Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 825072, 11 pages
http://dx.doi.org/10.1155/2014/825072

Research Article

Password-Only Authenticated Three-Party Key Exchange with Provable Security in the Standard Model

Junghyun Nam,[1] Kim-Kwang Raymond Choo,[2] Junghwan Kim,[1] Hyun-Kyu Kang,[1] Jinsoo Kim,[1] Juryon Paik,[3] and Dongho Won[3]

[1] *Department of Computer Engineering, Konkuk University, 268 Chungwondaero, Chungju, Chungcheongbukdo 380-701, Republic of Korea*
[2] *Information Assurance Research Group, Advanced Computing Research Centre, University of South Australia, Mawson Lakes, SA 5095, Australia*
[3] *Department of Computer Engineering, Sungkyunkwan University, 2066 Seoburo, Suwon, Gyeonggido 440-746, Republic of Korea*

Correspondence should be addressed to Dongho Won; dhwon@security.re.kr

Received 23 January 2014; Accepted 27 February 2014; Published 14 April 2014

Academic Editors: T. Cao, M. Ivanovic, and F. Yu

Protocols for password-only authenticated key exchange (PAKE) in the three-party setting allow two clients registered with the same authentication server to derive a common secret key from their individual password shared with the server. Existing three-party PAKE protocols were proven secure under the assumption of the existence of random oracles or in a model that does not consider insider attacks. Therefore, these protocols may turn out to be insecure when the random oracle is instantiated with a particular hash function or an insider attack is mounted against the partner client. The contribution of this paper is to present the first three-party PAKE protocol whose security is proven without any idealized assumptions in a model that captures insider attacks. The proof model we use is a variant of the indistinguishability-based model of Bellare, Pointcheval, and Rogaway (2000), which is one of the most widely accepted models for security analysis of password-based key exchange protocols. We demonstrated that our protocol achieves not only the typical indistinguishability-based security of session keys but also the password security against undetectable online dictionary attacks.

1. Introduction

Authenticated key exchange is one of the most fundamental problems in cryptography and network security. In 1992, Bellovin and Merritt [1] introduced encrypted key exchange (or EKE) protocols, which allow

> *arbitrary two parties, who share only a low-entropy password, to establish a common high-entropy secret key (called a session key) over an insecure public network.*

Since the work of Bellovin and Merritt [1], password-only authenticated key exchange (PAKE) protocols have attracted much greater attention mainly due to the persistent popularity of passwords as a practical (and cheap) authentication method [2]. Since the publication of the first EKE protocol (with only heuristic security arguments), many provably secure PAKE protocols have been published. Recent examples include the protocol of Katz and Vaikuntanathan [3], which enjoys both round optimality and provable security in the standard model (i.e., without random oracles and ideal ciphers).

A major challenge in designing PAKE protocols is to protect passwords from a *dictionary attack*, in which an adversary enumerates all possible passwords while testing each one against known password verifiers in order to determine the correct one. The design of two-party PAKE protocols secure against dictionary attacks has been extensively studied over the past two decades and is now fairly well understood. However, three-party PAKE protocols have received far less attention and preventing dictionary attacks is more challenging in the three-party setting. Unlike the two-party setting that assumes the same password is shared between the two parties, the three-party setting assumes that the two parties (commonly known as *clients*) wishing to establish a session key do not share the same password but hold their individual password shared only with a trusted server. This implies that in the three-party setting, a malicious client can attempt to mount an insider dictionary attack against its partner client. Indeed, many published three-party PAKE protocols were subsequently found to be vulnerable to an insider online/offline dictionary attack (e.g., [4–10]).

It is widely regarded that the design of key exchange protocols (including PAKE protocols) is notoriously hard, and conducting security analysis for such protocols is time-consuming and error-prone [11–13]. The many flaws discovered in published protocols have promoted the use of formal models and rigorous security proofs [14–16]. In the provable security paradigm for protocol analysis, a deductive reasoning process is adopted whereby emphasis is placed on a proven reduction from the problem of breaking the protocol to another problem believed to be computationally hard. A complete mathematical proof with respect to cryptographic definitions provides a strong assurance that a protocol is behaving as desired. It is now standard practice for protocol designers to provide security proofs in a well-defined formal model in order to assure protocol implementers about the security properties of protocols.

Over the past decade, we have seen a number of PAKE protocols proposed in the three-party setting [4–8, 17–29]. Many of these published protocols either did not have a proof of security [5, 6, 17, 22–25] or were subsequently found to be flawed [4–10, 12, 23, 24, 27, 30–36]. There are only a handful of provably secure three-party PAKE protocols [4, 7, 8, 21] whose claimed security properties have not been invalidated. However, there are limitations in the security proof of these protocols. For example, the protocols of [7, 8] are proven secure in the random oracle model. Although a proof of security in the random oracle model is definitely better than having no proof, it may not guarantee security in the real world (currently an open question). The protocols of [4, 21] are proven secure in a restricted model where the adversary is not allowed to corrupt protocol participants. Note that a protocol proven secure in such a restricted model cannot guarantee its security against attacks by malicious clients including insider online/offline dictionary attacks. (Readers who are unfamiliar with formal security models are referred to Section 2.1.) Although Yang and Cao [37] proposed a new three-party key exchange protocol that was proven secure in the standard model, the protocol is based on the ElGamal encryption scheme and thus requires a server's public key as well as clients' passwords to be preestablished before the protocol is ever executed. We refer the readers to [33, 38–44] for other recently published protocols designed to work in a "hybrid" setting where a cryptographic key is required in addition to passwords.

To the best of our knowledge, there is no published three-party PAKE protocol whose security is proven secure in the standard model that allows an adversary to corrupt protocol participants. In this work, we present the first three-party PAKE protocol that achieves provable security in the standard model against an active adversary with the corruption capability. We prove the security of session keys for our protocol in the widely accepted indistinguishability-based model of Bellare, Pointcheval, and Rogaway [14]—this model is, probably, one of the most popular proof models in the provable security paradigm for key exchange protocols. However, the indistinguishability-based security of session keys proven in the Bellare-Pointcheval-Rogaway model (and several other standard models) does not imply the security of passwords against undetectable online dictionary attacks, in which each guess on the password is checked undetectably via an online transaction with the server (see Section 2.3 for more details). We address this problem by providing a separate proof of security for the protocol against undetectable online dictionary (UDOD) attacks. This second proof is compact and elegant and does not rely upon idealized assumptions about the cryptographic primitives. Table 1 compares our protocol against other provably secure three-party PAKE protocols in terms of security proofs.

The remainder of this paper is structured as follows. Section 2 describes a formal proof model along with the associated definitions of security. Section 3 presents our proposed three-party PAKE protocol. In Section 4, we prove that the proposed protocol achieves not only the typical indistinguishability-based security of session keys but also the password security against undetectable online dictionary attacks. We conclude the paper in Section 5.

2. Formal Setting

In this section, we

(1) first describe a security model adapted from the Bellare-Pointcheval-Rogaway 2000 model [14],

(2) define a typical indistinguishability-based security of session keys, which we call the *SK security*,

(3) provide a simple and intuitive definition of security against undetectable online dictionary attacks.

2.1. The Security Model

Protocol Participants. Let $\mathscr{C}$ be the set of all clients registered with the trusted authentication server S. Clients $C, C' \in \mathscr{C}$ who are both registered with S may run a three-party PAKE protocol P at any point in time to establish a session key. Let $\mathscr{U} = \mathscr{C} \cup \{S\}$. A party $U \in \mathscr{U}$ may have several instances involved in distinct, possibly concurrent, executions of protocol P. We use Π_U^i to denote the ith instance of party U. A client instance Π_C^i is said to *accept* when it successfully computes its session key sk_C^i in a protocol execution.

Long-Term Keys. Each client $C \in \mathscr{C}$ chooses a password pw_C from a fixed dictionary PW and shares it with the server S via a secure channel. Accordingly, S holds all the passwords $\{\mathrm{pw}_C \mid C \in \mathscr{C}\}$. Each password pw_C is used as the long-term secret key of C and S.

Table 1: Security proof comparison.

Protocol	Idealized assumption	Adversary capability	Resistance to UDOD attacks[†]
Our protocol	None	Not restricted	Proven
GPAKE [21]	None	Restricted from corrupting parties	No [4]
NGPAKE [4]	None		Not proven
Lin and Hwang [7]	Random oracles	Not restricted	Not proven
Wu et al. [8]	Random oracles	Not restricted	Not proven

[†]Resistance to undetectable online dictionary attacks.

