Robust Authenticated Key Exchange Using Passwords and Identity-Based Signatures

Jung Yeon Hwang[1], Seung-Hyun Kim[1], Daeseon Choi[2], Seung-Hun Jin[1], and Boyeon Song[3](✉)

[1] Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea
{videmot,ayo,jinsh}@etri.re.kr

[2] Department of Medical Information, Kongju University, Nonsan-si, Korea
sunchoi@kongju.ac.kr

[3] Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea
bysong@kisti.re.kr

Abstract. In the paper we propose new authenticated key exchange (AKE) protocols from a combination of identity-based signature (IBS) and a password-based authentication. The proposed protocols allows for a client to execute a convenient authentication by using only a human-memorable password and a server's identity. The use of an IBS gives security enhancements against threats from password leakage. A server authentication method is based on an IBS, which is independent of a password shared with a client. Even if a password is revealed on the side of a client protected poorly, server impersonation can be prevented effectively. In addition, our protocols have resilience to server compromise by using 'password verification data', not a true password at the server. An adversary cannot use the data revealed from server compromise directly to impersonate a client without additional off-line dictionary attacks. We emphasize that most of existing password-based AKE protocols are vulnerable to subsequent attacks after password leakage.

Our first hybrid AKE protocol is constructed using concrete parameters from discrete logarithm based groups. It is designed to give resilience to server compromise. Our second protocol is a simplified version of the first protocol where the computation cost of a client is cheap. Generalizing the basic protocols, we present a modular method to convert Diffie-Hellman key exchange into an AKE protocol based on a password and an IBS. Finally, we give performance analysis for our protocols and comparison among known hybrid AKE protocols and ours. As shown later in the paper, our protocols provide better performance. Our experimental results show that the proposed protocols run in at most 20 ms. They can be widely applied for information security applications.

Keywords: Authentication · Password · Identity-based signature · Key exchange

This work was supported by the ICT R&D program of MSIP/IITP [B1206-15-1007, Development of Universal Authentication Platform Technology with Context-Aware Multi-Factor Authentication and Digital Signature].

L. Chen and S. Matsuo (Eds.): SSR 2015, LNCS 9497, pp. 43–69, 2015.
DOI: 10.1007/978-3-319-27152-1_3

1 Introduction

Explosive growth of computing environment is opening the era of big data. Application domains of the pervasive computing are connected densely, and entities with smart devices exchange information frequently in the advanced network. Massive amounts of data can be collected from various sources and processed for user-centric services. Furthermore, automated analysis and deep learning technologies are actively developed to extract highly valuable knowledge from a huge amount of data. In the upcoming future, a user will be able to enjoy unprecedented convenience timely from the advanced information service.

These services will be available when not only feasibility but also security properties are well provided. For secure data transmission between entities, key exchange is one of the most crucial and fundamental protocols. That is, to access valuable data resources on a server (or a service provider), a client should establish a temporal digital key with the server securely. The shared key builds a *secure channel* between a client and a server, which provides security properties including data integrity and confidentiality. Diffie-Hellman (DH) key exchange (KE) [19] is a well-known popular protocol for key establishment between two entities.

In general, a network used to establish a shared key is public and insecure, where an adversary may control exchanged messages in an adaptive way. An adversary is able to perform impersonation attacks to a user by intercepting and/or modifying messages. Thus, authentication of legal participants is necessary for secure key exchange. An authenticated key exchange (AKE) protocol has been widely studied and developed for a long time, to achieve key establishment and authentication of participants over a public network. AKE is mainly constructed by combining the DH KE and an authentication method, for which entities' computing capability and users' convenience should be considered. In a client-server model, a client is a typically human who has a device(s) with limited computing resources, while a server is a powerful machine which can store high-entropy secret numbers.

In a client-server model, a password is a commonly used authentication factor, because a client can generate and memorize it easily. In practice, most of IT services use ID/password as a log-in method. To construct a secure password-based AKE (PAKE) lots of research have been performed [7,32,40]. As a password used for authentication is low-entropy, PAKE is vulnerable to *dictionary* attacks which systematically check all possible passwords from a password space of small size until the correct one is found. Dictionary attacks can be mounted in two types, i.e., on-line and off-line. An on-line dictionary attack is mounted by using guessed passwords. It is easy to prevent it just by limiting the number of on-line password trials. An off-line dictionary attack is performed without interaction with a server, when some information to confirm a true password is obtained. Thus, to resist an off-line dictionary attack, a PAKE protocol makes sure not to reveal any information related to a password. when sending a message including the password.

