarantee LEP, our scheme needs an additional encryption besides the elements which the scheme [2] needs.

There is an alternative construction which is more efficient than the proposed construction in the computational cost. That is, we remove $H_1(pw_A, A, B)$ and add A, B as inputs to H_2. By this simple modification, the time of 1 hash evaluation for the client and the time of 2 hash evaluations for the server can be

Table 2 Comparison of the efficiency between previous schemes and our scheme.

	Client computation cost		Server computation cost		Communication cost
[AFP05]					
...KOY	1 signature verify	5	2 signature verify	10	$20\|G\| + 2\|VerKey\|$ $+2\|Signature\|$
	2 exp.	2	6 exp.	6	
	2 double exp.	2.6	2 double exp.	2.6	
	2 multi exp.	5.3	4 multi exp.	10.6	
...key dist.	1 symm. enc.		2 symm. enc.		$2k$ bit
...other	2 exp.	2			$4\|G\| + 2k$ bit
	2 MAC evaluation				
	Total	16.9	Total	29.2	$24\|G\| + 2\|VerKey\|$ $+2\|Signature\| + 4k$ bit
[AP05]	2 exp.	2	2 exp.	2	
	2 hash evaluation		4 hash evaluation		
	Total	2	Total	2	$4\|G\|$
[WH06]					
...KOY		14.9		29.2	$20\|G\| + 2\|VerKey\|$ $+2\|Signature\|$
...other	2 exp.	2			$8\|G\| + 4k$ bit
	2 MAC evaluation				
	Total	16.9	Total	29.2	$28\|G\| + 2\|VerKey\|$ $+2\|Signature\| + 4k$ bit
[CTB06]	2 exp.	2			
	2 PK enc.	8	4 PK dec.	12	
	Total	10	Total	12	$14\|G\| + 4\|CipherText\| + 5k$ bit
Our scheme	2 exp.	2	2 exp.	2	
	1 PK enc.	4	2 PK dec.	6	
	3 hash evaluation		4 hash evaluation		
	Total	6	Total	8	$4\|G\| + 4\|CipherText\| + 2k$ bit

reduced.

5. Security of Our Scheme

In this section, we show the security properties of our scheme.

5.1 Building Blocks

We recall the definition of the decisional Diffie-Hellman assumptions which we use in the security proof of our scheme. Let p be a prime and let g be a generator of a finite cyclic group G of order p.

5.1.1 Decisional Diffie-Hellman Assumption (DDH)

We can define the DDH assumption by defining two experiments, $\mathsf{Exp}_{g,p}^{ddh-real}(\mathcal{I})$ and $\mathsf{Exp}_{g,p}^{ddh-rand}(\mathcal{I})$. For a solver $\mathcal{I}$, inputs (g^u, g^v, Z) are provided, where u, v are drawn at random from $\mathbb{Z}_p^*$. $Z = g^{uv}$ in $\mathsf{Exp}_{g,p}^{ddh-real}(\mathcal{I})$ and $Z = g^w$ in $Exp_{g,p}^{ddh-rand}(\mathcal{I})$, where w is drawn at random from $\mathbb{Z}_p^*$. We define the advantage of $\mathcal{I}$ in violating the DDH assumption, $\mathsf{Adv}_{g,p}^{ddh}(\mathcal{I})$, as $|\Pr[\mathsf{Exp}_{g,p}^{ddh-real}(\mathcal{I}) = 1] - \Pr[\mathsf{Exp}_{g,p}^{ddh-rand}(\mathcal{I}) = 1]|$. The advantage function of the group, $\mathsf{Adv}_{g,p}^{ddh}(t)$, is defined as the maximum value of $\mathsf{Adv}_{g,p}^{ddh}(\mathcal{I})$ over all $\mathcal{I}$ with a time-complexity at most t.

5.2 Main Theorems

Theorem 5.1 Assuming $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically secure public-key encryption scheme and the DDH problem is hard, our scheme satisfies the indistinguishability in Section 3.2.

