A Strengthened PAKE Protocol with Identity-Based Encryption*  

SeongHan SHIN†, Member

1. Introduction

Password-based authenticated key exchange protocols provide password authentication and establishment of session keys to be used for protecting subsequent communications. Because password authentication is commonly used in the real world, these protocols have been widely deployed in many applications (e.g., TLS, SSH, IPsec, WPA, HTTP). Since the appearance of EKE [3], [4] (known as PAKE), many applications (e.g., TLS, SSH, IPsec, WPA, HTTP) in the real world, these protocols have been widely deployed in

Because password authentication is commonly used in the real world, these protocols have been widely deployed in

Since the appearance of EKE [3], [4] (known as PAKE), many applications (e.g., TLS, SSH, IPsec, WPA, HTTP). Because password authentication is commonly used in the real world, these protocols have been widely deployed in

1. Motivation and Our Contributions

In [2], Choi et al. proposed the iPAKE protocol and its generic construction where the latter construction named ‘Unbalanced Key Agreement Mechanism with Password and Identity-based Encryption (UKAM-PiE)’ was standardized in ISO/IEC 11770-4/AMD 1 [21]. So, it is of significant importance to thoroughly analyze the security of these protocols.

In this paper, we first revisit the iPAKE protocol [2] using the Boneh-Franklin IBE scheme [15], [18], and the UKAM-PiE protocol in ISO/IEC 11770-4/AMD 1 [21]. After describing the iPAKE and UKAM-PiE protocols, we show that they are insecure against passive/active attacks by a malicious PKG (Private Key Generator) where the malicious PKG can find out all clients’ passwords by just eavesdropping on the communications, and the PKG can share a session key with any client by impersonating the server. Then, we propose a strengthened PAKE (for short, SPAIBE) protocol with IBE, which prevents such a malicious PKG’s passive/active attacks. Also, we formally prove the security of the SPAIBE protocol in the random oracle model and compare relevant PAKE protocols in terms of efficiency, number of passes, and security against a malicious PKG.

SUMMARY In [2], Choi et al. proposed an identity-based password-authenticated key exchange (iPAKE) protocol using the Boneh-Franklin IBE scheme, and its generic construction (UKAM-PiE) that was standardized in ISO/IEC 11770-4/AMD 1. In this paper, we show that the iPAKE and UKAM-PiE protocols are insecure against passive/active attacks by a malicious PKG (Private Key Generator) where the malicious PKG can find out all clients’ passwords by just eavesdropping on the communications, and the PKG can share a session key with any client by impersonating the server. Then, we propose a strengthened PAKE (for short, SPAIBE) protocol with IBE, which prevents such a malicious PKG’s passive/active attacks. Also, we formally prove the security of the SPAIBE protocol in the random oracle model and compare relevant PAKE protocols in terms of efficiency, number of passes, and security against a malicious PKG.

key words: PAKE, IBE, PKG, Online/Offline dictionary attacks, provable security

Copyright © 2022 The Institute of Electronics, Information and Communication Engineers

DOI: 10.1587/transinf.2022NGP0009
2. Preliminaries

2.1 Notation

Let $k \in \mathbb{N}$ and $\lambda \in \mathbb{N}$ be the security parameters. Let $\{0, 1\}^*$ be the set of finite binary strings and $\{0, 1\}^k$ be the set of binary strings of length $k$. Let $A \| B$ be the concatenation of $A$ and $B$. If $U$ is a set, then $u \in U$ indicates the process of selecting $u$ at random and uniformly over $U$. If $U$ is a function (whatever it is), then $u = U$ indicates the process of assigning the result to $u$. Let $\mathbb{D}_{pw}$ be a dictionary size of passwords whose cardinality is $N$. Let $\mathcal{C}$ and $\mathcal{S}$ be the identities of client and server, respectively, with each $\text{ID} \in \{0, 1\}^*$. 

Also, let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two (multiplicative) cyclic groups of order $q$ for some large prime $q$. An admissible (symmetric) bilinear map $\epsilon : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ has the following properties: 1) Bilinear: For all $g_1, g_2 \in \mathcal{G}_1$ and all $\alpha, \beta \in \mathbb{Z}_q^*$, $\epsilon(g_1^\alpha, g_2^\beta) = \epsilon(g_1, g_2)^{\alpha \beta}$; 2) Non-degenerate: For all pairs $g_1, g_2 \in \mathcal{G}_1$, $\epsilon(g_1, g_2) \neq 1$. If $g$ is a generator of $\mathcal{G}_1$, then $\epsilon(g, g)$ is a generator of $\mathcal{G}_2$; and 3) Computable: There is an efficient algorithm to compute $\epsilon(g_1, g_2)$ for any $g_1, g_2 \in \mathcal{G}_1$. Note that the modified Weil and Tate pairings on elliptic curves are examples of the admissible bilinear map [18].

2.2 Computational Assumptions

First, we define the Computational Diffie-Hellman (CDH) problem. Let $\mathcal{G}_1$ be the group generation algorithm that takes as input $1^\lambda$ and outputs a group description $(\mathcal{G}_1, q, g)$ where $\mathcal{G}_1$ is a finite cyclic group of prime order $q$ with $g$ as a generator.

**Definition 1:** (CDH Problem) Let $\mathcal{G}_1$ be the group generation algorithm described above. A $(t_1, \varepsilon_1)$-CDH$_{\mathcal{G}_1}$ adversary is a probabilistic polynomial time (PPT) machine $B$, running in time $t_1$, such that its success probability $\text{Succ}_{\text{CDH}(B)}$ given random elements $g^\alpha$ and $g^\beta$ to output $g^{\alpha \beta}$, is greater than $\varepsilon_1$. We denote by $\text{Succ}_{\text{CDH}(B)}(t_1)$ the maximal success probability over all adversaries, running within time $t_1$. The CDH problem states that $\text{Succ}_{\text{CDH}(B)}(t_1) \leq \varepsilon_1$ for any $t_1/\varepsilon_1$ not too large.