For this reason, Encrypted Key Exchange (EKE) was introduced in [7], where at least one party encrypts a key value using a password and sends it to a second party who decrypts it to negotiate a shared key with the first party. Since then, EKE has been modified and extended in various ways. Some PAKE protocols have been developed as international standards by IEEE [31], IETF, ISO/IEC JTC 1/SC 27 [32], and ITU-T [33]. Most PAKE protocols including EKE are constructed in a shared-password authentication model where a client and a server identifies each other by using a shared password (or variant of a password). Intrinsically, the model is vulnerable to threats from password exposure, in the sense that a password stolen from one party can be used to impersonate the other party. There are various possibilities to leak a password, for example, by malware, hacking, shoulder surfing attacks or from lost/stolen portable devices.[1] Since a client's device may be insufficiently protected, password exposure could be more realistic. When a password of a client is revealed, it is inevitable that an adversary impersonate the client. But, if a server can be impersonated to the client, it will bring more dangerous and serious risks. For example, malicious modification of critical information such as clients' financial services or healthcare will be possible.

As a solution to the above issue, a *hybrid* AKE has been introduced which is constructed in conjunction of an asymmetric encryption scheme and password-based authentication. The protocols by [23,25] make use of a public key encryption, and the protocol of [53] is based on an identity-based encryption (IBE). The intuition to prevent server impersonation attacks is to use an independent decryption key for a server. For password-based authentication, the protocols take a simple approach to encrypt a password with a server's public key. To guarantee confidentiality on a password, they apply a highly secure encryption to meet so-called CCA-security.[2]

Hybrid protocols based on IBE may be preferable in a client-server model because a client is assumed to be a human who can merely memorize limited information such as a password or server's ID. However, CCA-secure IBE encryption entails relatively complex computation of parameters. It will impose expensive computation or communication costs on a client side using a device with a limited resource. In addition, the above-mentioned hybrid protocols do not consider server compromise. A server manages a password file for a large number of clients. The file contains secret values to be used to authenticate clients. If the password file is revealed, it will cause disastrous results, because any client can be immediately subject to impersonation.

1.1 Our Contributions

In the paper, we propose new efficient yet robust AKE protocols from a combination of a password-based authentication and an identity-based signature (IBS).

[1] These are different from dictionary attacks to reveal a password.

[2] The notion of CCA security means that a PKE scheme should reveal no meaningful information about the original message from public ciphertexts to attackers who can probe the decryption oracle with chosen ciphertexts.

For distinction, they are called *IBS-PAKE* protocols. An IBS can be used by not only a powerful server but also a client, while a client is enough to memorize a password to invoke the protocol. The adoption of an IBS gives desired solutions to party compromise issues as follows.

- Basically, a server executes an independent authentication based on an IBS. Even if a password is revealed from a client, server impersonation is impossible without access to the server's IBS key. It will be reasonable for a server to manage a sinlge key secretly, rather than the whole password file of large size.
- In an IBS scheme, a public key may be defined by an arbitrary public string such as an e-mail address or a company/brand name. A client can authenticate a server by verifying a server's IBS with a server's publicly known identity. For himself or herself, the client executes password-based authentication. Thus he or she can do a convenient authentication based on only a human-memorable password and a server's identity without holding a high entropy secret key.
- Finally, our protocol allows for a client to use an IBS by accessing an IBS key stored at a server. More concretely, the client receives an encryption of the IBS key from the server and decrypt it with his or her password. The client should know the original password to obtain the correct signing key. It involves a kind of a knowledge proof of a password. This idea can be applied to achieve resilience to server compromise [24]. Assume that the knowledge proof is required to a client for each login. An adversary cannot use the password file stolen the server directly for impersonation attacks but additionally making an off-line dictionary attack to extract clients' real passwords.

In order to construct an IBS-PAKE protocol, we take a modular approach first, that is, present two modular methods to yield IBS-PAKE protocols generically when a symmetric PAKE and an IBS are given. The first method is designed to achieve resilience to server compromise. The underlying idea to achieve resilience to server compromise is similar to that of [24] based on a normal signature scheme. However, there is a difference between the ideas because a public verification key, i.e., a client's identity is known to an adversary in our protocol while hidden in [24]. Our second one is a simplified version of the first method, to handle the situation that server compromise is not mainly considered due to strong security at the side of a server. Compared to the first method, it can run in a single round which consists of two passes independently sent from a client and a server. In addition, the computation cost at the side of a client is quite cheap.

In the modular methods, IBS schemes can be selected flexibly and independently according to a design strategy. For example, we can pick an IBS scheme with low signing (or verifying) cost for a client. Also, an IBS-PAKE protocol can be constructed from more realistic hardness assumptions. As instances of IBS-PAKE resulting from the modular methods, two protocols are presented by using concrete discrete logarithm parameters. They are built on the IBS scheme constructed from the Schnorr signature.

Finally, we give performance analysis for our protocols and comparison among known AKE protocols using an identity-based cryptosystem and a password [53], and ours. As shown in the comparison table, our AKE protocols provide better performance with a robust security property. We also present experimental results to show that the proposed protocol runs in at most 20 ms.