[Proof]

We define a sequence of hybrid experiments, starting with the real attack and ending in an experiment in which the adversary has no advantage. Each experiment addresses a different security aspect. For each experiment Exp_n, we define an event Succ_n as Succ^{ind} in Exp_n. The proof follows the one of Abdalla et al.'s 3-party PAKE [2].

5.2.1 Experiment Exp_0.

This experiment corresponds to the real execution, in the random oracle model. By definition, we have

$$\mathsf{Adv}^{ind}(\mathcal{A}) = |2 \cdot \Pr[\mathsf{Succ}_0] - 1|. \quad (1)$$

5.2.2 Experiment Exp_1.

In this experiment, we simulate the random oracles H_1, H_2 and H_3 as usual by maintaining hash lists Λ_{H_1}, Λ_{H_2}, and Λ_{H_3} as follows:

- On hash query $H_1(Q)$ (resp. $H_2(Q)$) for which there exists a record (Q, R) in the list Λ_{H_1} (resp. Λ_{H_2}), return R. Otherwise, choose an element $R \in G$, add the record (Q, R) to the list Λ_{H_1} (resp. Λ_{H_2}), and return R.
- On hash query $H_3(Q)$ for which there exists a record (Q, R) in the list Λ_{H_3}, return R. Otherwise, choose an element $R \in \{0,1\}^k$, add the record (Q, R) to the list Λ_{H_3}, and return R.

The Execute, SessionKeyReveal, SendClient, SendServer, StaticKeyReveal, EphemeralKeyReveal, EstablishParty and Test oracles are also simulated as in the real attack as follows:

- SendClient **query**
  - On a query $\mathsf{SendClient}(U_1^i, (U_2, \mathbf{start}))$, assuming U_1^i is in the correct state,

we proceed as follows:

$\theta \stackrel{R}{\leftarrow} \mathbb{Z}_p; \Theta \leftarrow g^\theta$

$\Theta^* \leftarrow \Theta \cdot H_1(pw_{U_1}, U_1, U_2)$

$\gamma_{U_1} \leftarrow \mathsf{Enc}_{pk_S}((\Theta^*, pw_{U_1}); \omega_{U_1})$

return (U_1, U_2, γ_{U_1})

- On a query $\mathsf{SendClient}(U_1^i, (S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*))$, assuming U_1^i is in the correct state, S is the server and U_2 is the intended partner, we proceed as follows:

  $K_{U_1} \leftarrow (\bar{\Phi}^*/H_2(N, pw_{U_1}, \Theta^*))^\theta$

  $SK_{U_1} \leftarrow H_3(U_1, U_2, S, \gamma_{U_1}, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*, K_{U_1})$

- SendServer **query**
  - On query $\mathsf{SendServer}(S^i, ((U_1, U_2, \gamma_{U_1}), (U_2, U_1, \gamma_{U_2})))$, we proceed as follows:

    $(\widetilde{\Theta^*}, \widetilde{pw_{U_1}}) \leftarrow \mathsf{Dec}_{sk_S}(\gamma_{U_1})$; $(\widetilde{\Phi^*}, \widetilde{pw_{U_2}}) \leftarrow \mathsf{Dec}_{sk_S}(\gamma_{U_2})$

    $\widetilde{pw_{U_1}} \stackrel{?}{=} pw_{U_1}$; $\widetilde{pw_{U_2}} \stackrel{?}{=} pw_{U_2}$

    $\hat{\Theta} := \widetilde{\Theta^*}/H_1(pw_{U_1}, U_1, U_2)$; $\hat{\Phi} := \widetilde{\Phi^*}/H_1(pw_{U_2}, U_2, U_1)$

    $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$

    $N \stackrel{R}{\leftarrow} \{0,1\}^k$

    $\bar{\Phi} \leftarrow \hat{\Phi}^r$; $\bar{\Theta} \leftarrow \hat{\Theta}^r$

    $\bar{\Phi}^* \leftarrow \bar{\Phi} \cdot H_2(N, pw_{U_1}, \widetilde{\Theta^*})$; $\bar{\Theta}^* \leftarrow \bar{\Theta} \cdot H_2(N, pw_{U_2}, \widetilde{\Phi^*})$

    return $((S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*), (S, N, \gamma_{U_1}, \bar{\Theta}^*, \bar{\Phi}^*))$

- SessionKeyReveal **query**
  - On query $\mathsf{SessionKeyReveal}(U^i)$, we proceed as follows:

    if session key SK is defined for instance U^i

    then return SK,

    else return $\perp$.