Next, we define the Bilinear Diffie-Hellman (BDH) problem. Let $\mathcal{G}_2$ be the BDH group generation algorithm that takes as input $1^\lambda$ and outputs a group description $(\mathcal{G}_1, \mathcal{G}_2, e, q, g)$ where $\mathcal{G}_1$ and $\mathcal{G}_2$ are two groups of prime order $q$, $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ is an admissible bilinear map and $g$ is a generator of $\mathcal{G}_1$.

**Definition 2:** (BDH Problem) Let $\mathcal{G}_2$ be the BDH group generation algorithm described above. A $(t_2, \varepsilon_2)$-BDH$_{\mathcal{G}_1, \mathcal{G}_2, e}$ adversary is a probabilistic polynomial time (PPT) machine $B$, running in time $t_2$, such that its success probability $\text{Succ}^{\text{BDH}}_{\mathcal{G}_1, \mathcal{G}_2, e}(B)$, given random elements $g^\alpha$, $g^\beta$ and $g^\gamma$ to output $\epsilon(g, g)^{\alpha \beta \gamma}$, is greater than $\varepsilon_2$. We denote by $\text{Succ}^{\text{BDH}}_{\mathcal{G}_1, \mathcal{G}_2, e}(t_2)$ the maximal success probability over every adversaries, running within time $t_2$. The BDH problem states that $\text{Succ}^{\text{BDH}}_{\mathcal{G}_1, \mathcal{G}_2, e}(t_2) \leq \varepsilon_2$ for any $t_2/\varepsilon_2$ not too large.

2.3 An Identity-Based Encryption Scheme

In this subsection, we define the syntax of identity-based encryption (IBE) and describe the Boneh-Franklin IBE (BF-IBE) scheme [15], [18] that is the most efficient construction among IBE schemes (see [19]).

**Definition 3:** (Identity-Based Encryption) An identity-based encryption (IBE) scheme is a quadruple of probabilistic polynomial time algorithms $(\text{Setup}_{\text{IBE}}, \text{Extract}, \text{Encrypt}, \text{Decrypt})$ such that:

- **Setup$_{\text{IBE}}$:** The setup algorithm takes as input $1^\lambda$ and outputs public parameters $pp_{\text{IBE}}$ and a master secret key $\text{msk}$ where (mpk, msk) is a pair of master public/secret keys and mpk is included in $pp_{\text{IBE}}$. Also, the public parameters include descriptions of a finite message space $\mathcal{M}$ and a finite ciphertext space $C$. The $pp_{\text{IBE}}$ will be publicly known, while the msk will be known only to the Private Key Generator (PKG).
- **Extract:** The key extraction algorithm takes as input $pp_{\text{IBE}}, \text{msk}$ and an identity ID $\in \{0, 1\}^*$, and outputs a private key $d_{\text{ID}}$ corresponding to the user with this identity.
- **Encrypt:** The encryption algorithm takes as input $pp_{\text{IBE}}, \text{ID}$ and a message $M \in \mathcal{M}$, and outputs a ciphertext $C \in \mathcal{C}$.
- **Decrypt:** The decryption algorithm takes as input $pp_{\text{IBE}}, C$ and a private key $d_{\text{ID}}$, and outputs $M \in \mathcal{M}$.

It is required that $M = \text{Decrypt}(pp_{\text{IBE}}, C, d_{\text{ID}})$ for all $\text{ID} \in \{0, 1\}^*$ and $M \in \mathcal{M}$ where $C = \text{Encrypt}(pp_{\text{IBE}}, \text{ID}, M)$.

Also, we define the semantic security (i.e., semantic security against adaptively chosen identity and message attacks) for an IBE scheme.

**Definition 4:** (Semantic Security of IBE) We say that an IBE scheme $IEB$ is semantically secure (IND-ID-CPA secure) if, for any PPT adversary $B$, an advantage $\text{Adv}_{\text{IBE}^{\text{ind-id-CPA}}} = 2 \mathbb{P}[b = b'] - 1$ of the adversary $B$ against the scheme $IEB$ is negligible in the following game:

- **Setup$_{\text{Challenger}}$:** A challenger runs the setup algorithm Setup$_{\text{IBE}}$ and gives the adversary public parameters $pp_{\text{IBE}}$. The master secret key msk is kept by the challenger.
- **Phase 1:** The adversary issues key extraction queries $\text{ID}_1, \ldots, \text{ID}_m$. The challenger responds with a private key $d_{\text{ID}}$ corresponding to $\text{ID}_i$ by running the key extraction algorithm Extract. These queries may be asked adaptively.
- **Challenge:** When Phase 1 is over, the adversary outputs two equal length messages $M_0, M_1 \in \mathcal{M}$ and an identity $\text{ID}$. The only constraint is that $\text{ID}$ did not
appear in any key extraction query in Phase 1. The challenger chooses a random bit $b \in \{0, 1\}$ and sends $C = \text{Encrypt}(pp_{\text{IBE}}, \text{ID}, M_b)$ as a challenge to the adversary.

- **Phase 2**: The adversary issues more key extraction queries $\text{ID}_{m+1}, \ldots, \text{ID}_n$. The only constraint is that $\text{ID}_i \neq \text{ID}$. The challenger responds as in Phase 1.
- **Guess**: Finally, the adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

### 2.3.1 Boneh-Franklin IBE (BF-IBE)

Here, we describe the Boneh-Franklin IBE (BF-IBE) scheme [15], [18] that is proven to be IND-CD-CPA secure (i.e. $\text{Adv}_{\text{BF-IBE}}^{\text{IND-CD-CPA}}(\mathcal{B})$ is negligible) in the random oracle model [22] under the BDH problem in Definition 2.