1.2 Related Work

Since the introduction of Diffie-Hellman KE protocol [19], KE protocols have been widely studied to achieve various authentication goals [10,11,37]. AKE protocols have been developed largely according to two authentication types, i.e., symmetric and asymmetric. Symmetric authentication type assumes that participants share a secret key before running a protocol [10,11]. A password-based KE (PAKE) protocol is a primary example of symmetric authentication where a client and a server share a password as an authentication factor. The formal treatment for PAKE was given in [8,10]. Refer to [40] for a survey of PAKE. Some research proposes PAKE protocols with security under standard assumptions [3,35]. Recently, research on PAKE protocols [3] focuses on meeting highly theoretical security requirement such as UC model [16]. Asymmetric authentication type assumes that a participant uses a secret key and its corresponding public key. The secret key is kept secret by the participant while the public key is set to be public and so anyone can access it. Since the secret key is a random long bit string, a client needs a mean to store it. For example, we can consider KE based on a standard public key digital signature [37] and identity-based KE [17]. A hybrid authentication type combines symmetric and asymmetric types to gain merits of the two types [25]. In contrast to the symmetric and asymmetric types, hybrid authentication and KE have not been studied intensively.

1.3 Organization

The remainder of this paper is organized as follows. In Sect. 2, we briefly review some preliminaries. In Sect. 3 we give a security model for an IBS-PAKE. In Sect. 4 we present an IBS scheme based on Schnorr signature. In Sect. 5 we present modular methods to yield IBS-PAKE protocols, and also concrete IBS-PAKE protocols using discrete logarithm parameters and prove the security. In Sect. 6 we give a performance analysis and comparison among known AKE protocols. Finally, we conclude in Sect. 7.

2 Preliminaries

In this section, we review some background knowledge for our construction.

Let $\mathsf{poly}(\lambda)$ denote a polynomial in variable λ. We define that $\nu(\lambda)$ is a negligible function if $\nu(\lambda) < 1/\mathsf{poly}(\lambda)$ for any $\mathsf{poly}(\lambda)$ and sufficiently large λ. We denote by $A \overset{?}{=} B$ the equality test between two group elements, A and B. We denote by $s \overset{R}{\leftarrow} S$ the operation that picks an element s of set S uniformly at random. We denote by '$||$' the concatenation operation on strings.

Computational Assumptions. For the security of our construction, computational assumptions such as discrete logarithm, decisional Diffie-Hellman, and computational Diffie-Hellman assumptions, are needed. For more details, refer to Appendix.

Symmetric Encryption. A symmetric encryption (SE) scheme consists of two functions, $\mathcal{E}$ and $\mathcal{D}$ associated to key space $\mathcal{K}_{SE}$=$\{0,1\}^\lambda$.

- $\mathcal{E}_k(m)$. It takes as input a key $k \in \mathcal{K}_{SE}$ and a message $m \in \{0,1\}^n$ and outputs a ciphertext $\chi \in \{0,1\}^n$.
- $\mathcal{D}_k(\chi)$. It takes as input a key k and a ciphertext $\chi \in \{0,1\}^n$, and then outputs a message $m \in \{0,1\}^n$.

To define *one-time indistinguishability* for SE, we consider the following game:

$$\begin{aligned} \textit{Challenge} &: (m_0, m_1, \eta) \leftarrow \mathcal{A}(1^\lambda) \\ \textit{Response} &: k \overset{R}{\leftarrow} \mathcal{K}_{SE},\ b \overset{R}{\leftarrow} \{0,1\},\ \chi^* \leftarrow \mathcal{E}_k(m_b) \\ \textit{Guess} &: b' \leftarrow \mathcal{A}(\eta, \chi^*) \end{aligned}$$

Assume that m_0 and m_1 has a same length. We define $\mathsf{Adv}^{\textit{IND-OTK}}_{\mathcal{A},\mathsf{SE}}(t)= |\Pr[b = b'] - 1/2|$, where $\mathcal{A}$ runs in time t, and $\mathsf{Adv}^{\textit{IND-OTK}}_{\mathsf{SE}}(t) = \max_{\mathcal{A}}[\mathsf{Adv}^{\textit{IND-OTK}}_{\mathcal{A},\mathsf{SE}}(t)]$ where the maximum is taken over all $\mathcal{A}$. We say that SE is *one-time secure* if $\mathsf{Adv}^{\textit{IND-OTK}}_{\mathsf{SE}}(t)$ is negligible.

Identity-Based Signature. An IBS scheme consists of four algorithms for setup, private key extraction, signing, and verifying [9,45]. These are denoted by Setup, KeyExt, Sign, and Vrfy, respectively.

- Setup takes as input a security parameter λ, and outputs a master secret key, msk and a set of public parameters, pp.
- KeyExt takes as input (pp,msk) and an identity $ID \in \{0,1\}^{\ell_{\mathsf{id}}}$ for $\ell_{\mathsf{id}} \in \mathbb{N}$, and then outputs a private key, sk_{ID}.
- Sign takes as input pp, an identity ID, a key sk_{ID}, a message $m \in \{0,1\}^*$, and then outputs a signature, σ.
- Vrfy takes as input pp, ID, a signature σ, and a message m, then outputs 0 (meaning 'invalid') or 1 (meaning 'valid').