- Execute **query**
  - On query $\mathsf{Execute}(U_1^{i_1}, U_2^{i_2}, S^{i_3})$, we proceed as follows:

    $(U_1, U_2, \gamma_{U_1}) \leftarrow \mathsf{SendClient}(U_1^{i_1}, (U_2, \mathbf{start}))$

    $(U_2, U_1, \gamma_{U_2}) \leftarrow \mathsf{SendClient}(U_2^{i_2}, (U_1, \mathbf{start}))$

    $((S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*), (S, N, \gamma_{U_1}, \bar{\Theta}^*, \bar{\Phi}^*))$

    $\leftarrow \mathsf{SendServer}(S^{i_3}, ((U_1, U_2, \gamma_{U_1}), (U_2, U_1, \gamma_{U_2})))$

    $\mathsf{SendClient}(U_1^{i_1}, (S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*))$

    $\mathsf{SendClient}(U_2^{i_2}, (S, N, \gamma_{U_1}, \bar{\Theta}^*, \bar{\Phi}^*))$

    return $((U_1, U_2, \gamma_{U_1}), (U_2, U_1, \gamma_{U_2}), (S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*), (S, N, \gamma_{U_1}, \bar{\Theta}^*, \bar{\Phi}^*))$

- EstablishParty **query**
  - On query $\mathsf{EstablishParty}(U^i, pw_U)$, we proceed as follows:

    if there is U^i **then** do nothing

    else establish instance U^i with the static secret pw_U.

- StaticKeyReveal **query**
  - On query $\mathsf{StaticKeyReveal}(P)$, we proceed as follows:

    if P is a client **then return** pw_P,

    else P is a server **then return** the static secret of the corresponding client of P.

- EphemeralKeyReveal **query**
  - On query $\mathsf{EphemeralKeyReveal}(P^i)$, we proceed as follows:

    if the ephemeral key of P^i is already generated **then return** it.

    (i.e., output $(\theta, \Theta, \Theta^*, \omega_{U_1})$ (resp. $(\phi, \Phi, \Phi^*, \omega_{U_2})$) if $P = U_1$

    (resp. $P = U_2$), otherwise output an empty symbol)

- Test **query**
  - On query $\mathsf{Test}(U^i)$, we proceed as follows:

    $SK \leftarrow \mathsf{SessionKeyReveal}(U^i)$

    if $SK = \perp$ **then return** $\perp$

    else

    $b \stackrel{R}{\leftarrow} \{0,1\}$

    if $b = 1$ **then** $SK' \leftarrow SK$ **else** $SK' \stackrel{R}{\leftarrow} \{0,1\}^k$

    return SK'

The term "$\stackrel{?}{=}$" means check of equation. One can easily see that this experiment is perfectly indistinguishable from the real experiment. Hence,

$$\Pr[\mathsf{Succ}_1] = \Pr[\mathsf{Succ}_0]. \quad (2)$$

5.2.3 Experiment Exp_2.

In this experiment, we simulate all oracles as in Experiment Exp_1, except that we halt all executions in which a collision occurs in the output of the H_1 and H_2 oracles or in the transcript $((U_1, U_2, \gamma_{U_1}), (U_2, U_1, \gamma_{U_2}), (S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*), (S, N, \gamma_{U_1}, \bar{\Theta}^*, \bar{\Phi}^*))$. According to the birthday paradox, the probability of collisions

in the output of the H_i oracle is at most $q_{H_i}^2/(2p)$ for $i = 1, 2$ where q_{H_i} denotes the maximum number of queries to H_i. Similarly, the probability of collisions in the transcripts is at most $(q_{\text{start}} + q_{\text{exe}})^2/(2p)$ where q_{start} represents the number of queries to the SendClient oracle used to initiate a client oracle instance and q_{exe} represents the number of queries to the Execute oracle, because either γ_{U_1} or γ_{U_2} was simulated and thus chosen uniformly at random. Consequently,

$$|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_1]| \leq \frac{q_{H_1}^2 + q_{H_2}^2 + (q_{\text{start}} + q_{\text{exe}})^2}{2p}. \quad (3)$$