Partnership. The notion of partnership is a key element in defining the security of the protocol. Two instances are *partners* if both participate in a protocol execution and establish a (shared) session key. We define the partnership relations between instances using the notions of session identifiers and partner identifiers (see [45] on the role and the possible construct of session and partner identifiers as a form of partnering mechanism that enables the right session key to be identified in concurrent protocol executions.). A session identifier (sid) is a unique identifier of a protocol session and is defined as a function of the messages transmitted in the protocol session. We use sid_U^i to denote the sid of instance Π_U^i. A partner identifier (pid) is the set of participants of a specific protocol session. Instances should receive as input a pid before they can run the protocol. By pid_U^i, we denote the pid given to instance Π_U^i. Notice that pid_C^i consists of three participants: server S, client C, and another client C' with whom Π_C^i believes it runs the protocol. We say that any two instances Π_C^i and $\Pi_{C'}^j$ are *partners* if (1) both Π_C^i and $\Pi_{C'}^j$ have accepted, (2) $\mathsf{sid}_C^i = \mathsf{sid}_{C'}^j$, and (3) $\mathsf{pid}_C^i = \mathsf{pid}_{C'}^j$.

Adversary. In the model, the probabilistic polynomial-time (PPT) adversary, $\mathcal{A}$, controls all the communications that take place between parties via a predefined set of oracle queries. For example, the adversary can ask participants to reveal session keys and passwords using Reveal and $\mathsf{Corrupt}$ queries as described below.

(i) $\mathsf{Execute}(\Pi_C^i, \Pi_{C'}^j, \Pi_S^k)$. This query models passive eavesdropping of a protocol execution. It prompts an honest execution of the protocol between the instances Π_C^i, $\Pi_{C'}^j$ and Π_S^k. The transcript of the protocol execution is returned as the output of the query.

(ii) $\mathsf{Send}(\Pi_U^i, m)$. This query models active attacks against the protocol. It sends message m to instance Π_U^i and returns the message that Π_U^i sends out in response to m. A query of the form $\mathsf{Send}(\Pi_C^i, \mathtt{start} : (C, C', S))$ prompts Π_C^i to initiate the protocol with $\mathsf{pid}_C^i = (C, C', S)$.

(iii) $\mathsf{Reveal}(\Pi_C^i)$. This query returns the session key sk_C^i. This query captures the notion of known key security (and it is often reasonable to assume that the adversary will be able to obtain session keys from any session different from the one under attack). Any client, Π_C^i, upon receiving such a query and if it has accepted and holds some session key, will send this session key back to $\mathcal{A}$. However, the adversary is not allowed to ask this query if it has already made a Test query to the instance Π_C^i or its partner instance (see below for explanation of the Test oracle).

(iv) $\mathsf{Corrupt}(U)$. This query captures not only the notion of forward secrecy but also unknown key share attacks and insider attacks. The query provides the adversary with U's password pw_U. Notice that a $\mathsf{Corrupt}$ query does not result in the release of the session keys since the adversary already has the ability to obtain session keys through Reveal queries. If $U = S$ (i.e., the server is corrupted), all clients' passwords stored by the server will be returned.

(v) $\mathsf{Test}(\Pi_C^i)$. This query is the only oracle query that does not correspond to any of the adversary's abilities. If Π_C^i has accepted with some session key and is being asked a $\mathsf{Test}(\Pi_C^i)$ query, then depending on a randomly chosen bit b, the adversary is given either the actual session key (when $b = 1$) or a session key drawn randomly from the session key distribution (when $b = 0$). The adversary can access the Test oracle as many times as necessary. All the queries to the oracle are answered using the same value of the hidden bit b. Namely, the keys returned by the Test oracle are either all real or all random. But, we require that for each different set of partners, the adversary should access the Test oracle only once.

We represent the number of queries used by an adversary as an ordered sequence of five nonnegative integers, $Q = (q_{\mathrm{ex}}, q_{\mathrm{se}}, q_{\mathrm{re}}, q_{\mathrm{co}}, q_{\mathrm{te}})$, where the five elements refer to the numbers of queries that the adversary made, respectively, to its $\mathsf{Execute}$, Send, Reveal, $\mathsf{Corrupt}$, and Test oracles. We call this usage of queries by an adversary the *query complexity* of the adversary.

2.2. Session Key (SK) Security.

We now define the basic security, called the SK security, of a 3-party PAKE protocol. As usual, we define the SK security via the notion of *freshness*. Intuitively, a fresh instance is one that holds a session key which should not be known to the adversary $\mathcal{A}$, and an unfresh instance is one whose session key (or some information about the key) can be known by trivial means. The formal definition of freshness is explained in Definition 1.

Definition 1. An instance Π_C^i is fresh if none of the following occurs: (1) $\mathscr{A}$ queries $\mathsf{Reveal}(\Pi_C^i)$ or $\mathsf{Reveal}(\Pi_{C'}^j)$, where $\Pi_{C'}^j$ is the partner of Π_C^i and (2) $\mathscr{A}$ queries $\mathsf{Corrupt}(U)$, for some $U \in \mathsf{pid}_C^i$, before Π_C^i or its partner $\Pi_{C'}^j$ accepts.

The SK security of a 3-party PAKE protocol P is defined in the context of the following two-stage experiment.

Stage 1. $\mathscr{A}$ makes any oracle queries at will except that:

(i) $\mathscr{A}$ is not allowed to ask the $\mathsf{Test}(\Pi_C^i)$ query if the instance Π_C^i is unfresh.

(ii) $\mathscr{A}$ is not allowed to ask the $\mathsf{Reveal}(\Pi_C^i)$ query if it has already made a Test query to Π_C^i or $\Pi_{C'}^j$, where $\Pi_{C'}^j$ is the partner of Π_C^i.

Stage 2. Once $\mathscr{A}$ decides that Phase 1 is over, it outputs a bit b' as a guess on the hidden bit b chosen by the Test oracle. $\mathscr{A}$ is said to succeed if $b = b'$.

Let Succ be the event that $\mathscr{A}$ succeeds in this experiment. Then we define the advantage of $\mathscr{A}$ in breaking the SK security of protocol P as

$$\begin{aligned} \mathsf{Adv}_P^{\mathrm{sk}}(\mathscr{A}) &= 2 \cdot \Pr[\mathsf{Succ}] - 1, \\ \mathsf{Adv}_P^{\mathrm{sk}}(t, Q) &= \max_{\mathscr{A}} \left\{ \mathsf{Adv}_P^{\mathrm{sk}}(\mathscr{A}) \right\}, \end{aligned} \tag{1}$$

where the maximum is over all PPT adversaries $\mathscr{A}$ with time complexity at most t and query complexity at most Q.

Definition 2. A 3-party PAKE protocol P is *SK-secure* if, for any PPT adversary $\mathscr{A}$ asking at most q_{se} Send queries, $\mathsf{Adv}_P^{\mathrm{sk}}(\mathscr{A})$ is only negligibly larger than $c \cdot q_{\mathrm{se}}/|\mathsf{PW}|$, where c is a very small constant (usually around 2 or 4) when compared with $|\mathsf{PW}|$.

2.3. Modelling Undetectable Online Dictionary Attacks. The SK security does not imply security against undetectable online dictionary attacks. In other words, a 3-party PAKE protocol that is not secure against an undetectable online dictionary attack may be rendered SK-secure. To see this, suppose that a 3-party PAKE protocol P is susceptible to undetectable online dictionary attacks whereby an attacker A can find out the password of any registered client B. Then, we can construct an adversary $\mathscr{A}$ who attacks protocol P with advantage 1 as follows.

Corruption. If A is a registered client, $\mathscr{A}$ queries $\mathsf{Corrupt}(A)$ to obtain the password pw_A. Otherwise, $\mathscr{A}$ skips this step.

Undetectable Online Dictionary Attacks. Next, $\mathscr{A}$ runs the protocol P in the same way as A conducts its undetectable online dictionary attacks against client B. Note that $\mathscr{A}$ can perfectly simulate A's attack by using the disclosed password pw_A and by asking oracle queries appropriately. At the end of this step, $\mathscr{A}$ will obtain the password pw_B as a result of the attacks.

Impersonation. $\mathscr{A}$ then initiates a new protocol session by querying $\mathsf{Send}(\Pi_C^i, \mathtt{start}\colon (B, C, S))$, where Π_C^i is an unused instance of an uncorrupted client C. $\mathscr{A}$ runs this session as per the protocol specification, but simulating by itself all the actions of B (by using pw_B). At the end of the session, the instance Π_C^i will accept with its session key sk_C^i.

Test. The instance Π_C^i is fresh as (1) no Reveal query has been made on Π_C^i or its partner (which does not exist) and (2) no $\mathsf{Corrupt}$ query has been made against any of B, C, and S. Thus, $\mathscr{A}$ may ask the $\mathsf{Test}(\Pi_C^i)$ query. Since $\mathscr{A}$ can compute the session key sk_C^i by itself, it follows that $\Pr_{P,\mathscr{A}}[\mathsf{Succ}] = 1$ and thus $\mathsf{Adv}_P^{\mathrm{sk}}(\mathscr{A}) = 1$.