We say that an IBS scheme is *correct* if the following condition holds: $1 \leftarrow \mathsf{Vrfy}(pp, ID, \sigma, m)$ for any pair of (m, ID) where $(pp, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$, $sk_{ID} \leftarrow \mathsf{KeyExt}(pp, \mathsf{msk}, ID)$, and $\sigma \leftarrow \mathsf{Sign}(pp, ID, sk_{ID}, m)$.

Next we consider a game to define the existential unforgeability under *chosen message and identity attacks (CMIDA)* for $\mathsf{IBS} = (\mathsf{Setup}, \mathsf{KeyExt}, \mathsf{Sign}, \mathsf{Vrfy})$. The game consists of SETUP, QUERY, and FORGE phases. Let EQ and SQ denote an extraction query and a signing query, respectively. In the QUERY phase, $\mathcal{F}$ is allowed to make extraction and signing queries to the key extraction and signing oracles adaptively.

$$\begin{aligned}
&\textit{Setup}: (pp, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)\\
&\textit{Query}: EQ(ID) \leftarrow \mathcal{F},\ \ sk_{ID} \leftarrow \mathsf{KeyExt}(pp, \mathsf{msk}, ID)\\
&\qquad\quad\ SQ(ID', m) \leftarrow \mathcal{F},\ \ \sigma \leftarrow \mathsf{Sign}(pp, ID', sk_{ID'}, m)\\
&\textit{Forge}: (ID^*, m^*, \sigma^*) \leftarrow \mathcal{F}
\end{aligned}$$

Let L_{Ex} and L_{Sign} denote a list of all extraction and signing queries that $\mathcal{F}$ have made. We say that the forgery, (ID^*, m^*, σ^*) is *valid* if $ID^* \notin L_{Ex}$ and $(ID^*, m^*) \notin L_{Sign}$, and $\mathsf{Vrfy}(pp, ID^*, \sigma^*, m^*)$=1. The EUF-CMIDA-advantage of adversary $\mathcal{F}$, denoted by $\mathsf{Adv}_{\mathsf{IBS}}^{\textit{EUF-CMIDA}}(\mathcal{F})$, is defined as the probability that $\mathcal{F}$ outputs a valid forgery in the above experiment. That is, $\mathsf{Adv}_{\mathcal{F},\mathsf{IBS}}^{\textit{EUF-CMIDA}}(t)$= $\Pr[ID^* \notin L_{Ex} \wedge \mathsf{Vrfy}(pp, m^*, ID^*, \sigma^*) = 1]$, where $\mathcal{F}$ runs in time t. We also define that $\mathsf{Adv}_{\mathsf{IBS}}^{\textit{EUF-CMIDA}}(t) = \max_{\mathcal{F}}[\mathsf{Adv}_{\mathcal{F},\mathsf{IBS}}^{\textit{EUF-CMIDA}}(t)]$ where the maximum is taken over all $\mathcal{F}$. We say that IBS is *existentially unforgeable* if $\mathsf{Adv}_{\mathsf{IBS}}^{\textit{EUF-CMIDA}}(t)$ is negligible.

Various IBS schemes have been proposed from various mathematical parameters such as DL, composite numbers, and bilinear maps [9,45].

3 Security Model

We present a security model for a hybrid AKE protocol based on a password and an IBS by modifying known security models for AKE such as [10,24,37] and extending the model of [15,53]. Our model captures security by considering resilience to server compromise.

PARTICIPANTS. Let **Clients** and **Servers** denote sets of clients and servers, respectively. Let $\mathcal{U}$ be a set of all principals and defined by **Clients** $\cup$ **Servers**. The number of principals is bounded by a polynomial in a security parameter. We assume that each principal is labeled by a unique identity of a ℓ_{id}-bit string for a positive integer ℓ_{id}. For example, ID_C and ID_S are used to denote client C and server S, respectively.

In the model, we assume that there is a trusted third party, called KGA (Key Generation Authority) who manages the key extraction algorithm, KeyExt of an IBS scheme and keeps the master secret key, msk.[3] Whenever a principal requests, KGA issues a long-term secret signing key corresponding to the identity of the principal. A principal with ID obtains a long-term secret signing key sk_{ID} from KGA. We note that a client is enough to use only a human-memorable password to execute the protocol. The password or its verifier is shared between a client and a server. However, though the client holds no long-term secret key of high-entropy, he or she can use the IBS key transmitted from the server during a run of a protocol.

[3] To prevent misuse of the master secret key, the authority of KGA can be distributed into multiple authorities by using known threshold techniques [4,12].