5.2.4 Experiment Exp_3.

In this experiment, we change the simulation of queries to the SendClient oracles. This time, we select at random a matching session for (U_1^i, U_2^j) executed by some honest parties U_1 and U_2. When $\mathsf{SendClient}(U_1^i; (U_2; \textbf{start}))$ and $\mathsf{SendClient}(U_2^j; (U_1; \textbf{start}))$ are asked, we compute $\Theta = g^\theta$ and $\Phi = g^\phi$ normally but set $\Theta = g^u$ and $\Phi = g^v$ where u, v are drawn at random from $\mathbb{Z}_p^*$. Also, when $\mathsf{SendClient}(U_1^i, (S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*))$ and $\mathsf{SendClient}(U_2^j, (S, N, \gamma_{U_1}, \bar{\Theta}^*, \bar{\Phi}^*))$ are asked, we compute $K_{U_1} = (\bar{\Phi}^*/H_2(N, pw_{U_1}, \Theta^*))^\theta$ and $K_{U_2} = (\bar{\Theta}^*/H_2(N, pw_{U_2}, \Phi^*))^\phi$ normally but set $K_{U_1} = K_{U_2} = (g^{uv})^r$. Since the selected session is matching, $(\bar{\Phi}^*/H_2(N, pw_{U_1}, \Theta^*))^\theta$ and $(\bar{\Theta}^*/H_2(N, pw_{U_2}, \Phi^*))^\phi$ have to be equivalent to $(g^{uv})^r$ by the definition of matching session in Section 2.1.

So, one can easily see that this experiment is perfectly indistinguishable from the real experiment. Hence,

$$\Pr[\mathsf{Succ}_3] = \Pr[\mathsf{Succ}_2]. \quad (4)$$

5.2.5 Experiment Exp_4.

In this experiment, we once again change the simulation of queries to the SendClient oracle for the selected session. This time, we change the way we compute the values K_{U_1} and K_{U_2} so that K_{U_1} and K_{U_2} become independent with passwords and ephemeral keys. When $\mathsf{SendClient}(U_1^i, (S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*))$ and $\mathsf{SendClient}(U_2^j, (S, N, \gamma_{U_1}, \bar{\Theta}^*, \bar{\Phi}^*))$ are asked, we compute $K_{U_1} = K_{U_2} = (g^{uv})^r$ in Exp_3 but set $K_{U_1} = K_{U_2} = (g^w)^r$, where w is drawn at random from $\mathbb{Z}_p^*$.

As the following lemma shows, the difference between the current experiment and the previous one is negligible as long as the DDH assumption holds.

Lemma 5.1 $|\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_3]| \leq q_{\text{exe}} \cdot \mathsf{Adv}_{g,p}^{ddh}(t)$.

[Proof]

By assuming a successful distinguisher (an adversary) against Exp_3 and Exp_4, we construct a DDH solver. The only difference between Exp_3 and Exp_4 is in the way to generate K_{U_1} and K_{U_2} for a session.

First, the DDH solver obtains a DDH tuple (g, g^u, g^v, Z). As Exp_3 and Exp_4, the DDH solver selects a matching session for (U_1^i, U_2^j) executed by some honest parties U_1 and U_2. Then, when $\mathsf{SendClient}(U_1^i; (U_2; \textbf{start}))$ and $\mathsf{SendClient}(U_2^j; (U_1; \textbf{start}))$ are asked, the DDH solver sets $\Theta = g^u$ and $\Phi = g^v$. Also, when $\mathsf{SendClient}(U_1^i, (S, N, \gamma_{U_2}, \bar{\Theta}^*, \bar{\Phi}^*))$ and $\mathsf{SendClient}\ (U_2^j, (S, N, \gamma_{U_1}, \bar{\Theta}^*, \bar{\Phi}^*))$ are asked, the DDH solver sets $K_{U_1} = K_{U_2} = Z^r$. For all other queries, the DDH solver returns as Exp_3 and Exp_4.