Since verifying the correctness of a password guess may require more than one Send queries to be asked, $\mathscr{A}$ may have to ask Send queries as many times as $d \cdot |\mathsf{PW}|$, for some integer $d \geq 1$, to correctly determine the password pw_A. Then, even if $\mathsf{Adv}_P^{\mathrm{sk}}(\mathscr{A}) = 1$, the following holds for some $c \geq 1$:

$$\mathsf{Adv}_P^{\mathrm{sk}}(\mathscr{A}) \leq \frac{cd\,|\mathsf{PW}|}{|\mathsf{PW}|}, \tag{2}$$

and the protocol P is rendered SK-secure by Definition 2.

This result is not surprising since we call a protocol SK-secure if mounting an online dictionary attack by asking Send queries is the best an adversary can do. However, we want to be able to distinguish undetectable online dictionary attacks from detectable online dictionary attacks, and ensure that the best an adversary can do is to mount a detectable online dictionary attack. The following new definitions together provide a simple and intuitive way of capturing security against undetectable online dictionary attacks.

Definition 3. The $\mathsf{Send}(\Pi_S^k, m)$ query models an *online dictionary attack* if both the following are true at the time of the termination of instance Π_S^k: (1) m was not output by a previous Send query asked to an instance of C by which, Π_S^k believes, m was sent and (2) the adversary $\mathscr{A}$ queried neither $\mathsf{Corrupt}(S)$ nor $\mathsf{Corrupt}(C)$.

In Definition 3, the first condition implies that a straightforward delivery of a message between instances is not considered as an online dictionary attack while the second condition implies that, when C' is the (assumed) peer of client C, the adversary $\mathscr{A}$ can corrupt the peer client C' to mount an (insider) online dictionary attack. Note that our definition of an online dictionary attack does not impose any restriction on asking Reveal queries.

Consider the two-stage experiment described in the previous section. Let Undet be the event that in the experiment, a server instance terminates normally when an online dictionary attack was mounted against the instance. We say that the adversary $\mathscr{A}$ succeeds in mounting an undetectable online dictionary attack if the event Undet occurs.

Definition 4. A 3-party PAKE protocol P is secure against an undetectable online dictionary attack if, for any PPT adversary $\mathcal{A}$ asking at most q_{se} Send queries, $\Pr_{P,\mathcal{A}}[\mathsf{Undet}]$ is only negligibly larger than $c \cdot q_{\text{se}}/|\mathsf{PW}|$, where c is as in Definition 2.

3. Our Proposed Protocol

As we have earlier claimed, our proposed protocol presented in this section is the first three-party PAKE protocol proven secure in the standard model against an active adversary who has the corruption ability.

3.1. Preliminaries. We begin by reviewing some cryptographic primitives which underlie the security of our protocol.

Decisional Diffie-Hellman (DDH) Assumption. Let $\mathbb{G}$ be a cyclic (multiplicative) group of prime order q. Since the order of $\mathbb{G}$ is prime, all the elements of $\mathbb{G}$, except 1, are generators of $\mathbb{G}$. Let g be a random fixed generator of $\mathbb{G}$ and let x, y, z be randomly chosen elements in $\mathbb{Z}_q$ where $z \neq xy$. Informally stated, the DDH problem for $\mathbb{G}$ is to distinguish between the distributions of (g^x, g^y, g^{xy}) and (g^x, g^y, g^z), and the DDH assumption is said to hold in $\mathbb{G}$ if it is computationally infeasible to solve the DDH problem for $\mathbb{G}$. More formally, we define the advantage of $\mathcal{D}$ in solving the DDH problem for $\mathbb{G}$ as $\mathsf{Adv}_{\mathbb{G}}^{\text{ddh}}(\mathcal{D}) = |\Pr[\mathcal{D}(\mathbb{G}, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(\mathbb{G}, g, g^x, g^y, g^z) = 1]|$. We say that the DDH assumption holds in $\mathbb{G}$ if $\mathsf{Adv}_{\mathbb{G}}^{\text{ddh}}(\mathcal{D})$ is negligible for all PPT algorithms $\mathcal{D}$. We denote by $\mathsf{Adv}_{\mathbb{G}}^{\text{ddh}}(t)$ the maximum value of $\mathsf{Adv}_{\mathbb{G}}^{\text{ddh}}(\mathcal{D})$ over all algorithms $\mathcal{D}$ running in time at most t.

Message Authentication Codes. Let $\Sigma = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Ver})$ be a message authentication code (MAC) scheme. The key generation algorithm Gen takes as input a security parameter 1^ℓ and outputs a key k chosen uniformly at random from $\{0,1\}^\ell$. The MAC generation algorithm Mac takes as input a key k and a message m and outputs a MAC (also known as a tag) σ. The MAC verification algorithm Ver takes as input a key k, a message m, and a MAC σ and outputs 1 if σ is valid for m under k or outputs 0 if σ is invalid. Let $\mathsf{Adv}_{\Sigma}^{\text{euf-cma}}(\mathcal{A})$ be the probability that an adversary $\mathcal{A}$ succeeds in breaking the existential unforgeability of Σ under adaptive chosen message attacks. We say that the MAC scheme Σ is secure if $\mathsf{Adv}_{\Sigma}^{\text{euf-cma}}(\mathcal{A})$ is negligible for every PPT adversary $\mathcal{A}$. We use $\mathsf{Adv}_{\Sigma}^{\text{euf-cma}}(t, q_{\text{mac}}, q_{\text{ver}})$ to denote the maximum value of $\mathsf{Adv}_{\Sigma}^{\text{euf-cma}}(\mathcal{A})$ over all PPT adversaries $\mathcal{A}$ running in time at most t and asking at most q_{mac} and q_{ver} queries to its MAC generation and verification oracles, respectively.

Two-Party PAKE Protocols. Let $\mathsf{2PAKE}$ be a two-party PAKE protocol that outputs session keys distributed in $\{0,1\}^\ell$. We assume that $\mathsf{2PAKE}$ is SK-secure against an adversary who is given access to all the oracles: Send, $\mathsf{Execute}$, Reveal, $\mathsf{Corrupt}$, and Test. Let $\mathsf{Adv}_{\mathsf{2PAKE}}^{\mathsf{sk}}(\mathcal{A})$ be the advantage of an adversary $\mathcal{A}$ in breaking the SK security of $\mathsf{2PAKE}$. We require that, for all PPT adversaries $\mathcal{A}$ making at most q_{se} Send queries, $\mathsf{Adv}_{\mathsf{2PAKE}}^{\mathsf{sk}}(\mathcal{A})$ is only negligibly larger than $q_{\text{se}}/|\mathsf{PW}|$. We denote by $\mathsf{ADV}_{\mathsf{2PAKE}}^{\mathsf{sk}}(t, Q)$ the maximum value of $\mathsf{Adv}_{\mathsf{2PAKE}}^{\mathsf{sk}}(\mathcal{A})$ over all PPT adversaries $\mathcal{A}$ with time complexity at most t and query complexity at most Q.

3.2. Protocol Description. Let A and B be two clients who wish to establish a session key, and let S be a trusted server with which A and B have secretly shared their respective passwords pw_A and pw_B. Our protocol proceeds as follows.

Step 1. A and S establish a shared secret key k_{AS} by running the two-party protocol $\mathsf{2PAKE}$. Likewise, B and S establish a shared secret key k_{BS}.

Step 2. A (resp., B and S) selects a nonce n_A (resp., n_B and n_S) at random from $\mathbb{Z}_q$ and sends $A \parallel n_A$ (resp., $B \parallel n_B$ and $S \parallel n_S$) to the other two parties. All the parties (A, B, and S) define their session identifiers as $\mathsf{sid}_A = \mathsf{sid}_B = \mathsf{sid}_S = A \parallel n_A \parallel B \parallel n_B \parallel S \parallel n_S$.

Step 3. A chooses a random $x \in \mathbb{Z}_q$, computes $X = g^x$ and $\sigma_{AS} = \mathsf{Mac}_{k_{AS}}(A \parallel X \parallel \mathsf{sid}_A)$, and sends $A \parallel X \parallel \sigma_{AS}$ to S. Meanwhile, B chooses a random $y \in \mathbb{Z}_q$, computes $Y = g^y$ and $\sigma_{BS} = \mathsf{Mac}_{k_{BS}}(B \parallel Y \parallel \mathsf{sid}_B)$, and sends $B \parallel Y \parallel \sigma_{BS}$ to S.