Initialization. In the initialization phase, Setup, $\mathsf{Extract}$, and $\mathsf{Registration}$ processes are executed. Setup generates global system parameters and keys including the master secret key, $\mathtt{msk}$ of IBS. The global system parameters, denoted by pp and identities, ID_U are publicly known to a client and a server, and also an adversary. Through $\mathsf{Extract}$, a principal obtains a signing key corresponding to an identity. An IBS generated by the signing key can be verified by the identity. We assume that a principal is identified in a pre-defined way before issuing a key.

Whenever a client of ID_C wants to join as a valid user (of a service) to a server of ID_S, $\mathsf{Registration}$ is executed between them. Let $\mathcal{D}_{PW}$ denote a password space, i.e., a dictionary of passwords. Assume that a password, $pw_C \in \mathcal{D}_{PW}$ is generated by the client according to a pre-defined password creation policy. After completing the registration, the server stores a password verifier, pv_C which is derived from pw_C. For example, pv_C can be computed from a hash function or some deterministic function with input pw_C. Let $\pi_S[C] = (ID_C, pv_C)$ and $\mathcal{PF}_S = \{\pi_S[C]\}_{C \in \mathbf{Clients}}$, which is called a password file, including all authentication information registered for clients.

Protocol Execution. A principal is allowed to invoke the protocol (multiple times) with a partner principal to establish a session key. It can be run in a concurrent way. Multiple executions of a principal is modeled via instance. Let s be a positive integer. The s^{th} instance of a principal with ID_U is represented by Π_U^s where $U \in \mathcal{U} = \mathbf{Clients} \cup \mathbf{Servers}$. The index, s sequentially increases according to the number of executions of the principal [11,37].

Partnering. A session id of instance Π_U^s is defined as the concatenation of all transcripts sent and received between an instance of a client and an instance of a server during the execution of the protocol. For $U \in \mathcal{U}$, let sid_U^s denote a session id for instance sid_U^s.

Partner identifier pid_U^s for instance Π_U^s is defined by a set of the identities of protocol participants who intend to establish a session key. We say that instance Π_U^s *accepts* when it computes a session key, sk_U^s. Let acc_U^s denote an boolean variable to show whether a given instance has accepted or not. Assume that, if an instance computes a session key, sk_U^s, it outputs $(\mathsf{sid}_U^s, \mathsf{sk}_U^s)$. For $C \in \mathbf{Clients}$ and $S \in \mathbf{Servers}$, we say that Π_C^i and Π_S^j are *partnered* if and only if (1) $\mathsf{pid}_C^i = \mathsf{pid}_S^j$; (2) $\mathsf{sid}_C^i = \mathsf{sid}_S^j$ and (3) they have both accepted.

Adversarial model. An adversary $\mathcal{A}$ is a PPT algorithm that has complete control over all the communications. Attacks that $\mathcal{A}$ can make are modeled via the following queries.

- $\mathsf{Extract}(ID_U)$: By this query, $\mathcal{A}$ is given the long-term secret key of ID_U where $U \in \mathcal{U} = \mathbf{Clients} \cup \mathbf{Servers}$.
- $\mathsf{Execute}(ID_C, ID_S)$: $\mathcal{A}$ is given the complete transcripts of an honest execution between C and S. It models passive attacks eavesdropping an execution of the protocol.

- $\mathsf{Send}(\Pi_U^s, m)$: $\mathcal{A}$ is given the response generated by Π_U^s according to the protocol, when m is given to Π_U^s. It models an active attack where the adversary controls messages elaborately. By this query, a message m can be sent to instance Π_U^s.
- $\mathsf{Reveal}(\Pi_U^s)$: $\mathcal{A}$ is given the session key that instance Π_U^s has generated. It models a *known key* attack.
- $\mathsf{Corrupt}(ID_U)$: It models exposure of the long-term secret key held by ID_U where $U \in \mathcal{U}$. $\mathcal{A}$ is given $\mathcal{PF}_S = \{\pi_S[C]\}_{C\in}$ **Clients** if $U \in$ **Servers**, and otherwise pw_U.
- $\mathsf{Test}(\Pi_U^s)$: It is used to define the advantage of $\mathcal{A}$. When $\mathcal{A}$ asks this query to an instance Π_U^s, a random bit b is chosen; if $b = 1$ then the session key is returned. Otherwise, a random string is drawn from the space of session keys, and returned. $\mathcal{A}$ is allowed to make a Test query once, at any time.

In the model we consider two types of adversaries according to their attack types. The attack types are simulated by the queries issued by an adversary. A *passive adversary* is allowed to issue $\mathsf{Execute}$, Reveal, $\mathsf{Corrupt}$, and Test queries, while an *active adversary* is additionally allowed to issue Send and $\mathsf{Extract}$ queries. Even though $\mathsf{Execute}$ query can be using Send queries repeatedly, we use $\mathsf{Execute}$ query for more concrete analysis.