With probability $1/q_{\text{exe}}$ the distinguisher picks the selected session as the test session, i.e., the distinguisher asks $\mathsf{Test}(U_1^i)$ or $\mathsf{Test}(U_2^j)$. By Definition 3.2, the distinguisher is allowed to reveal static keys (pw_{U_1}, pw_{U_2}, sk_S) and ephemeral keys of the server (empty) but it is not allowed to reveal ephemeral keys of clients $((u, \Theta, \Theta^*, \omega_{U_1}), (v, \Phi, \Phi^*, \omega_{U_2}))$. So, the DDH solver simulates all oracle queries without knowing u and v. From the obtained information, the distinguisher can compute ($\Theta = g^u$, $\Phi = g^v$) but cannot compute (K_{U_1}, K_{U_2}). In the case of $Z = g^{uv}$, this environment for the distinguisher is equivalent to Exp_3. In the case of $Z = g^w$, this environment for the distinguisher is equivalent to Exp_4.

Finally, if the distinguisher decides that he interacted with Exp_3, then the DDH solver outputs 1. And, if the distinguisher decides that he interacted with Exp_4, then the DDH solver outputs 0. Since the distinguisher can distinguish with a non-negligible probability, $\mathsf{Adv}_{g,p}^{ddh}(t)$ is also non-negligible.

□

In Exp_4, Diffie-Hellman keys K_{U_1} and K_{U_2} are random and independent with passwords and ephemeral keys. So, there are three possible cases where the adversary distinguishes the real session key and the random key as follows:

Case 1. the adversary queries (U_1, U_2, S, γ_{U_1}, γ_{U_2}, $\bar{\Theta}^*$, $\bar{\Phi}^*$, g^w) to H_3.

Case 2. the adversary asks SendClient query except $\mathsf{SendClient}(U_2^j, m)$ query, and successfully impersonates U_1 to U_2.

Case 3. the adversary asks SendClient query except $\mathsf{SendClient}(U_1^i, m)$ query, and successfully impersonates U_2 to U_1.

(Case 1) The probability that this event occurs is q_{H_3}/p.

(Case 2) By Definition 3.2, the distinguisher is allowed to reveal the static key pw_{U_2} of U_2 and ephemeral keys of the server (empty) but it is not allowed to reveal static keys $(pw_{U_1}, pw_{U_2}, sk_S)$ of S (and U_1), and ephemeral keys of clients $((u, \Theta, \Theta^*, \omega_{U_1}), (v, \Phi, \Phi^*, \omega_{U_2}))$. Thus, in order to impersonate U_1, i.e., the adversary has to compute a valid ciphertext to S_1, the adversary has to obtain some information of the password pw_{U_1} of U_1 from the ciphertext of U_1. However, by semantic security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ the adversary can obtain no information with respect to pw_{U_1} from the ciphertext of U_1. Therefore, the probability that this event occurs is lower than $1/|\mathcal{D}| + \mathsf{Adv}^{SS}(t')$, where the advantage function of the attack to the semantic security, $\mathsf{Adv}^{SS}(t)$, is defined with a time-complexity at most t'.

(Case 3) By similar reason, the probability that this event occurs is lower than $1/|\mathcal{D}| + \mathsf{Adv}^{SS}(t')$.

Hence,

$$\Pr[\mathsf{Succ}_4] = 1/2 + max\{q_{H_3}/p, 1/|\mathcal{D}| + \mathsf{Adv}^{SS}(t')\}. \quad (5)$$

From (1), (2), (3), (4), (5) and Lemma 5.1, Theorem 5.1 is proven.

□

Theorem 5.2 Assuming $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically secure public-key encryption scheme, our scheme satisfies the password protection in Section 3.3.

[Proof]

For each experiment Exp_n, we define an event Succ_n as Succ^{pw} in Exp_n.