Step 4. S checks that $\mathsf{Ver}_{k_{AS}}(A \parallel X \parallel \mathsf{sid}_S, \sigma_{AS}) = 1$ and $\mathsf{Ver}_{k_{BS}}(B \parallel Y \parallel \mathsf{sid}_S, \sigma_{BS}) = 1$. If either of these is untrue, S aborts the protocol. Otherwise, S computes $\sigma_{SA} = \mathsf{Mac}_{k_{AS}}(S \parallel Y \parallel \mathsf{sid}_S)$ and $\sigma_{SB} = \mathsf{Mac}_{k_{BS}}(S \parallel X \parallel \mathsf{sid}_S)$ and sends $S \parallel Y \parallel \sigma_{SA}$ and $S \parallel X \parallel \sigma_{SB}$ to A and B, respectively.

Step 5. A computes the session key $\text{sk} = Y^x$ if $\mathsf{Ver}_{k_{AS}}(S \parallel Y \parallel \mathsf{sid}_A, \sigma_{SA}) = 1$, while B computes the session key $\text{sk} = X^y$ if $\mathsf{Ver}_{k_{BS}}(S \parallel X \parallel \mathsf{sid}_B, \sigma_{SB}) = 1$. A and B abort the protocol if their verification fails.

The operation of this protocol is illustrated in Figure 1. Steps 1 and 2 of the protocol are independent and can be performed in parallel. The session-key computation in the protocol is the same as in the Diffie-Hellman key exchange protocol (i.e., $\text{sk} = g^{xy}$). Hence, it is straightforward to verify the correctness of the protocol.

4. Security Proofs

In this section we prove that our three-party PAKE protocol is SK-secure and is resistant to undetectable online dictionary attacks. The proofs of both properties rely on neither random oracles nor ideal ciphers. Therefore, if $\mathsf{2PAKE}$ is instantiated with a protocol proven secure in the standard model (e.g., [3, 49]), our three-party PAKE protocol would also be provably secure in the standard model. Hereafter, we denote our protocol by $\mathsf{3PAKEsm}$ ("sm" for "standard model").

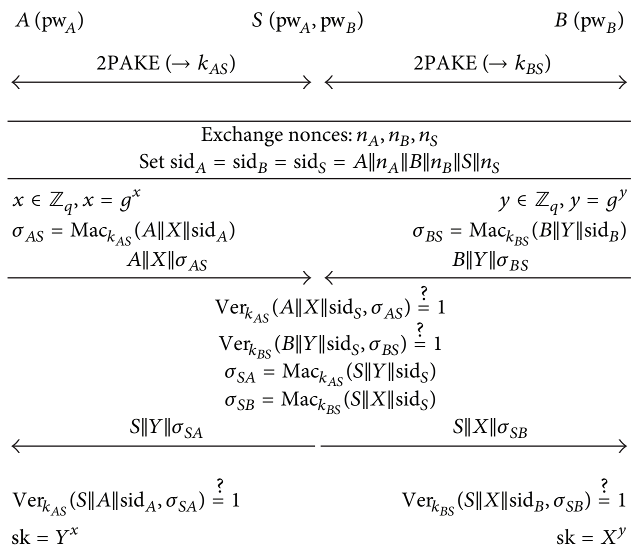
$A\,(\mathrm{pw}_A)$
$S\,(\mathrm{pw}_A, \mathrm{pw}_B)$
$B\,(\mathrm{pw}_B)$
2PAKE ($\to k_{AS}$)
2PAKE ($\to k_{BS}$)
Exchange nonces: n_A, n_B, n_S
Set $\mathrm{sid}_A = \mathrm{sid}_B = \mathrm{sid}_S = A\|n_A\|B\|n_B\|S\|n_S$
$x \in \mathbb{Z}_q, x = g^x$
$y \in \mathbb{Z}_q, y = g^y$
$\sigma_{AS} = \mathrm{Mac}_{k_{AS}}(A\|X\|\mathrm{sid}_A)$
$\sigma_{BS} = \mathrm{Mac}_{k_{BS}}(B\|Y\|\mathrm{sid}_B)$
$A\|X\|\sigma_{AS}$
$B\|Y\|\sigma_{BS}$
$\mathrm{Ver}_{k_{AS}}(A\|X\|\mathrm{sid}_S, \sigma_{AS}) \overset{?}{=} 1$
$\mathrm{Ver}_{k_{BS}}(B\|Y\|\mathrm{sid}_S, \sigma_{BS}) \overset{?}{=} 1$
$\sigma_{SA} = \mathrm{Mac}_{k_{AS}}(S\|Y\|\mathrm{sid}_S)$
$\sigma_{SB} = \mathrm{Mac}_{k_{BS}}(S\|X\|\mathrm{sid}_S)$
$S\|Y\|\sigma_{SA}$
$S\|X\|\sigma_{SB}$
$\mathrm{Ver}_{k_{AS}}(S\|A\|\mathrm{sid}_A, \sigma_{SA}) \overset{?}{=} 1$
$\mathrm{Ver}_{k_{BS}}(S\|X\|\mathrm{sid}_B, \sigma_{SB}) \overset{?}{=} 1$
$\mathrm{sk} = Y^x$
$\mathrm{sk} = X^y$

FIGURE 1: The proposed three-party PAKE protocol.

4.1. Proof of SK Security. We first claim that, if the underlying two-party protocol 2PAKE is SK-secure, then the 3PAKEsm protocol is SK-secure as well under the DDH assumption in $\mathbb{G}$ and the security of the MAC scheme Σ.

Theorem 5. *Let $Q = (q_{ex}, q_{se}, q_{re}, q_{co}, q_{te})$. For any adversary with query complexity at most Q and time complexity at most t, its advantage in attacking protocol* 3PAKEsm *is bounded by*

$$\begin{aligned}\mathsf{Adv}^{sk}_{\mathsf{3PAKEsm}}(t, Q) \le{}& 2\cdot \mathsf{Adv}^{sk}_{\mathsf{2PAKE}}\left(t', Q'\right) + \frac{\left(q_{se}+q_{ex}\right)^2}{|\mathbb{G}|}\\ &+ 2\cdot q_{se}\cdot \mathsf{Adv}^{euf\text{-}cma}_{\Sigma}\left(t', 4, 4\right)\\ &+ 4\cdot \mathsf{Adv}^{ddh}_{\mathbb{G}}\left(t'\right),\end{aligned} \tag{3}$$

where $Q' = (2q_{ex}, q_{se}, 0, q_{co}, 2q_{ex} + q_{se})$ and t' is the maximum time required to perform an entire experiment involving an adversary who attacks protocol 3PAKEsm *with time complexity t.*

Proof. Assume an adversary $\mathscr{A}$ attacking protocol 3PAKEsm with time complexity t and query complexity $Q = (q_{\mathrm{ex}}, q_{\mathrm{se}}, q_{\mathrm{re}}, q_{\mathrm{co}}, q_{\mathrm{te}})$. We prove Theorem 5 by introducing a sequence of experiments $\mathbf{Expr}_0, \ldots, \mathbf{Expr}_5$ and bounding the difference in $\mathscr{A}$'s advantage between two consecutive experiments. $\mathbf{Expr}_0$ is the original experiment (described in Section 2.2) in which $\mathscr{A}$ attacks the actual protocol, and $\mathbf{Expr}_5$ is the experiment in which the advantage of $\mathscr{A}$ is 0. Let Succ_i be the event that $\mathscr{A}$ correctly guesses the hidden bit b (chosen by the Test oracle) in experiment $\mathbf{Expr}_i$. By definition, we get $\mathsf{Adv}^{\mathsf{sk}}_{\mathsf{3PAKEsm}}(\mathscr{A}) = 2\cdot \Pr[\mathsf{Succ}_0] - 1$.

Before providing details of the proof, we first define the notion of an *uncorrupted* instance. □

Definition 6. We say an instance Π^i_U is clean if no one in pid^i_U has been asked a Corrupt query. Otherwise, we say it is unclean.

Experiment $\mathbf{Expr}_1$. We modify the experiment so that each different MAC key is chosen uniformly at random from $\{0,1\}^\ell$ for all clean instances. The difference in $\mathscr{A}$'s success probability between $\mathbf{Expr}_0$ and $\mathbf{Expr}_1$ is bounded by

Claim 1.

$$\left|\Pr\left[\mathsf{Succ}_1\right] - \Pr\left[\mathsf{Succ}_0\right]\right| \le \mathsf{Adv}^{\mathsf{sk}}_{\mathsf{2PAKE}}\left(t', Q'\right). \tag{4}$$

Proof. Assume that the advantage of $\mathscr{A}$ in attacking protocol 3PAKEsm is different between two experiments $\mathbf{Expr}_0$ and $\mathbf{Expr}_1$. Then we prove the claim by constructing, from $\mathscr{A}$, an adversary $\mathscr{A}_{\mathsf{2PAKE}}$ attacking protocol 2PAKE with time complexity t' and query complexity Q'.