FRESHNESS. An instance Π_U^s is said *fresh* (or holds a *fresh* key *ssk*) if the following conditions hold:

1. $\mathsf{Reveal}(\Pi_U^s)$ has not been asked for all $U \in \mathsf{pid}_U^s$,
2. $\mathsf{Corrupt}(ID_{U'})$ has not been asked for $U' \in \mathsf{pid}_U^s$.

An instance Π_U^s is said *semi-fresh* (or holds a *semi-fresh* key *ssk*) if the following conditions hold:

1. $\mathsf{Reveal}(\Pi_U^s)$ has not been asked for all $U \in \mathsf{pid}_U^s$,
2. $\mathsf{Corrupt}(ID_U)$ has not been asked, and
3. $\mathsf{Corrupt}(ID_S)$ has not been asked for $U \in$ **Clients**.

AKE SECURITY. Let $\mathcal{P}$ be an IBS-PAKE protocol and $\mathcal{A}$ an adversary to attack it. $\mathcal{A}$ is allowed to make oracle queries in an adaptive way and receives the corresponding responses from oracles. At some point during the game a Test query is asked to a fresh or semi-fresh oracle, and $\mathcal{A}$ may continue to make other queries. Finally $\mathcal{A}$ outputs its guess b' for the bit b used by the Test oracle, and terminates. Let Succ denote an event that $\mathcal{A}$ correctly guesses the bit b. The advantages of $\mathcal{A}$ must be measured in terms of the security parameter λ.

The IBS-PAKE-advantage, $\mathsf{Adv}_{\mathcal{P}}^{IBS\text{-}PAKE}(\mathcal{A})$ of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ correctly guesses the bit in the above experiment. That is, $\mathsf{Adv}_{\mathcal{A},\mathcal{P}}^{IBS\text{-}PAKE}$ $(\lambda) = 2 \cdot \Pr[\mathsf{Succ}] - 1$, where $\mathcal{A}$ runs in time t. We also define that $\mathsf{Adv}_{\mathcal{P}}^{IBS\text{-}PAKE}$ $(\lambda) = \max_{\mathcal{A}}[\mathsf{Adv}_{\mathcal{A},\mathcal{P}}^{IBS\text{-}PAKE}(\lambda)]$ where the maximum is taken over all $\mathcal{A}$.

Let $|\mathcal{D}_{\mathcal{PW}}|$ be the size of the password space. Let q_s be the maximum number of Send queries. We say a protocol $\mathcal{P}$ is a *secure IBS-PAKE* protocol if the following condition hold: For a negligible function $\epsilon(\lambda)$, $\mathsf{Adv}_{\mathcal{P}}^{IBS\text{-}PAKE}(\lambda)$ is bounded by $\frac{q_S}{|\mathcal{D}_{\mathcal{PW}}|} + \epsilon(\lambda)$.

4 Our Identity-Based Signature Scheme

In this section, we present an IBS scheme that works with discrete logarithm parameters. It will be used for our AKE as a building block in the next section.

– Setup. For a given security λ, it generates a cyclic group, $\mathbb{G}$ of prime order q and a random generator, g of $\mathbb{G}$. It picks $\theta \in \mathbb{Z}_q^*$ uniformly at random and computes $u = g^{\theta}$. It also generates independent cryptographic hash functions, $H : \{0,1\}^* \to \{0,1\}^{\ell}$. It outputs the system public parameters $pp = (\mathbb{G}, q, g, u, H)$ and the corresponding master secret key $\mathsf{msk} = \theta$.
– $\mathsf{KeyExt}(pp, \mathsf{msk}, ID)$. It picks $r \in \mathbb{Z}_q^*$ uniformly at random. It computes $R = g^r$, $w = H(ID, R)$, and $v = r + \theta w \pmod q$. It then returns $sk_{ID} = (R, v)$.
– $\mathsf{Sign}(pp, sk_{ID}, ID, m)$. It takes as input pp, a message $m \in \{0,1\}^*$, an identity ID and a key $sk_{ID} = (R, v)$. It picks $e \in \mathbb{Z}_q^*$ and compute $E = g^e$, $z = H(m, ID, E)$ and $d = e - vz \pmod q$. The signature on m is $\sigma = (d, z, R)$.
– $\mathsf{Vrfy}(pp, ID, \sigma, m)$. It takes as inputs pp, a message m, ID and a signature $\sigma = (d, z, R)$. It computes $E' = g^d \cdot (R \cdot u^w)^z$ and $w = H(ID, R)$. Finally, it checks if $z = H(m, ID, E')$ holds. If the equality holds, it outputs 1, and otherwise, 0.

The above IBS scheme is correct because we have $g^d \cdot (R^z \cdot u^{wz}) = g^{e-(r+\theta w)z} \cdot (g^{rz} \cdot g^{\theta wz}) = g^e$ and so $z = H(m, ID, E) = H(m, ID, g^d \cdot R^z \cdot u^{wz})$ where $w = H(ID, R)$ and $(R = g^r, v = r + \theta w)$ is generated from KeyExt for ID.