Experiment Exp_0. This experiment corresponds to the real execution, in the random oracle model. By the definition, we have

$$\mathsf{Adv}^{pw}(\mathcal{A}) = \Pr[\mathsf{Succ}_0]. \quad (6)$$

Experiment Exp_1. In this experiment, we simulate the random oracles H_1, H_2 and H_3 as usual by maintaining hash lists Λ_{H_1}, Λ_{H_2}, and Λ_{H_3} as the proof of Theorem 5.1. Also, the $\mathsf{Execute}$, $\mathsf{SessionKeyReveal}$, $\mathsf{SendClient}$, $\mathsf{SendServer}$, $\mathsf{StaticKeyReveal}$, $\mathsf{EphemeralKeyReveal}$ and $\mathsf{EstablishParty}$ oracles are simulated as in the real attack as the proof of Theorem 5.1. So, we describe only the simulation of $\mathsf{TestPassword}$ oracle as follows:

◇ $\mathsf{TestPassword}$ **query**

- On query $\mathsf{TestPassword}(U, pw')$, we proceed as follows:

  $pw \leftarrow \mathsf{StaticKeyReveal}(U)$

  if $pw' = pw$ **then return** 1

  else return 0.

One can easily see that this experiment is perfectly indistinguishable from the real experiment. Hence,

$$\Pr[\mathsf{Succ}_1] = \Pr[\mathsf{Succ}_0]. \quad (7)$$

Experiment Exp_2. In this experiment, we change the simulation of queries to the $\mathsf{SendClient}$, $\mathsf{SendServer}$, $\mathsf{StaticKeyReveal}$, $\mathsf{EphemeralKeyReveal}$ and $\mathsf{TestPassword}$ oracles. This time, we select at random a honest client A which has the corresponding server S. When $\mathsf{SendClient}(A^i; (U_2; \mathbf{start}))$ is asked, we choose a random value $z \leftarrow \mathbb{Z}_p^*$ and compute $\gamma_A \leftarrow \mathsf{Enc}_{pk_S}(g^z; \omega_A)$. And, when $\mathsf{StaticKeyReveal}(A)$, $\mathsf{StaticKeyReveal}(S)$, $\mathsf{EphemeralKeyReveal}(A^i)$ or $\mathsf{EphemeralKeyReveal}(S^i)$ for any i is asked, we finish the simulation as failed. Moreover, when $\mathsf{TestPassword}(U, pw')$ where U is not A is asked, we finish the simulation as failed.

Also, when $\mathsf{SendServer}(S^{i_1}, ((A, U_2, \gamma'_A), (U_2, A, \gamma'_{U_2})))$ is asked, we verify $\gamma'_A \stackrel{?}{=} \gamma_A$ instead of verifying $\widetilde{pw_A} \stackrel{?}{=} pw_A$, and let $\hat{\Theta} := \widetilde{\Theta^*}$.

As the following lemma shows, the difference between the current experiment and the previous one is negligible as long as semantic security for $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ holds where the advantage function of the attack to the semantic security, $\mathsf{Adv}^{SS}(t)$, is defined with a time-complexity at most t'.

Lemma 5.2 $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_1]| \leq |\mathcal{U}| \cdot \mathsf{Adv}^{SS}(t')$.

[Proof]

By assuming a successful distinguisher (an adversary) against Exp_2 and Exp_1, we construct a semantic security (SS) attacker. The only difference between Exp_2 and Exp_1 is in the way to generate γ_A for a client A.

First, as Exp_2, the SS attacker selects a honest client A which has the corresponding server S. Then, when $\mathsf{SendClient}(A; (U_2; \mathbf{start}))$ is asked, the SS attacker sends messages (m_0, m_1) to the challenge oracle for the SS game where m_0 is computed as Exp_1 and m_1 is computed as Exp_2. Upon receiving $\gamma^* \leftarrow \mathsf{Enc}_{pk_S}(m_b)$ where $b = 0$ or $b = 1$ from the challenge oracle, the SS attacker sets $\gamma_A = \gamma^*$. For all other queries, the SS attacker returns as Exp_1 and Exp_2.