$\mathscr{A}_{\mathsf{2PAKE}}$ begins by choosing a bit b uniformly at random. $\mathscr{A}_{\mathsf{2PAKE}}$ then invokes $\mathscr{A}$ as a subroutine and answers the oracle queries of $\mathscr{A}$ on its own as follows.

Execute *Queries.* $\mathscr{A}_{\mathsf{2PAKE}}$ answers Execute queries of $\mathscr{A}$ by making Execute and Test queries to its own oracles. Specifically, $\mathscr{A}_{\mathsf{2PAKE}}$ handles each Execute(Π^i_A, Π^j_B, Π^k_S) query as follows.

(i) If anyone in $\{A, B, S\}$ has been corrupted, then $\mathscr{A}_{\mathsf{2PAKE}}$ answers the Execute query as in experiment $\mathbf{Expr}_0$.

(ii) Otherwise, $\mathscr{A}_{\mathsf{2PAKE}}$ first makes two queries Execute(Π^i_A, Π^k_S) and Execute($\Pi^j_B, \Pi^{k'}_S$). Let $\mathsf{T}_{\mathsf{2PAKE}}$ and $\mathsf{T}'_{\mathsf{2PAKE}}$ be two transcripts returned in response to the Execute queries. Next, $\mathscr{A}_{\mathsf{2PAKE}}$ makes the queries Test(Π^i_A) and Test(Π^j_B) and receives in return two keys $\overline{k}_{AS}$ and $\overline{k}_{BS}$ (either real or random). $\mathscr{A}_{\mathsf{2PAKE}}$ then generates the messages of Steps 2–4 of protocol 3PAKEsm, using $\overline{k}_{AS}$ and $\overline{k}_{BS}$ as the MAC keys. Finally, $\mathscr{A}_{\mathsf{2PAKE}}$ returns these messages prepended by $\mathsf{T}_{\mathsf{2PAKE}}$ and $\mathsf{T}'_{\mathsf{2PAKE}}$.

Send *Queries.* At a high level, the simulation of the Send oracle is similar to that of the Execute oracle. Specifically, $\mathscr{A}_{\mathsf{2PAKE}}$ handles each Send(Π^i_U, m) query as follows.

(i) If the instance Π^i_U is clean or the message m belongs to Step 2 or later steps, then $\mathscr{A}_{\mathsf{2PAKE}}$ answers the query as in experiment $\mathbf{Expr}_0$.

(ii) Otherwise, $\mathscr{A}_{\mathsf{2PAKE}}$ answers it by making the same query to its own Send oracle. If the query causes Π^i_U to accept, then $\mathscr{A}_{\mathsf{2PAKE}}$ also makes a Test(Π^i_U) query (if it had not previously asked a Test query to the partner of Π^i_U). As in the simulation of the Execute oracle, $\mathscr{A}_{\mathsf{2PAKE}}$ uses the output of this Test query as the MAC key in generating the messages of Steps 2–4 of protocol 3PAKEsm.

Reveal *Queries*. These queries are answered in the obvious way. Namely, $\mathscr{A}_{\mathsf{2PAKE}}$ responds to the query $\mathsf{Reveal}(\Pi_C^i)$ by returning the session key sk_C^i.

$\mathsf{Corrupt}$ *Queries*. When $\mathscr{A}$ queries $\mathsf{Corrupt}(U)$, $\mathscr{A}_{\mathsf{2PAKE}}$ makes the same query to its own $\mathsf{Corrupt}$ oracle and simply forwards the output to $\mathscr{A}$.

Test *Queries*. $\mathscr{A}_{\mathsf{2PAKE}}$ answers these queries according to the bit b chosen at the beginning of the simulation. That is, $\mathscr{A}_{\mathsf{2PAKE}}$ returns real session keys, which it has computed on its own, if $b = 1$, and otherwise returns random keys chosen uniformly at random from $\mathbb{G}$.

Now at some point in time, when $\mathscr{A}$ terminates and outputs its guess b', $\mathscr{A}_{\mathsf{2PAKE}}$ outputs 1 if $b = b'$ and outputs 0 otherwise.

From the simulation above, it is easy to see that $\mathscr{A}_{\mathsf{2PAKE}}$ has at most time complexity t' and query complexity Q'. The advantage of $\mathscr{A}_{\mathsf{2PAKE}}$ in attacking protocol $\mathsf{2PAKE}$ is immediate if we notice the following.

(i) The probability that $\mathscr{A}_{\mathsf{2PAKE}}$ outputs 1 when its Test oracle returns real session keys is equal to $\Pr[\mathsf{Succ}_0]$, the probability that $\mathscr{A}$ correctly guesses the bit b in experiment $\mathbf{Expr}_0$.

(ii) The probability that $\mathscr{A}_{\mathsf{2PAKE}}$ outputs 1 when its Test oracle returns random keys is equal to $\Pr[\mathsf{Succ}_1]$, the probability that $\mathscr{A}$ correctly guesses the bit b in experiment $\mathbf{Expr}_1$.

This means that $\mathsf{Adv}^{\mathsf{sk}}_{\mathsf{2PAKE}}(\mathscr{A}_{\mathsf{2PAKE}}) = |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_0]|$. Claim 1 then follows. □

Experiment $\mathbf{Expr_2}$. Let Repeat be the event that a nonce selected by an instance of a party is selected again by another instance of the same party. The experiment $\mathbf{Expr}_2$ is aborted, and the adversary does not succeed, if the event Repeat occurs. This is the only difference between $\mathbf{Expr}_1$ and $\mathbf{Expr}_2$. By a straightforward calculation, we get the following.

Claim 2.

$$\left|\Pr\left[\mathsf{Succ}_2\right] - \Pr\left[\mathsf{Succ}_1\right]\right| \le \frac{\left(q_{\mathrm{se}} + q_{\mathrm{ex}}\right)^2}{2\left|\mathbb{G}\right|}. \quad (5)$$

Experiment $\mathbf{Expr_3}$. Let Forge be the event that the adversary $\mathscr{A}$ makes a Send query of the form $\mathsf{Send}(\Pi_U^i, V \parallel * \parallel \sigma)$ before querying $\mathsf{Corrupt}(W)$, for some $W \in \mathsf{pid}_U^i$, such that (1) σ is a valid tag on $V \parallel * \parallel \mathsf{sid}_U^i$ and (2) no oracle had not previously generated a tag on $V \parallel * \parallel \mathsf{sid}_U^i$. If Forge occurs, this experiment is aborted and the adversary does not succeed. Then we have the following.

Claim 3.

$$\left|\Pr\left[\mathsf{Succ}_3\right] - \Pr\left[\mathsf{Succ}_2\right]\right| \le q_{\mathrm{se}} \cdot \mathsf{Adv}_{\Sigma}^{\text{euf-cma}}\left(t', 4, 4\right). \quad (6)$$

Proof. Assuming that the event Forge occurs, we construct, from $\mathscr{A}$, an algorithm $\mathscr{F}$ who outputs, with a nonnegligible probability, a forgery against the MAC scheme Σ. The algorithm $\mathscr{F}$ is given oracle access to $\mathsf{Mac}_k(\cdot)$ and $\mathsf{Ver}_k(\cdot)$. The goal of $\mathscr{F}$ is to produce a message/tag pair (m, σ) such that (1) σ is a valid tag on the message m (i.e., $\mathsf{Ver}_k(m, \sigma) = 1$) and (2) $\mathscr{F}$ had not previously queried its oracle $\mathsf{Mac}_k(\cdot)$ on the message m.

Let n be the number of all active sessions that $\mathscr{A}$ initiates by asking a Send query. First, $\mathscr{F}$ chooses a random $\alpha \in \{1, \ldots, n\}$. $\mathscr{F}$ then simulates the oracle calls of $\mathscr{A}$ as in experiment $\mathbf{Expr}_2$; except that in the αth session, it answers Send queries by accessing its MAC generation and verification oracles. If Forge occurs in the αth session, $\mathscr{F}$ halts and outputs the message/tag pair generated by $\mathscr{A}$ as its forgery. Otherwise, $\mathscr{F}$ halts and outputs a failure indication. This simulation is perfect unless the adversary $\mathscr{A}$ makes a $\mathsf{Corrupt}$ query against a participant of the αth session. But note that the event of $\mathscr{A}$ making such a $\mathsf{Corrupt}$ query should not happen if Forge occurs in the αth session.