The idea behind the above construction is to combine two Schnorr signatures [46] sequentially. Similar constructions are known in the literature [22]. Using the DL assumption in a group $\mathbb{G}$, we can formally prove the security of the above IBS scheme, that is, existential unforgeability against adaptive chosen message and identity attacks in the random oracle model. A proof idea is actually similar to that of [22] with a slight modification on the so-called Forking Lemma [42]. For more details, refer to the full version of this paper. Our IBS consists of one group element and two hash outputs, while the IBS of [22] consists of two group elements and one hash output. Since the size of a hash output is smaller than that of an group element, our scheme gives a shorter IBS.

5 Our IBS-PAKE Protocols

In this section, we present IBS-PAKE protocols, i.e., AKE protocols using a password and an IBS as authentication means. First we present two generic methods to construct IBS-PAKE protocols. We then present two concrete IBS-PAKE protocols using DL parameters.

5.1 Generic Construction

Our generic methods are presented by using two-party PAKE and an IBS in a modular way. The first generic method gives resilience to server compromise. That is, even if a password file is revealed from a server compromised, an adversary can only obtain password verification information, not a real password. Thus, to impersonate a client, he or she must mount an offline dictionary attack additionally. In the protocol, an IBS scheme is used for both of a client and a server. A server uses an IBS to authenticate itself to a client. A client uses an IBS to prove the possession of his or her own password.

Let PAKE denote a two-party PAKE protocol. For example, we can consider EKE [7], PAK [8], SRP, SPEKE, and AMP [32].

Assume that an IBS scheme, $\mathsf{IBS} = (\mathsf{Setup}, \mathsf{KeyExt}, \mathsf{Sign}, \mathsf{Vrfy})$ and a symmetric encryption scheme, $SE = (\mathcal{E}, \mathcal{D})$ are given. A client or a server obtains a signing key from KGA running KeyExt in the initialization phase. For distinction, a set of public parameters of IBS is denoted by pp_{IBS}.

The protocol consists of two phases, initialization and key establishment as follows.

Initialization Phase. Three processes Setup, $\mathsf{Extract}$, and $\mathsf{Registration}$ are executed as follows. Let $pw_U \in \mathcal{D}_{PW}$ denote a password chosen by a user, U.

- Setup: For a given security parameter λ, it generates pp_{PAKE} for the given PAKE protocol. It also generates independent cryptographic hash functions, $H_i : \{0,1\}^* \rightarrow \{0,1\}^{\ell_i}$ for $i = 1, 2, 3$. It runs Setup of the IBS scheme to generate $(\mathsf{msk}, pp_{\mathsf{IBS}})$. It outputs the system public parameters, $pp = (pp_{\mathsf{PAKE}}, pp_{\mathsf{IBS}}, H_{i=1,2,3}, SE)$. The master secret key, msk is kept secret.
- $\mathsf{Extract}$. For a given identity ID, KeyExt is run (by KGA) to output a private key, sk_{ID}. Assume that the private key is transmitted to the user ID via a secure channel.
- $\mathsf{Registration}$. A client, C generates a password, pw_C according to a pre-defined password creation policy. Let sk_{ID_C} be a signing key of C. Assume that a secure channel is established between the client and a server, S. To register a service, C sends $(\mathtt{Register\text{-}Request}, ID_C, \pi_1 = H_1(pw_C), ESK = \mathcal{E}_{H_2(pw_C)}(sk_{ID_C}))$ to the server via a secure channel.[4]

Key Establishment Phase. A client, C and a server, S execute a run of the protocol to agree on a temporal key to be used for a session as follows. See Fig. 1.

1. PAKE-**Client.** The client C computes $\pi_1 = H_1(pw_C)$. Using π_1 instead of pw_C, the client performs its part in PAKE with the following modification: Whenever the client receives a pair of a message and a signature, $(\overline{m}_S, \sigma_S)$ from the server S, the client verifies the signature, σ_S on $\overline{m}_S$, that is, checks if $1 = \mathsf{Vrfy}(pp_{\mathsf{IBS}}, ID_S, \sigma_S, \overline{m}_S)$. If the signature is valid then C performs its part of PAKE for $\overline{m}_S$. Finally, C obtains a common key K.

[4] Note that a secure channel is needed because $\pi_1 = H_1(pw_C)$ or $ESK = \mathcal{E}_{H_2(pw_C)}(sk_{ID_C})$ can be used to mount off-line dictionary attacks by an adversary.