With probability $1/|\mathcal{U}|$ the distinguisher picks the selected client as the client to guess the password, i.e., the distinguisher asks $\mathsf{TestPassword}(A)$. By Definition 3.3, the distinguisher is allowed to reveal static keys except A's and S's (pw_{U_2}) and ephemeral keys except A's and S's $(\phi, \Phi, \Phi^*, \omega_{U_2})$ but it is not allowed to reveal static and ephemeral keys of A's and S's. Thus, the SS attacker simulates all oracle queries without knowing sk_S. In the case of $b = 0$, this environment for the distinguisher is equivalent to Exp_1. In the case of $b = 1$, this environment for the distinguisher is equivalent to Exp_2.

Finally, if the distinguisher decides that he interacted with Exp_1, then the SS attacker outputs 0. And, if the distinguisher decides that he interacted with Exp_2, then the SS attacker outputs 1. Since the distinguisher can distinguish with a non-negligible probability, $\mathsf{Adv}^{SS}(t')$ is also non-negligible.

□

Experiment Exp_3. In this experiment, we once again change the simulation of queries to the $\mathsf{SendClient}$ and $\mathsf{SendServer}$ oracle for the selected client and server. This time, we change the way we compute the values $\bar{\Phi}^*$ and K_A so that $\bar{\Phi}^*$ is independent from A's password. When $\mathsf{SendServer}(S^{i_1}, ((A, U_2, \gamma'_A), (U_2, A, \gamma'_{U_2})))$ is asked, we compute $\bar{\Phi}^* = \bar{\Phi} \cdot H_2(N, \widetilde{\Theta^*})$ instead of $\bar{\Phi}^* = \bar{\Phi} \cdot H_2(N, pw_A, \widetilde{\Theta^*})$. And, when $\mathsf{SendClient}(A, (S, N, \gamma'_A, \bar{\Theta}^*, \bar{\Phi}^*))$ is asked, we compute $K_A \leftarrow (\bar{\Phi}^*/H_2(N, \Theta^*))^z$.

As the following lemma shows, this experiment is perfectly indistinguishable from the previous one.

Lemma 5.3 $\Pr[\mathsf{Succ}_3] = \Pr[\mathsf{Succ}_2]$.

[Proof]

The only difference between Exp_3 and Exp_2 is in the way to generate $\bar{\Phi}^*$ for a client A.

By Definition 3.3, the distinguisher is allowed to reveal static keys except A's and S's (pw_{U_2}) and ephemeral keys except A's $(\theta, \Theta, \Theta^*, \omega_A)$ but it is not allowed to reveal the static and ephemeral key of A's. So, $\widetilde{\Theta^*}$ is an unknown value for the distinguisher. Therefore, $\bar{\Phi} \cdot H_2(N, \widetilde{\Theta^*})$ and $\bar{\Phi} \cdot H_2(N, pw_A, \widetilde{\Theta^*})$ are perfectly indistinguishable for the distinguisher.

Also, though the way to compute K_A is changed, the final session key is not changed. So, Exp_3 and Exp_2 are indistinguishable.

□

In Exp_3, all transcripts are random and independent with A's password. So, the only way that the adversary guesses the password is by random guessing. If the adversary fails, the message is declared as "invalid". Thus, the adversary can try to guess only q_{send} times. The probability that the adversary succeeds is $q_{send}/|\mathcal{D}|$. Hence,

$$\Pr[\mathsf{Succ}_3] = q_{send}/|\mathcal{D}|. \quad (8)$$

From (6), (7), (8), Lemma 5.2 and Lemma 5.3, Theorem 5.2 is proven.

□

6. Conclusion

Firstly, we pointed out that previous security definitions of the 3-party PAKE cannot capture all desirable security requirements. Next, we proposed a new stronger definition of the 3-party PAKE which captures all desirable security requirements. Finally, we introduced a 3-party PAKE protocol in the same setting as PSAKE with optimal rounds for the client and proved its security in the sense of our stronger definition.

Our scheme use the public-key encryption as a building block in order to guarantee the resistance to UDonDA. However, public-key encryption schemes are time-consuming. Thus, a remaining problem of further researches is an efficient construction which satisfies stronger security requirements.