- **Setup_{\text{IBE}}**: The setup algorithm on input $1^\lambda$ outputs public parameters $pp_{\text{IBE}}$ and a master secret key $\text{msk}$ where $(G_1, G_2, e, q, g)$ is generated by calling the BDH group generation algorithm $G_2$ on input $1^\lambda$, and $G : \{0, 1\}^\ast \rightarrow G_1$ and $H : G_2 \rightarrow \{0, 1\}^k$ are descriptions of cryptographic hash functions. The message space $M = \{0, 1\}^k$ and the ciphertext space $C = G_1 \times \{0, 1\}^k$. Let $z$ be a random element from $\mathbb{Z}_q^\ast$ and set $(\text{mpk}, \text{msk}) = (g^z, z)$. It outputs $(pp_{\text{IBE}}, \text{msk}) = ((G_1, G_2, e, q, g, g^z, G, H), z)$.
- **Extract**: The key extraction algorithm, on input $pp_{\text{IBE}}$, $\text{msk}(= z)$ and an identity $\text{ID}$, computes $Q_{\text{ID}} = G(\text{ID})$ and outputs a private key $d_{\text{ID}} \equiv Q_{\text{ID}}^z$.
- **Encrypt**: The encryption algorithm, on input $pp_{\text{IBE}}$, an identity $\text{ID}$ and a message $M$, chooses a random element $r \leftarrow \mathbb{Z}_q^\ast$ and computes $g_{\text{ID}} = e(G(\text{ID}), g^r)$, $U_1 \equiv g^r$ and $U_2 = M \cdot H(g_{\text{ID}}^r)$. Then, it outputs a ciphertext $C = (U_1, U_2)$.
- **Decrypt**: The decryption algorithm, on input $pp_{\text{IBE}}$, $C = (U_1, U_2)$ and a private key $d_{\text{ID}}(\equiv Q_{\text{ID}}^z)$, computes $\delta = e(d_{\text{ID}}, U_1)$ and outputs $M = U_2 \cdot H(\delta)$.

In the above, the consistency of the BF-IBE scheme can be easily checked by

$$\delta = e(d_{\text{ID}}, U_1) = e(Q_{\text{ID}}^z, g^r) = e(Q_{\text{ID}}, g)^r = e(Q_{\text{ID}}, g^z)^r = g_{\text{ID}}^z. \quad (1)$$

### 3. The iPAKE Protocol

In this section, we describe the iPAKE protocol [2] which consists of **Initialization** and **Key Establishment** phases.

#### 3.1 Initialization

In this phase, it executes the following three processes **Setup**, **Extract** and **Registration**.

- **Setup**

The **Setup** on input $1^\lambda$ outputs public parameters $pp$ and a master secret key $\text{msk}$ by running the setup algorithm **Setup_{\text{IBE}}** of Sect. 2.3.1 where $H', H_1 : \{0, 1\}^\ast \rightarrow \{0, 1\}^k$ are descriptions of additional cryptographic hash functions. It outputs $(pp, \text{msk}) = ((pp_{\text{IBE}}, H', H_1), z)$.

#### 3.1.2 Extract

The key extraction algorithm **Extract** (run by PKG), on input the public parameters $pp_{\text{IBE}}$, the master secret key $\text{msk}(= z)$ and an identity $S$, computes $Q_S = G(S)$ and outputs a private key $d_S \equiv Q_S^z$ that is securely transmitted to the corresponding server $S$.

- **Registration**

First, client $C$ randomly chooses his/her password $pw$ from a dictionary $\mathcal{D}_{pw}$ and sends $(C, H'(pw))$ to server $S$. Then, the server stores $(C, H'(pw))$ to a password file. This registration process should be done securely between client $C$ and server $S$.

#### 3.2 Key Establishment

In this phase, client $C$ and server $S$ execute the iPAKE protocol over insecure networks in order to share a session key. This phase of iPAKE has three steps as below.

- **Step 1**: The client $C$ runs the encryption algorithm **Encrypt** of Sect. 2.3.1 on input $pp_{\text{IBE}}$, an identity $S$ and a message $H'(pw)$. Then, client $C$ sends $(C, U_1, U_2)$ to server $S$.

- **Step 2**: After receiving a message $(C, U_1, U_2)$ from client $C$, server $S$ runs the decryption algorithm **Decrypt** of Sect. 2.3.1 on input $pp_{\text{IBE}}$, $(U_1, U_2)$ and a private key $d_S$. If $(U_2 \equiv H(\delta)) \neq H'(pw)$, the server aborts the protocol. Otherwise, server $S$ chooses a random element $y \leftarrow \mathbb{Z}_q^\ast$, and computes $Y \equiv g^y$ and $Z \equiv U_2^y$. Also, the server computes a session key $S K_S = H_3(C||S||sid||\text{id}||Z)$, where $sid = U_1||U_2||Y$, and then sends $(S, Y)$ to the client.

- **Step 3**: After receiving a message $(S, Y)$ from server $S$, client $C$ computes $Z \equiv Y'$ and a session key $S K_C = H_3(C||S||sid||g_{S}^z||Z)$ where $sid = U_1||U_2||Y$.