From the simulation, it is immediate that $\mathsf{Adv}_{\Sigma}^{\text{euf-cma}}(\mathscr{F}) = \Pr[\mathsf{Forge}]/n$. Since $n \le q_{\mathrm{se}}$, we get $\Pr[\mathsf{Forge}] \le q_{\mathrm{se}} \cdot \mathsf{Adv}_{\Sigma}^{\text{euf-cma}}(\mathscr{F})$. Then, Claim 3 follows by noticing that $\mathscr{F}$ has at most time complexity t' and makes at most 4 queries to $\mathsf{Mac}_k(\cdot)$ and $\mathsf{Ver}_k(\cdot)$. □

Experiment $\mathbf{Expr_4}$. This experiment is different from $\mathbf{Expr}_3$ in that the session key sk of each pair of instances partnered via an $\mathsf{Execute}$ query is chosen uniformly at random from $\mathbb{G}$ instead of being computed as $\mathrm{sk} = g^{xy} = X^y = Y^x$. As the following claim states, the difference in $\mathscr{A}$'s advantage between $\mathbf{Expr}_3$ and $\mathbf{Expr}_4$ is negligible if the DDH assumption holds in $\mathbb{G}$.

Claim 4.

$$\left|\Pr\left[\mathsf{Succ}_4\right] - \Pr\left[\mathsf{Succ}_3\right]\right| \le \mathsf{Adv}_{\mathbb{G}}^{\mathrm{ddh}}\left(t'\right). \quad (7)$$

Proof. Assume that the advantage of $\mathscr{A}$ is nonnegligibly different between $\mathbf{Expr}_3$ and $\mathbf{Expr}_4$. We prove the claim by constructing, from $\mathscr{A}$, a distinguisher $\mathscr{D}$ that solves the DDH problem in $\mathbb{G}$. Let $(g_1, g_2, g_3) \in \mathbb{G}^3$ be an instance of the DDH problem given as input to $\mathscr{D}$. $\mathscr{D}$ begins by choosing a bit b uniformly at random. $\mathscr{D}$ then invokes $\mathscr{A}$ as a subroutine and proceeds to simulate the oracles. $\mathscr{D}$ answers all the oracle queries of $\mathscr{A}$ as in experiment $\mathbf{Expr}_3$, except that it handles each $\mathsf{Execute}(\Pi_A^i, \Pi_B^j, \Pi_S^k)$ query by

(1) selecting two random $a_i, b_i \in \mathbb{Z}_q$,

(2) computing $X' = g_1^{a_i}$ and $Y' = g_2^{b_i}$,

(3) returning a transcript generated with X' and Y' in place of X and Y,

(4) then setting $\mathrm{sk}_A^i = \mathrm{sk}_B^j = g_3^{a_i b_i}$.

Let b' be the output of $\mathscr{A}$. $\mathscr{D}$ outputs 1 if $b = b'$ and outputs 0, otherwise.

Then, the following is clear:

(i) The probability that $\mathscr{D}$ outputs 1 on a true Diffie-Hellman triple is exactly the probability that $\mathscr{A}$ correctly guesses the bit b in experiment $\mathbf{Expr}_3$.

(ii) The probability that $\mathscr{D}$ outputs 1 on a random triple is exactly the probability that $\mathscr{A}$ correctly guesses the bit b in experiment $\mathbf{Expr}_4$.

This completes the proof of Claim 4. □

Experiment $\mathbf{Expr}_5$. In this experiment, the session key sk_C^i of each instance Π_C^i activated by a Send query is chosen uniformly at random from $\mathbb{G}$ if no one in pid_C^i has been corrupted before Π_C^i determines its session identifier sid_C^i. The difference in $\mathscr{A}$'s advantage between $\mathbf{Expr}_4$ and $\mathbf{Expr}_5$ is bounded by the following.

Claim 5.

$$\left|\Pr\left[\mathsf{Succ}_5\right] - \Pr\left[\mathsf{Succ}_4\right]\right| \le \mathsf{Adv}_{\mathbb{G}}^{\mathrm{ddh}}\left(t'\right). \tag{8}$$

Proof. The proof of this claim is essentially similar to that of Claim 4. From the adversary $\mathscr{A}$ whose advantage is non-negligibly different between $\mathbf{Expr}_4$ and $\mathbf{Expr}_5$, we construct a distinguisher $\mathscr{D}$ that solves the DDH problem in $\mathbb{G}$. Let $(g_1, g_2, g_3) \in \mathbb{G}^3$ be an instance of the DDH problem given as input to $\mathscr{D}$. $\mathscr{D}$ begins by selecting a bit b uniformly at random and generating a list DDHList which is used to link an instance of the DDH problem to a session identifier, $\mathscr{D}$ then runs $\mathscr{A}$ as a subroutine and simulates the oracles. It handles all the queries of $\mathscr{A}$ as in experiment $\mathbf{Expr}_4$ except for Send queries.

Consider a query of the form $\mathsf{Send}(\Pi_C^i, U \parallel n_U)$ which delivers a random nonce n_U to instance Π_C^i. Whenever such a query is made, $\mathscr{D}$ answers it as follows.

(i) If n_U is not the last nonce that Π_C^i is expected to receive, $\mathscr{D}$ simply waits for the next nonce.

(ii) Otherwise, $\mathscr{D}$ defines sid_C^i and checks that anyone in pid_C^i was corrupted.

  (a) If so, $\mathscr{D}$ responds to the query as in experiment $\mathbf{Expr}_4$.

  (b) If not, $\mathscr{D}$ checks if the list DDHList contains an entry of the form $(\mathsf{sid}_C^i, X', Y', Z')$, where $X', Y', Z' \in \mathbb{G}$.

    (1) If it does, $\mathscr{D}$ computes $\sigma_{\mathrm{CS}} = \mathsf{Mac}_{k_{\mathrm{CS}}}(C \parallel Y' \parallel \mathsf{sid}_C^i)$ and returns $C \parallel Y' \parallel \sigma_{\mathrm{CS}}$ in response to the query.

    (2) Otherwise, $\mathscr{D}$ selects two random $a_i, b_i \in \mathbb{Z}_q$, computes $X' = g_1^{a_i}, Y' = g_2^{b_i}, Z' = g_3^{a_i b_i}$, and $\sigma_{\mathrm{CS}} = \mathsf{Mac}_{k_{\mathrm{CS}}}(C \parallel X' \parallel \mathsf{sid}_C^i)$, returns $C \parallel X' \parallel \sigma_{\mathrm{CS}}$ to $\mathscr{A}$, and finally adds the tuple $(\mathsf{sid}_C^i, X', Y', Z')$ to DDHList.

When $\mathscr{A}$ makes a Send query that causes an instance Π_C^i to accept, $\mathscr{D}$ checks if DDHList contains an entry of the form $(\mathsf{sid}_C^i, X', Y', Z')$. If so, $\mathscr{D}$ sets $\mathsf{sk}_C^i = Z'$. Otherwise, $\mathscr{D}$ computes sk_C^i as in experiment $\mathbf{Expr}_4$. For all other Send queries of $\mathscr{A}$, $\mathscr{D}$ answers them as in experiment $\mathbf{Expr}_4$. Now when $\mathscr{A}$ terminates and outputs its guess b', $\mathscr{D}$ outputs 1 if $b = b'$ and outputs 0 otherwise.

One can easily see the following.

(i) The probability that $\mathscr{D}$ outputs 1 on a true Diffie-Hellman triple is exactly the probability that $\mathscr{A}$ correctly guesses the bit b in experiment $\mathbf{Expr}_4$.

(ii) The probability that $\mathscr{D}$ outputs 1 on a random triple is exactly the probability that $\mathscr{A}$ correctly guesses the bit b in experiment $\mathbf{Expr}_5$.

This implies Claim 5. □

In experiment $\mathbf{Expr}_5$, the session keys of all fresh instances are chosen uniformly at random from $\mathbb{G}$ and thus the adversary $\mathscr{A}$ obtains no information on the bit b chosen by the Test oracle. Therefore, it follows that $\Pr[\mathsf{Succ}_5] = 1/2$. This result combined with the previous claims yields the statement of Theorem 5.

4.2. Proof of Resistance to Undetectable Online Dictionary Attacks. We now claim that 3PAKEsm is secure against undetectable online dictionary attacks as long as the 2PAKE protocol is SK-secure.

Theorem 7. *Let* Undet *be as defined in Section 2.3 and assume that for any* PPT *adversary* $\mathscr{A}'$ *asking at most* q_{se} Send *queries,* $\mathsf{Adv}_{\mathsf{2PAKE}}^{\mathsf{sk}}(\mathscr{A}')$ *is only negligibly larger than* $q_{\mathrm{se}}/|\mathsf{PW}|$. *Then, for any* PPT *adversary* $\mathscr{A}$ *asking at most* q_{se} Send *queries,* $\Pr_{\mathsf{3PAKEsm},\mathscr{A}}[\mathsf{Undet}]$ *is only negligibly larger than* $2 \cdot q_{\mathrm{se}}/|\mathsf{PW}|$.