References

1) Abdalla, M., Fouque, P.-A. and Pointcheval, D.: Password-Based Authenticated Key Exchange in the Three-Party Setting, *Public Key Cryptography 2005*, pp.65–84 (2005).
2) Abdalla, M. and Pointcheval, D.: Interactive Diffie-Hellman Assumptions with Applications to Password-Based Authentication, *Financial Cryptography 2005*, pp.341–356 (2005).
3) Bellare, M., Pointcheval, D. and Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks, *EUROCRYPT 2000*, pp.139–155 (2000).
4) Bellovin, S.M. and Merritt, M.: Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks, *IEEE S&P 1992*, pp.72–84 (1992).
5) Boyko, V., MacKenzie, P.D. and Patel, S.: Provably Secure Password-

Authenticated Key Exchange Using Diffie-Hellman, *EUROCRYPT 2000*, pp.156–171 (2000).
6) Canetti, R. and Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels, *EUROCRYPT 2001*, pp.453–474 (2001).
7) Chang, Y.-F. and Chang, C.-C.: Password-authenticated 3PEKE with Round Efficiency without Server's Public Key, *CW 2005*, pp.340–344 (2005).
8) Choo, K.-K.R., Boyd, C. and Hitchcock, Y.: Examining Indistinguishability-Based Proof Models for Key Establishment Protocols, *ASIACRYPT 2005*, pp.585–604 (2005).
9) Cliff, Y., Tin, Y.S.T. and Boyd, C.: Password Based Server Aided Key Exchange, *ACNS 2006*, pp.146–161 (2006).
10) Diffie, W., van Oorschot, P.C. and Wiener, M.J.: Authentication and Authenticated Key Exchanges, *Des. Codes Cryptography*, Vol.2, No.2, pp.107–125 (1992).
11) Ding, Y. and Horster, P.: Undetectable On-line Password Guessing Attacks, *Operating Systems Review*, Vol.29, No.4, pp.77–86 (1995).
12) Jablon, D.P.: Strong Password-Only Authenticated Key Exchange, *Computer Communication Review, ACM SIGCOMM*, Vol.26, No.5, pp.5–26 (1996).
13) Katz, J., Ostrovsky, R. and Yung, M.: Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords, *EUROCRYPT 2001*, pp.475–494 (2001).
14) LaMacchia, B., Lauter, K. and Mityagin, A.: Stronger Security of Authenticated Key Exchange, *Provsec 2007* (2007).
15) Lin, C.-L., Sun, H.-M. and Hwang, T.: Three-party Encrypted Key Exchange: Attacks and A Solution, *ACM Operating Systems Review*, Vol.34, No.4, pp.12–20 (2000).
16) Lomas, T.M.A., Gong, L., Saltzer, J.H. and Needham, R.M.: Reducing Risks from Poorly Chosen Keys, *SOSP 1989*, pp.14–18 (1989).
17) Steiner, J.G., Neuman, B.C. and Schiller, J.I.: Kerberos: An Authentication Service for Open Network Systems, *USENIX Winter 1988*, pp.191–202 (1988).
18) Steiner, M., Tsudik, G. and Waidner, M.: Refinement and Extension of Encrypted Key Exchange, *ACM Operating Systems Review*, Vol.29, No.3, pp.22–30 (1995).
19) Wang, W. and Hu, L.: Efficient and Provably Secure Generic Construction of Three-Party Password-Based Authenticated Key Exchange Protocols, *INDOCRYPT 2006*, pp.118–132 (2006).

(Received December 1, 2008)
(Accepted June 4, 2009)
(Original version of this article can be found in the Journal of Information Processing Vol.17, pp.202–215.)

Kazuki Yoneyama was born in 1981. He received the B.E., M.E. and Ph.D. degrees from University of Electro-Communications, Tokyo, Japan, in 2004, 2006 and 2008, respectively. He has been a postdoctoral research fellow at University of Electro-Communications since 2008. He is presently engaged in research on cryptography. He is a member of the International Association for Cryptologic Research, IEICE and JSIAM.