**Remark 1**: In [2], Choi et al. proposed the iPAKE protocol and its generic construction using an identity-based KEM/DEM scheme where the latter construction named as ‘Unbalanced Key Agreement Mechanism with Password and Identity-based Encryption (UKAM-PiE)’ was standardized in ISO/IEC 11770-4/AMD 1 [21]. The UKAM-PiE protocol, instantiated with the BF-IBE scheme [15], [18], is somewhat different from the iPAKE protocol as follows:

- **Step 1’**: First, client $C$ chooses a random element $x \leftarrow \mathbb{Z}_q^\ast$.
SHIN: A STRENGTHENED PAKE PROTOCOL WITH IDENTITY-BASED ENCRYPTION

and computes $X \equiv g^X$. Next, the client runs the encryption algorithm $Encrypt$ of Sect. 2.3.1 on input $pp_{IBE}$, an identity $S$ and a message $H'(pw)||X$. Then, client $C$ sends $(C, U_1, U_2)$ to server $S$.

Step 2*. After receiving a message $(C, U_1, U_2)$ from client $C$, server $S$ runs the decryption algorithm $Decrypt$ of Sect. 2.3.1 on input $pp_{IBE}$, $(U_1, U_2)$ and a private key $d_S$. If the first component of $(U_2 \oplus H(\delta))$ is not equal to $H'(pw)$, the server aborts the protocol. Otherwise, server $S$ chooses a random element $y \leftarrow Z_q^*$, and computes $Y \equiv g^y$ and $Z \equiv X^y$ where $X$ is the second component of $(U_2 \oplus H(\delta))$. Also, the server computes a session key $S K_S = H_3(C||S||X||Y||Z)$ and then sends $(S, Y)$ to the client.

Step 3*. After receiving a message $(S, Y)$ from server $S$, client $C$ computes $Z \equiv Y^r$ and a session key $S K_C = H_3(C||S||X||Y||Z)$.

4. Passive/Active Attacks on iPAKE and UKAM-PiE

This section shows passive and active attacks by a malicious PKG (Private Key Generator) on the iPAKE [2] and UKAM-PiE [21] protocols.

4.1 A Passive Attack on iPAKE

Here, we show that a malicious PKG can find out all clients’ passwords by just eavesdropping on the communications of the iPAKE protocol. After eavesdropping on the first message $(C, U_1, U_2)$ in the Key Establishment phase, the malicious PKG who has the master secret key $z$ can decrypt the ciphertext $(U_1, U_2)$ since $d_S \equiv (G(S))^\gamma$. With all possible password candidates, the PKG can find out the client’s password $pw$ by performing offline dictionary attacks on $H'(pw) = U_2 \oplus H(e(d_S, U_1))$. Of course, these offline dictionary attacks can be used for all clients who registered to server $S$.

4.2 An Active Attack on iPAKE

Here, we show that a malicious PKG can share a session key with any client by impersonating the server in the iPAKE protocol. After receiving the first message $(C, U_1, U_2)$ in the Key Establishment phase, the malicious PKG who has the master secret key $z$ just executes Step 2 except the check of $W \neq H'(pw)$ and then can share the same session key $S K_C = H_3(C||S||sid||b||Z)$ with client $C$. Note that in this active attack the PKG does not need to perform offline dictionary attacks at all.

4.3 Passive/Active Attacks on UKAM-PiE

As in the same way of Sects. 4.1 and 4.2, a malicious PKG can perform passive and active attacks on the UKAM-PiE protocol [21].

5. A Strengthened PAKE Protocol with IBE

In this section, we propose a strengthened PAKE (for short, SPAIBE) protocol with IBE which provides security against passive/active attacks by a malicious PKG (Private Key Generator). A main idea of the SPAIBE protocol is 1) to double mask a Diffie-Hellman public key on client $C$ where the first mask is performed with the password verification data and the second one is with the encryption algorithm $Encrypt$ of BF-IBE [15], [18], and 2) to make server $S$ to send its authenticator. Note that security against a malicious PKG’s passive/active attacks is a stronger security guarantee than security against an outside adversary’s passive/active attacks. The SPAIBE protocol consists of Initialization and Key Establishment phases.

5.1 Initialization

In this phase, it executes the following three processes Setup, Extract and Registration.

5.1.1 Setup

The Setup on input $1^k$ outputs public parameters $pp$ and a master secret key $msk$ by running the setup algorithm $Setup_{IBE}$ of Sect. 2.3.1 where $h$ is another random generator of $G_1$, and $H_1 : [0, 1]^* \rightarrow Z_q^*$ and $H_2, H_3 : [0, 1]^* \rightarrow [0, 1]^k$ are descriptions of additional cryptographic hash functions. It outputs $(pp, msk) = ((pp_{IBE}, h, H_1, H_2, H_3, z))$.

5.1.2 Extract

The key extraction algorithm $Extract$ (run by PKG), on input the public parameters $pp_{IBE}$, the master secret key $msk = z$ and an identity $S$, computes $Q_S = G(S)$ and outputs a private key $d_S \equiv Q_S^z$ that is securely transmitted to the corresponding server $S$.

5.1.3 Registration

First, client $C$ randomly chooses his/her password $pw$ from a dictionary $D_{pw}$ and sends $(C, h^{-H_1(pw)})$ to server $S$. Then, the server stores $(C, h^{-H_1(pw)})$ to a password file. Note that password $pw$ is kept by client $C$ secretly, and $(d_S, (C, h^{-H_1(pw)}))$ are held by server $S$ secretly. This registration process should be done securely between client $C$ and server $S$.

5.2 Key Establishment

In this phase, client $C$ and server $S$ execute the SPAIBE protocol over insecure networks (e.g., the Internet) in order to share a session key to be used for protecting subsequent communications. This phase of SPAIBE has three steps as below (see also Fig. 1).

Step 1. First, client $C$ chooses two random elements $x, r \leftarrow S$.
and computes a Diffie-Hellman public value $X \equiv g^x$ and its masked value $W \equiv X \cdot h^{H(pk)}$ to the password verification data $h^{H(pk)}$. For the second mask, the client encrypts the value $W$ by computing $g_S = e(G(S), g^x)$. If an adversary gets the password $pw$, it can find out the password by performing offline dictionary attacks. This is the reason why we use two random elements $x, r$ where $x$ is used for the first mask and $r$ is for the second mask in the SPAIBE protocol.