Proof. Let $\mathscr{A}$ be a PPT adversary who asks q_{se} Send queries in mounting an undetectable online dictionary attack against 3PAKEsm. Consider the experiment $\mathbf{Expr}_1$ described in the proof of Theorem 5 (see Section 4.1). By Undet_1 (resp., Undet_0), we denote the event Undet defined in experiment $\mathbf{Expr}_1$ (resp., $\mathbf{Expr}_0$). We prove Theorem 7 by first proving Claim 6 and then Claim 7. □

Claim 6. $|\Pr_{\mathsf{3PAKEsm},\mathscr{A}}[\mathsf{Undet}_1] - \Pr_{\mathsf{3PAKEsm},\mathscr{A}}[\mathsf{Undet}_0]|$ is only negligibly larger than $q_{\mathrm{se}}/|\mathsf{PW}|$.

Claim 7. $\Pr_{\mathsf{3PAKEsm},\mathscr{A}}[\mathsf{Undet}_1]$ is only negligibly larger than $q_{\mathrm{se}}|\mathsf{PW}|$.

Proof of Claim 6. We prove the claim by constructing an adversary $\mathscr{A}'$ who attacks the SK security of 2PAKE with advantage equal to $|\Pr_{\mathsf{3PAKEsm},\mathscr{A}}[\mathsf{Undet}_1]-\Pr_{\mathsf{3PAKEsm},\mathscr{A}}[\mathsf{Undet}_0]|$.

$\mathscr{A}'$ chooses a random bit $b \in \{0, 1\}$ and invokes the adversary $\mathscr{A}$ as a subroutine. $\mathscr{A}'$ then simulates the oracles for $\mathscr{A}$ in the exactly same way as in the simulation for the proof of Claim 1. $\mathscr{A}'$ outputs 1 if Undet occurs and 0 otherwise. From the way the oracles are simulated, it is easy to see the following.

(i) The probability that $\mathscr{A}'$ outputs 1 when its Test oracle returns real session keys is equal to the probability that the event Undet occurs in experiment $\mathbf{Expr}_0$.

(ii) The probability that $\mathcal{A}'$ outputs 1 when its Test oracle returns random keys is equal to the probability that the event Undet occurs in experiment $\mathbf{Expr}_1$.

Since $\mathcal{A}'$ makes at most q_{se} Send queries, we obtain the statement of Claim 6. □

Proof of Claim 7. Assume that $\mathsf{Pr}_{\mathsf{3PAKEsm},\mathcal{A}}[\mathsf{Undet}_1]$ is non-negligibly larger than $q_{\mathsf{se}}/|\mathsf{PW}|$. Given the adversary $\mathcal{A}$, we construct an adversary $\mathcal{A}'$ against $\mathsf{2PAKE}$ who asks at most q_{se} Send queries but has an advantage nonnegligibly larger than $q_{\mathsf{se}}/|\mathsf{PW}|$.

$\mathcal{A}'$ runs $\mathcal{A}$ as a subroutine while simulating the oracles on its own. $\mathcal{A}'$ handles all the oracle queries of $\mathcal{A}$ as in the experiment $\mathbf{Expr}_1$ except for Send queries. When $\mathcal{A}$ makes a $\mathsf{Send}(\Pi_U^i, m)$ query, $\mathcal{A}'$ checks if m is a message for initiating a new session (of $\mathsf{3PAKEsm}$) or the Send query belongs to an execution of $\mathsf{2PAKE}$.

(1) If both are untrue, $\mathcal{A}'$ responds to the query as in experiment $\mathbf{Expr}_1$.

(2) Otherwise, $\mathcal{A}'$ answers it by making the same query to its own Send oracle. If the query prompts Π_U^i to accept, then $\mathcal{A}'$ checks if Π_U^i is clean.

  (a) If so, $\mathcal{A}'$ sets the MAC key of Π_U^i to be a random key drawn uniformly from $\{0,1\}^{\ell}$.

  (b) Otherwise, $\mathcal{A}'$ makes a $\mathsf{Reveal}(\Pi_U^i)$ query and sets the MAC key of Π_U^i to be the output of this Reveal query.

Let Π_S^t be any server instance against which $\mathcal{A}$ has mounted an online dictionary attack. Let k_S^t be the session key (i.e., the MAC key) that the instance Π_S^t has computed in its execution of $\mathsf{2PAKE}$. In order for the instance Π_S^t to terminate normally, the adversary $\mathcal{A}$ has to make a query of the form $\mathsf{Send}(\Pi_S^t, C \parallel * \parallel \sigma_{CS})$ such that $\mathsf{Ver}_{k_S^t}(C \parallel * \parallel \mathsf{sid}_S^t, \sigma_{CS}) = 1$. When $\mathcal{A}$ makes such a Send query (i.e., when the event Undet_1 occurs), $\mathcal{A}'$ makes a Test query against the instance Π_S^t. Note that the instance Π_S^t is fresh as (1) it is partnered with no instance and (2) S and C must have not been corrupted. Let $\overline{k}_S^t$ be the key returned in response to the Test query. $\mathcal{A}'$ outputs 1 if $\mathsf{Ver}_{\overline{k}_S^t}(C \parallel * \parallel \mathsf{sid}_S^t, \sigma_{CS}) = 1$ and outputs 0, otherwise. If Undet_1 does not occur, $\mathcal{A}'$ outputs a random bit. Then, it is not hard to see that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sk}}_{\mathsf{2PAKE}}(\mathcal{A}') &= 2 \cdot \Pr_{\mathsf{2PAKE},\mathcal{A}'}[\mathsf{Succ}] - 1 \\ &= 2 \cdot \Big(\Pr_{\mathsf{3PAKEsm},\mathcal{A}}[\mathsf{Undet}_1] \\ &\qquad + \frac{1}{2}\left(1 - \Pr_{\mathsf{3PAKEsm},\mathcal{A}}[\mathsf{Undet}_1]\right)\Big) - 1 \\ &= \Pr_{\mathsf{3PAKEsm},\mathcal{A}}[\mathsf{Undet}_1].\end{aligned} \tag{9}$$

This completes the proof of Claim 7. □

Theorem 7 immediately follows from Claims 6 and 7.

5. Conclusion

In this work, we have presented a three-party PAKE protocol whose security does not rely on the existence of random oracles. The model that we used to prove the security of our protocol allows the adversary to ask $\mathsf{Corrupt}$ queries and thus captures insider attacks as well as forward secrecy. It is a known fact that proving the security of protocols in such a model is of particular importance in the three-party setting as insider dictionary attacks are most serious threats to three-party PAKE protocols. To the best of our knowledge, our protocol is the first three-party PAKE protocol proven secure against insider, active adversaries in the standard model (i.e., without random oracles and ideal ciphers). Another advantage our protocol has over previously published protocols is that it also achieves provable security against undetectable online dictionary attacks. The latter property is also significant as designing three-party PAKE protocol secure against undetectable online dictionary attacks is an ongoing challenge (as evidenced by the number of three-party PAKE protocols found to be vulnerable to an undetectable online dictionary attack). We leave it as a future work to design a three-party PAKE protocol that achieves not only provable security in the standard model but is more efficient than our protocol.

Conflict of Interests

The authors of the paper do not have a direct financial relation with any institution or organization mentioned in the paper that might lead to a conflict of interest for any of the authors.

Acknowledgment

This work was supported by the Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2010-0020210).

References

[1] S. M. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 72–84, Oakland, Calif, USA, May 1992.

[2] C. Herley and P. van Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Security & Privacy*, vol. 10, no. 1, pp. 28–36, 2012.

[3] J. Katz and V. Vaikuntanathan, "Round-optimal password-based authenticated key exchange," *Journal of Cryptology*, vol. 26, no. 4, pp. 714–743, 2013.

[4] W. Wang and L. Hu, "Efficient and provably secure generic construction of three-party password-based authenticated key exchange protocols," in *Progress in Cryptology—INDOCRYPT 2006*, vol. 4329 of *Lecture Notes in Computer Science*, pp. 118–132, Springer, Berlin, Germany, 2006.

[5] H. Guo, Z. Li, Y. Mu, and X. Zhang, "Cryptanalysis of simple three-party key exchange protocol," *Computers & Security*, vol. 27, no. 1-2, pp. 16–21, 2008.

[6] J. Nam, J. Paik, H.-K. Kang, U. M. Kim, and D. Won, "An off-line dictionary attack on a simple three-party key exchange protocol," *IEEE Communications Letters*, vol. 13, no. 3, pp. 205–207, 2009.

[7] C.-Y. Lin and T. Hwang, "On 'a simple three-party password-based key exchange protocol'," *International Journal of Communication Systems*, vol. 24, no. 11, pp. 1520–1532, 2011.

[8] S. Wu, Q. Pu, S. Wang, and D. He, "Cryptanalysis of a communication-efficient three-party password authenticated key exchange protocol," *Information Sciences*, vol. 215, pp. 83–96, 2012.