6. Security Model

Here, we extend the security model [27], [28] in order to capture attacks of a malicious PKG, and define the semantic security of session keys.

Let $C$ and $S$ be sets of clients and servers, respectively. We denote by $C \in C$ and $S \in S$ two parties that participate in an authenticated key exchange protocol $P$. Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of $P$. We denote $C$ (resp., $S$) instances by $C^i$ (resp., $S^j$) where $i, j \in \mathbb{N}$, or by $I$ in the case of any instance. During the protocol execution, an adversary has the entire control of networks and has access to the master secret key. Let us show the capability of adversary $A$ each query captures:

- **Execute($C^i$, $S^j$)**: This query models passive attacks, where the adversary gets access to honest executions of $P$ between the instances $C^i$ and $S^j$ by eavesdropping.

- **Send($I$, msg)**: This query models active attacks by

---

---

1. From an implementation perspective, data type conversion functions for $W$ can be used. See Annex of [8].

**Fig. 1** A strengthened PAKE (for short, SPAIBE) protocol with IBE
having $\mathcal{A}$ send a message to instance $I$. The adversary $\mathcal{A}$ gets back the response $I$ generates in processing the message $msg$ according to the protocol $P$. A query $\text{Send}(C', \text{Start})$ initializes the protocol, and thus the adversary receives the first flow message.

- **Reveal($I$)**: This query handles misuse of the session key (e.g., use in a weak symmetric-key encryption) by any instance $I$. The query is only available to $\mathcal{A}$, if the instance actually holds a session key, and the latter is released to $\mathcal{A}$.
- **Reveal($PKG$)**: This query handles compromise of the PKG, and the adversary receives the master secret key.
- **Test($I$)**: This oracle is used to see whether or not the adversary $\mathcal{A}$ can obtain some information on the session key by giving a hint on the key. The Test-query can be asked at most once by the adversary $\mathcal{A}$ if the instance $I$ is “fresh” (see below). This query is answered as follows: One flips a (private) coin $b \in \{0, 1\}$ and forwards the corresponding session key $SK_b$ ($\text{Reveal}(I)$ would output) if $b = 1$, or a random value with the same size except the session key if $b = 0$.

We say that an instance $I$ is fresh only if an adversary $\mathcal{A}$ cannot distinguish the session key $SK_b$ in a trivial way (e.g., with the $\text{Reveal}(I)$-query).

The adversary $\mathcal{A}$ is provided with random coin tosses, some oracles and then is allowed to invoke any number of queries as described above, in any order. The aim of the adversary is to break the privacy of the session key (a.k.a., semantic security) in the context of executing $P$.

Definition 5: (AKE Security) The AKE security is defined by the game $\text{Game}^{\text{ake}}(\mathcal{A}, P)$, in which the ultimate goal of the adversary is to guess the bit $b$ involved in the Test-query by outputting this guess $b'$. We denote the AKE advantage, by $\text{Adv}_{P}^{\text{ake}}(\mathcal{A}) = 2 \Pr[b = b'] - 1$, as the probability that $\mathcal{A}$ can correctly guess the value of $b$. The protocol $P$ is said to be $(t,\epsilon)$-AKE-secure if $\mathcal{A}$’s advantage is smaller than $\epsilon$ for any adversary $\mathcal{A}$ running time $t$.

7. Security Proof of SPAIBE

In this section, we show that the SPAIBE protocol of Fig. 1 is provably secure in the random oracle model [22].

Theorem 1: Let $P$ be the SPAIBE protocol of Fig. 1 where passwords are chosen from a dictionary of size $N$. For any adversary $\mathcal{A}$ within a polynomial time $t$, less than $q_{se}$ active interactions with the parties (Send-queries), $q_{ex}$ passive eavesdroppings (Execute-queries) and asking $q_{ht}$ hash queries to any $H_j$, $\text{Adv}_{P}^{\text{ake}}(\mathcal{A}) \leq \epsilon$, with $\epsilon$ upper-bounded by

$$\frac{6q_{se}^2}{N} + 6q_{ht}^2 \times \text{Succ}^{\text{gen}}_{C_1} + 3(3q_{ex} + q_{se})^2 \times q + \frac{2q_{se}}{2^k} + 4nq_{se} \times \text{Adv}_{P}^{\text{ind-id-cpa}}(B) ,$$

where $n = |S|$, $k$ is the output length of $H$, and $\tau_e$ denotes the computational time for an exponentiation in $G_1$.

This theorem indicates that the SPAIBE protocol is secure against offline dictionary attacks since the advantage of the adversary essentially grows with the ratio of interactions to the number of passwords. Also, one can notice that this security result holds with any IBE scheme that is IND-ID-CPA secure.

Proof 1: In this proof, we define a sequence of games starting at the real protocol $G_0$ and ending up at $G_6$ where we bound the probability of each event by using Shoup’s difference lemma [29]. For clarity, we denote by Event, an event Event considered in Game $G_i$.

Game $G_0$: This is the real protocol in the random oracle model. We are interested in the following event:

- $S_0$ which occurs if the adversary correctly guesses the bit $b$ involved in the Test-query

$$\text{Adv}^{\text{ake}}_P(\mathcal{A}) = 2 \Pr[S_0] - 1 .$$

Game $G_1$: In this game, we simulate the hash oracles ($H_j$, but as well additional hash functions, for $j = 1, 2, 3$ which will appear in the game $G_4$) by maintaining hash lists $\Lambda_H$ and $\Lambda_{ht}$ (see below). We also simulate all the instances, as the real parties would do, for the Send-queries and for the Execute, Reveal, and Test-queries (see further below). From this simulation, we can easily see that the game is perfectly indistinguishable from the real attack.

Simulation of the hash functions: $H_j$ oracles

- For a hash-query $H_j(q)$ (resp., $H'_j(q)$), such that a record $(j, q, r)$ appears in $\Lambda_H$ (resp., $\Lambda_{ht}$), the answer is $r$. Otherwise, one chooses a random element $r \overset{\$}{\leftarrow} \{0, 1\}^k$, answers with it, and adds the record $(j, q, r)$ to $\Lambda_H$ (resp., $\Lambda_{ht}$).

Simulation of the SPAIBE protocol

Send-queries to $C$

We answer to the Send-queries to a C-instance as follows:

- A $\text{Send}(C', \text{Start})$-query is processed according to the following rules:

  **Rule C1**(1)

  Choose a random element $\theta \overset{\$}{\leftarrow} \mathbb{Z}_q^*$, and compute $X \equiv g_\theta^w$, $w = H_1(pw)$, $W \equiv X \cdot h^w$ and $(U_1, U_2) = \text{Encrypt}(s, W)$. Then the query is answered with $(C, U_1, U_2)$, and the instance goes to an expecting state.

  If the instance $C'$ is in an expecting state, a query $\text{Send}(C', (S, Y, V_S))$ is processed by computing the Diffie-Hellman key, authenticator and session key. We apply the following rules.

  **Rule C2**(1)
Game $G_3$: Here, we consider the following

$$\Pr[S_3] = \Pr[S_3]|_{\text{BreakIBE} \lor \text{BreakIBE}} = \Pr[S_3]|_{\text{BreakIBE}} + \Pr[S_3]|_{\text{BreakIBE}}$$

where $\text{BreakIBE}$ is an event that an adversary gets the master secret key by asking the $\text{Reveal}(PKG)$-query, and its complement is denoted by $\text{¬BreakIBE}$. For the first term, we construct an adversary $B$ who breaks the IND-ID-CPA security of the IBE scheme $IBE$ by using an adversary $A$ who breaks the semantic security of the SPAIBE protocol.

Let $n = |S|$ and $S$ be the challenge identity. $B$ chooses two random elements $\theta_0, \theta_1 \xleftarrow{} \mathbb{Z}_q^*$, computes $W_0 \equiv g^{\theta_0} \cdot h^w$ and $W_1 \equiv g^{\theta_1} \cdot h^w$, and forwards $(W_0, W_1, S)$ to the IBE challenger. The challenge $\text{Encrypt}(pp_{IBE}, S, W_j)$ is introduced in the simulation of the $\text{Send}(C', \text{Start})$-query. Finally, the output of $B$ is the same as the output of $A$, and the probability of $\Pr[S_3]|_{\text{BreakIBE}}$ is bounded by

$$\Pr[S_3]|_{\text{BreakIBE}} \leq 2nq_{ex} \times \text{Adv}_{\text{IBE}}^{\text{ind-id-CPA}}(B).$$

Note that all the bounds in the subsequent games are conditioned on the BreakIBE.

Game $G_4$: In order to make the authenticator and session key unpredictable to any adversary, we compute them using the private oracles $H_j'$ (instead of $H_j$) so that the values are completely independent from the random oracles. We reach this aim by using the following rules:

$\text{Rule C3}^{(4)}$

Compute the authenticator and session key:

$$V_S = H_2(C||S||U_1||U_2||Y||X||K),$$

$$S K_C = H_3(C||S||U_1||U_2||Y||X||K).$$

If $V_S \neq V_S$, it terminates. Otherwise, the instance accepts.

The probability is bounded by the birthday paradox:

$$\Pr[\text{Coll}_2] \leq \frac{(q_{ex} + q_{sa})^2}{2q}.$$  \hfill (4)

**Game $G_3$:** Here, we consider the following

$$\Pr[S_3] = \Pr[S_3]|_{\text{BreakIBE} \lor \text{BreakIBE}} = \Pr[S_3]|_{\text{BreakIBE}} + \Pr[S_3]|_{\text{BreakIBE}}$$

where $\text{BreakIBE}$ is an event that an adversary gets the master secret key by asking the $\text{Reveal}(PKG)$-query, and its complement is denoted by $\text{¬BreakIBE}$. For the first term, we construct an adversary $B$ who breaks the IND-ID-CPA security of the IBE scheme $IBE$ by using an adversary $A$ who breaks the semantic security of the SPAIBE protocol.

Let $n = |S|$ and $S$ be the challenge identity. $B$ chooses two random elements $\theta_0, \theta_1 \xleftarrow{} \mathbb{Z}_q^*$, computes $W_0 \equiv g^{\theta_0} \cdot h^w$ and $W_1 \equiv g^{\theta_1} \cdot h^w$, and forwards $(W_0, W_1, S)$ to the IBE challenger. The challenge $\text{Encrypt}(pp_{IBE}, S, W_j)$ is introduced in the simulation of the $\text{Send}(C', \text{Start})$-query. Finally, the output of $B$ is the same as the output of $A$, and the probability of $\Pr[S_3]|_{\text{BreakIBE}}$ is bounded by

$$\Pr[S_3]|_{\text{BreakIBE}} \leq 2nq_{ex} \times \text{Adv}_{\text{IBE}}^{\text{ind-id-CPA}}(B).$$

Note that all the bounds in the subsequent games are conditioned on the BreakIBE.
SHIN: A STRENGTHENED PAKE PROTOCOL WITH IDENTITY-BASED ENCRYPTION

Game $G_3$: It is now possible to evaluate the probability of the event $\text{AskH}$ (or more precisely, the sub-cases). Indeed, one can see that the password is never used during the simulation. It does not need to be chosen in advance, but at the very end only. Then, an information-theoretic analysis can be done which simply uses cardinalities of some sets.