[9] J. Nam, K.-K. R. Choo, J. Paik, and D. Won, "An offline dictionary attack against a three-party key exchange protocol," Tech. Rep. 2013/666, Cryptology ePrint Archive, 2013.

[10] J. Nam, K.-K. R. Choo, M. Kim, J. Paik, and D. Won, "Dictionary attacks against passwordbased authenticated three-party key exchange protocols," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 12, pp. 3244–3260, 2013.

[11] K. K. R. Choo, C. Boyd, and Y. Hitchcock, "Errors in computational complexity proofs for protocols," in *Advances in Cryptology—ASIACRYPT 2005*, vol. 3788 of *Lecture Notes in Computer Science*, pp. 624–643, Springer, Berlin, Germany, 2005.

[12] K.-K. R. Choo, C. Boyd, and Y. Hitchcock, "Examining indistinguishability-based proof models for key establishment protocols," in *Advances in Cryptology—ASIACRYPT 2005*, vol. 3788 of *Lecture Notes in Computer Science*, pp. 585–604, Springer, Berlin, Germany, 2005.

[13] K.-K. R. Choo, C. Boyd, and Y. Hitchcock, "The importance of proofs of security for key establishment protocols: formal analysis of Jan-Chen, Yang-Shen-Shieh, Kim-Huh-Hwang-Lee, Lin-Sun-Hwang, and Yeh-Sun protocols," *Computer Communications*, vol. 29, no. 15, pp. 2788–2797, 2006.

[14] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Advances in Cryptology—EUROCRYPT 2000*, vol. 1807 of *Lecture Notes in Computer Science*, pp. 139–155, Springer, Berlin, Germany, 2000.

[15] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Advances in Cryptology—EUROCRYPT 2001*, vol. 2045 of *Lecture Notes in Computer Science*, pp. 453–474, 2001.

[16] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," in *Provable Security*, vol. 4784 of *Lecture Notes in Computer Science*, pp. 1–16, Springer, Berlin, Germany, 2007.

[17] C.-L. Lin, H.-M. Sun, M. Steiner, and T. Hwang, "Three-party encrypted key exchange without server public-keys," *IEEE Communications Letters*, vol. 5, no. 12, pp. 497–499, 2001.

[18] T.-F. Lee, T. Hwang, and C.-L. Lin, "Enhanced three-party encrypted key exchange without server public keys," *Computers & Security*, vol. 23, no. 7, pp. 571–577, 2004.

[19] M. Abdalla and D. Pointcheval, "Interactive Diffie-Hellman assumptions with applications to password-based authentication," in *Financial Cryptography and Data Security*, vol. 3570 of *Lecture Notes in Computer Science*, pp. 341–356, Springer, Berlin, Germany, 2005.

[20] H.-A. Wen, T.-F. Lee, and T. Hwang, "Provably secure three-party password-based authenticated key exchange protocol using Weil pairing," *IEE Proceedings-Communications*, vol. 152, no. 2, pp. 138–143, 2005.

[21] M. Abdalla, P. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," *IEE Proceedings Information Security*, vol. 153, no. 1, pp. 27–39, 2006.

[22] R. Lu and Z. Cao, "Simple three-party key exchange protocol," *Computers & Security*, vol. 26, no. 1, pp. 94–97, 2007.

[23] H.-R. Chung and W.-C. Ku, "Three weaknesses in a simple three-party key exchange protocol," *Information Sciences*, vol. 178, no. 1, pp. 220–229, 2008.

[24] H.-S. Kim and J.-Y. Choi, "Enhanced password-based simple three-party key exchange protocol," *Computers and Electrical Engineering*, vol. 35, no. 1, pp. 107–114, 2009.

[25] H.-F. Huang, "A simple three-party password-based key exchange protocol," *International Journal of Communication Systems*, vol. 22, no. 7, pp. 857–862, 2009.

[26] E. Dongna, Q. Cheng, and C. Ma, "Password authenticated key exchange based on RSA in the three-party settings," in *Provable Security*, vol. 5848 of *Lecture Notes in Computer Science*, pp. 168–182, Springer, Berlin, Germany, 2009.

[27] T.-F. Lee and T. Hwang, "Simple password-based three-party authenticated key exchange without server public keys," *Information Sciences*, vol. 180, no. 9, pp. 1702–1714, 2010.

[28] W. Wang, L. Hu, and Y. Li, "How to construct secure and efficient three-party password-based authenticated key exchange protocols," in *Information Security and Cryptology*, vol. 6584 of *Lecture Notes in Computer Science*, pp. 218–235, Springer, Berlin, Germany, 2010.

[29] T.-Y. Chang, M.-S. Hwang, and W.-P. Yang, "A communication-efficient three-party password authenticated key exchange protocol," *Information Sciences*, vol. 181, no. 1, pp. 217–226, 2011.

[30] J. Nam, Y. Lee, S. Kim, and D. Won, "Security weakness in a three-party pairing-based protocol for password authenticated key exchange," *Information Sciences*, vol. 177, no. 6, pp. 1364–1375, 2007.

[31] R. C.-W. Phan, W.-C. Yau, and B.-M. Goi, "Cryptanalysis of simple three-party key exchange protocol (S-3PAKE)," *Information Sciences*, vol. 178, no. 13, pp. 2849–2856, 2008.

[32] E.-J. Yoon and K.-Y. Yoo, "Cryptanalysis of a simple three-party password-based key exchange protocol," *International Journal of Communication Systems*, vol. 24, no. 4, pp. 532–542, 2011.

[33] H. Liang, J. Hu, and S. Wu, "Re-attack on a three-party password-based authenticated key exchange protocol," *Mathematical and Computer Modelling*, vol. 57, no. 5-6, pp. 1175–1183, 2013.

[34] H.-T. Tsai and C.-C. Chang, "Provably secure three party encrypted key exchange scheme with explicit authentication," *Information Sciences*, vol. 238, pp. 242–249, 2013.

[35] J. Nam, K.-K. R. Choo, J. Paik, and D. Won, "On the security of a password-only authenticated three-party key exchange protocol," Tech. Rep. 2013/540, Cryptology ePrint Archive, 2013.

[36] J. Nam, K.-K. R. Choo, J. Paik, and D. Won, "Two-round password-only authenticated key exchange in the three-party setting," Cryptology ePrint Archive 2014/017, 2014.

[37] J.-H. Yang and T.-J. Cao, "Provably secure three-party password authenticated key exchange protocol in the standard model," *Journal of Systems and Software*, vol. 85, no. 2, pp. 340–350, 2012.

[38] K. Yoneyama, "Efficient and strongly secure password-based server aided key exchange," in *Progress in Cryptology—INDOCRYPT 2008*, vol. 5365 of *Lecture Notes in Computer Science*, pp. 172–184, Springer, Berlin, Germany, 2008.

[39] H.-Y. Chien and T.-C. Wu, "Provably secure password-based three-party key exchange with optimal message steps," *The Computer Journal*, vol. 52, no. 6, pp. 646–655, 2009.

[40] N. W. Lo and K.-H. Yeh, "Cryptanalysis of two three-party encrypted key exchange protocols," *Computer Standards & Interfaces*, vol. 31, no. 6, pp. 1167–1174, 2009.

[41] D.-C. Lou and H.-F. Huang, "Efficient three-party password-based key exchange scheme," *International Journal of Communication Systems*, vol. 24, no. 4, pp. 504–512, 2011.

[42] C. Lee, S. Chen, and C. Chen, "A computation-efficient three-party encrypted key exchange protocol," *Applied Mathematics & Information Sciences*, vol. 6, no. 3, pp. 573–579, 2012.

[43] J. Zhao and D. Gu, "Provably secure three-party password-based authenticated key exchange protocol," *Information Sciences*, vol. 184, no. 1, pp. 310–323, 2012.

[44] S. Wu, K. Chen, Q. Pu, and Y. Zhu, "Cryptanalysis and enhancements of efficient three-party password-based key exchange scheme," *International Journal of Communication Systems*, vol. 26, no. 5, pp. 674–686, 2013.

[45] K.-K. R. Choo, "A proof of revised Yahalom protocol in the Bellare and Rogaway (1993) model," *The Computer Journal*, vol. 50, no. 5, pp. 591–601, 2007.

[46] W. Diffie, P. C. van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[47] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, Springer, Berlin, Germany, 2003.

[48] B. S. Kaliski, "An unknown key-share attack on the MQV key agreement protocol," *ACM Transactions on Information and System Security*, vol. 4, no. 3, pp. 275–288, 2001.

[49] J. Katz, R. Ostrovsky, and M. Yung, "Efficient and secure authenticated key exchange using weak passwords," *Journal of the ACM*, vol. 57, no. 1, article 3, 2009.