To this aim, we first cancel a few more games, wherein for some pairs $(W, Y) \in G_1^2$, involved in a communication between an instance $S'$ and either the adversary or an instance $C$, there are two distinct elements $h^w$ such that the tuple $(U_1, U_2, Y, W/h^w, \text{CDH}_{G_1}(W/h^w, Y))$ is in $\Lambda_{h^w}$ (which event is denoted $\text{CollH}_h$). Note that the simulator can get the value $W$ from $(U_1, U_2)$. By Shoup’s difference lemma, the probability is upper-bounded by

$$|\Pr[\text{AskH}_5] − \Pr[\text{AskH}_4]| \leq \frac{q_{se} + q_{ex}}{q} . \quad (7)$$

The event $\text{CollH}_h$ can be upper-bounded, granted the following lemma:

**Lemma 1:** If for any pair $(W, Y) \in G_1^2$, involved in a communication with an instance $S'$, there are two elements $h^w_0$ and $h^w_1$ such that $(U_1, U_2, Y, W/h^w_i, Z_i = \text{CDH}_{G_1}(W/h^w_i, Y))$ is in $\Lambda_{h^w_i}$, one can solve the computational Diffie-Hellman problem:

$$|\Pr[\text{CollH}_h] − \Pr[\text{AskH}_6]| \leq \Pr[\text{CollH}_h] . \quad (8)$$

**Proof.** We prove this lemma by showing the reduction to the CDH problem when event $\text{CollH}_h$ happens. We assume that there exist $(W, Y \equiv P') \in G_1^2$ involved in a communication with an instance $S'$, and two elements $h^w_0 \equiv Q^0$ and $h^w_1 \equiv Q^1$ such that the tuple $(U_1, U_2, Y, W/h^w_i, Z_i = \text{CDH}_{G_1}(W/h^w_i, Y))$ is in $\Lambda_{h^w_i}$ for $i = 0, 1$. Then,

$$Z_i = \text{CDH}_{G_1}(W/h^w_i, Y) = \text{CDH}_{G_1}(W \times Q^{-w_i}, Y) = \text{CDH}_{G_1}(W, Y) \times \text{CDH}_{G_1}(Q, Y)^{w_i} = \text{CDH}_{G_1}(W, Y) \times \text{CDH}_{G_1}(P, Q)^{w_i}. \quad (9)$$

As a consequence, $Z_1/Z_0 = \text{CDH}_{G_1}(P, Q)^{(w_0−w_1)}$, and thus $\text{CDH}_{G_1}(P, Q) = (Z_1/Z_0)^{\psi}$ where $\psi$ is the inverse of $\gamma(w_0 − w_1)$ in $\mathbb{Z}_q^*$. The latter exists since $h^w_0 \equiv h^w_1$ and $y \neq 0$. By guessing the two queries asked to the $H_j$, one concludes the proof. \hfill $\Box$

In order to conclude the proof, let us study separately the three sub-cases of $\text{AskH}_2$, and then $\text{AskH}_3w2$ (keeping in mind the absence of several kinds of collisions for partial transcripts, and for $h^w$ in $H$-queries):

- **AskH2-Passive:** About the passive transcripts (in which both $W$ and $Y$ have been simulated), one...
can state the following lemma:

**Lemma 2:** If for any pair \((W, Y) \in \mathcal{G}_1^2\), involved in a passive transcript, there is an element \(h^w\) such that \((U_1, U_2, Y, W/h^w, Z = \text{CDH}_{\mathcal{G}_1}(W/h^w, Y))\) is in \(\Lambda_{\text{H}}\), one can solve the computational Diffie-Hellman problem:

\[
\Pr[\text{AskH2-Passive}_6] \leq q_{\text{se}} \times \text{Succ}_{\mathcal{G}_1}^{\text{cdh}}(t_1 + 2\tau_e). \tag{11}
\]

**Proof.** We prove this lemma by showing the reduction to the CDH problem when event AskH2-Passive \(_6\) happens. We assume that there exist \((W \equiv g^x, Y \equiv P^y) \in \mathcal{G}_1^2\) involved in a passive transcript and \(h^w \equiv Q^w\) such that the tuple \((U_1, U_2, Y, W/h^w, Z)\) is in \(\Lambda_{\text{H}}\). As above,

\[
Z = \text{CDH}_{\mathcal{G}_1}(W, Y) = \text{CDH}_{\mathcal{G}_1}(Q, Y)^{-w}
\]

\[
= P^y \times \text{CDH}_{\mathcal{G}_1}(P, Q)^{-w}. \tag{12}
\]

As a consequence, \(\text{CDH}_{\mathcal{G}_1}(P, Q) = (Z/P^y)^{\phi}\) where \(\phi\) is the inverse of \(-yw\) in \(\mathbb{Z}_q^*\). The latter exists since we have excluded the cases where \(y = 0\) or \(w = 0\). By guessing the query asked to the \(H_j\), one concludes the proof. \(\square\)

- **AskH2-WithC:** This corresponds to an attack where the adversary tries to impersonate \(S\) to \(C\). But each authenticator sent by the adversary has been computed with at most one \(h^w\) value:

\[
\Pr[\text{AskH2-WithC}_6] \leq \frac{q_{\text{se}}}{N}. \tag{13}
\]

- **AskH2-WithS:** The Lemma 1, applied to games where the event CollH\(_6\) did not happen, states that for each pair \((W, Y)\) involved in a transcript with an instance \(S\), there is at most one element \(h^w\) such that \(h^w \equiv Q^w\), the corresponding tuple is in \(\Lambda_{\text{H}}\): The probability for the adversary over a random password is as above:

\[
\Pr[\text{AskH2-WithS}_6] \leq \frac{q_{\text{se}}}{N}. \tag{14}
\]

About AskH3w2 (when the above three events did not happen), it means that only executions with an instance of \(S\) (and either \(C\) or the adversary) may lead to acceptance. Exactly the same analysis as for AskH2-Passive and AskH2-WithS leads to

\[
\Pr[\text{AskH3w2}_6] \leq \frac{q_{\text{se}}}{N} + q_{\text{H}} \times \text{Succ}_{\mathcal{G}_1}^{\text{cdh}}(t_1 + 2\tau_e). \tag{15}
\]

As a conclusion, we get an upper-bound for the probability of AskH\(_6\) by combining all the cases:

\[
\Pr[\text{AskH6}_6] \leq \frac{3q_{\text{se}}}{N} + 2q_{\text{H}} \times \text{Succ}_{\mathcal{G}_1}^{\text{cdh}}(t_1 + 2\tau_e). \tag{16}
\]

Combining inequalities (4), (5), (6), (7), (9) and (16), one gets

\[
\Pr[S_0] \leq \frac{q_{\text{se}}}{2^k} + \frac{1}{2} + \Delta, \tag{17}
\]

where

\[
\Delta \leq \frac{3q_{\text{se}}}{N} + 2q_{\text{H}} \times \text{Succ}_{\mathcal{G}_1}^{\text{cdh}}(t_1 + 2\tau_e) + \frac{q_{\text{se}} + q_{\text{ex}}}{2q} + q_{\text{H}} \times \text{Succ}_{\mathcal{G}_1}^{\text{cdh}}(t_1 + \tau_e) + \frac{(q_{\text{ex}} + q_{\text{se}})^2}{2q} + 3(q_{\text{ex}} + q_{\text{se}})^2 (16)
\]

\[
+ 2q_{\text{se}} \times \text{Adv}_{\text{IBE}}^{\text{ind-id-cpa}}(B) \tag{18}
\]

Finally, one can get the result as desired by noting that \(\text{Adv}_{\text{P}}^{\text{Adv}}(\mathcal{A}) = 2 \Pr[S_0] - 1\).

### 8. Comparison

In this section, we compare the PAKE protocols using the BF-IBE scheme [15, 18] (PAKE-CS [17], iPAKE [2], UKAM-Pie [21], SPAIBE of Sect. 5) in terms of efficiency, number of passes and security against a malicious PKG.

We summarize a comparative result in Table 1 where

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Computation costs</th>
<th>Communication costs</th>
<th># of passes</th>
<th>Security against a malicious PKG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client C</td>
<td>Server S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAKE-CS [17]</td>
<td>1Pairing + 5Exp(_{\mathcal{G}<em>1}) + 1Exp(</em>{\mathcal{G}_1})</td>
<td>1Pairing + 4Exp(_{\mathcal{G}_1})</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iPAKE [2]</td>
<td>1Pairing + 2Exp(_{\mathcal{G}<em>1}) + 1Exp(</em>{\mathcal{G}_1})</td>
<td>1Pairing + 2Exp(_{\mathcal{G}_1})</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UKAM-Pie [21]</td>
<td>1Pairing + 3Exp(_{\mathcal{G}<em>1}) + 1Exp(</em>{\mathcal{G}_1})</td>
<td>1Pairing + 2Exp(_{\mathcal{G}_1})</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPAIBE (Sect. 5)</td>
<td>1Pairing + 3.17Exp(_{\mathcal{G}<em>1}) + 1Exp(</em>{\mathcal{G}_1})</td>
<td>1Pairing + 2Exp(_{\mathcal{G}_1})</td>
<td>2</td>
<td>Yes</td>
</tr>
</tbody>
</table>
we have proposed a strengthened PAKE (SPAIBE) protocol to protect the session key with any client by impersonating the server. Then, the PKG can find out all clients’ passwords by just eavesdropping. The iPAKE and UKAM-PiE protocols are insecure against passive attacks by a malicious PKG (as in Sect. 4). Compared to the UKAM-PiE protocol [21], the SPAIBE protocol has almost same efficiency while providing security against passive/active attacks by a malicious PKG.

9. Conclusions

In this paper, we have revisited the iPAKE protocol [2] using the Boneh-Franklin IBE scheme [15], [18], and the UKAM-PiE protocol in ISO/IEC 11770-4/AMD 1 [21]. That is, the iPAKE and UKAM-PiE protocols are insecure against passive/active attacks by a malicious PKG where the malicious PKG can find out all clients’ passwords by just eavesdropping on the communications, and the PKG can share a session key with any client by impersonating the server. Then, we have proposed a strengthened PAKE (SPAIBE) protocol with IBE, which provides security against passive/active attacks by a malicious PKG. Also, we have formally proved the security of the SPAIBE protocol in the random oracle model [22] and compared the PAKE protocols using the BF-IBE scheme (PAKE-CS [17], iPAKE [2], UKAM-PiE [21], SPAIBE) in terms of efficiency, number of passes and security against a malicious PKG.

Acknowledgments

We sincerely appreciate the anonymous reviewers’ constructive and valuable comments on this paper.

References


SeongHan Shin received his B.S. and M.S. degrees in computer science from Pukyong National University, Busan, Korea, in 2000 and 2002, respectively. In 2005, he received his Ph.D. degree in information and communication engineering, information science and technology from The University of Tokyo, Tokyo, Japan. In 2006, he joined the National Institute of Industrial Science and Technology (AIST), Japan. Currently, he is a senior research scientist at the Cyber Physical Security Research Center (CPSEC), AIST. He received the CSS 2003 Student Paper Award, IWS2005/WPMC’05 Best Student Paper Award, and WISA 2019 Best Paper Premium Award. His research interests include information security, cryptography, and wireless security.