Client C		Server S
$pp = \{pp_{\mathsf{PAKE}}, pp_{\mathsf{IBS}}, H_{i=1,2,3}, SE\}$ $[ID_C, pw_C]$		$pp = \{pp_{\mathsf{PAKE}}, pp_{\mathsf{IBS}}, H_{i=1,2,3}, SE\}$ $[ID_S, sk_{ID_S}]$ $\pi_S[C] = (ID_C, \pi_1 = H_1(pw_C),$ $ESK = \mathcal{E}_{H_2(pw_C)}(sk_{ID_C}))$
Using $\pi_1 = H_1(pw_C)$ instead of pw_C, perform its part in PAKE with the following modification: Whenever $(\overline{m}_S, \overline{\sigma}_S)$ is received, if $0 = \mathsf{Vrfy}(pp_{\mathsf{IBS}}, ID_S, \overline{\sigma}_S, \overline{m}_S)$, abort. Otherwise, perform the client's part for given $\overline{m}_S$ in PAKE.	Modified Execution of PAKE $\xrightarrow{\text{with } H_1(pw_C)}$ $\longleftarrow$	perform its part in PAKE with the following modification: For each $\overline{m}_S$ to be sent to C in PAKE, $\overline{\sigma}_S \leftarrow \mathsf{Sign}(pp_{\mathsf{IBS}}, ID_S, sk_{ID_S}, \overline{m}_S)$, and then send $(\overline{m}_S, \overline{\sigma}_S)$.
Output K $ek = H_3(K)$, $ESK\|\|\sigma_S = \mathcal{D}_{ek}(CT_S)$ $M_S = ID_S\|\|\mathcal{T}_{\mathsf{PAKE}}\|\|ESK$ If $0 = \mathsf{Vrfy}(pp_{\mathsf{IBS}}, ID_S, \sigma_S, M_S)$, abort. Otherwise, $sk_{ID_C} = \mathcal{D}_{H_2(pw_C)}(ESK)$ $\sigma_C \leftarrow \mathsf{Sign}(pp_{\mathsf{IBS}}, ID_C, sk_{ID_C}, M_C)$ where $M_C = ID_C\|\|\mathcal{T}_{\mathsf{PAKE}}\|\|CT_S$ $CT_C = \mathcal{E}_{ek}(\sigma_C)$	$\xleftarrow{ID_S, CT_S}$ $\xrightarrow{ID_C, CT_C}$	Output K $ek = H_3(K)$, $CT_S = \mathcal{E}_{ek}(ESK\|\|\sigma_S)$ $\sigma_S \leftarrow \mathsf{Sign}(pp_{\mathsf{IBS}}, ID_S, sk_{ID_S}, M_S)$, $M_S = ID_S\|\|\mathcal{T}_{\mathsf{PAKE}}\|\|ESK$ $\sigma_C = \mathcal{D}_{ek}(CT_C)$ $M_C = ID_C\|\|\mathcal{T}_{\mathsf{PAKE}}\|\|CT_S$ If $0 = \mathsf{Vrfy}(pp_{\mathsf{IBS}}, ID_C, \sigma_C, M_C)$, abort. Otherwise,
$\mathsf{pid}_C = ID_C\|\|ID_S$ $\mathsf{sid}_C = \mathcal{T}_{\mathsf{PAKE}}\|\|ID_S\|\|CT_S\|\|ID_C\|\|CT_C$ $ssk = H_3(\mathsf{pid}_C\|\|\mathsf{sid}_C\|\|K)$		$\mathsf{pid}_S = ID_C\|\|ID_S$ $\mathsf{sid}_S = \mathcal{T}_{\mathsf{PAKE}}\|\|ID_S\|\|CT_S\|\|ID_C\|\|CT_C$ $ssk = H_3(\mathsf{pid}_S\|\|\mathsf{sid}_S\|\|K)$

Fig. 1. Generic construction of an IBS-PAKE protocol.

2. **PAKE-Server.** The server S performs its part in PAKE with the following modification: For each $\overline{m}_S$ to be sent to C in PAKE, S generates $\sigma_S \leftarrow \mathsf{Sign}(pp_{\mathsf{IBS}}, ID_S, sk_{ID_S}, \overline{m}_S)$, and then sends $(\overline{m}_S, \sigma_S)$. Finally, S obtains a common key K.
3. **Server.** Let $\mathcal{T}_{\mathsf{PAKE}}$ denote a concatenation of all transcripts generated from a run of PAKE. S generates $\sigma_S \leftarrow \mathsf{Sign}(pp_{\mathsf{IBS}}, ID_S, sk_{ID_S}, M_S)$ for $M_S = ID_S||\mathcal{T}_{\mathsf{PAKE}}||ESK$. It computes $ek = H_3(K)$ and $CT_S = \mathcal{E}_{ek}(ESK||\sigma_S)$, and then sends (ID_S, CT_S).
4. **Client.** Upon receiving $[ID_S,\ CT_S]$, the client C computes $ek = H_3(K)$ and obtains $ESK||\sigma_S = \mathcal{D}_{ek}(CT_S)$. Then C checks if σ_S is valid, i.e., $1 = \mathsf{Vrfy}(pp_{\mathsf{IBS}}, ID_S, \sigma_S, M_S)$ for $M_S = ID_S||\mathcal{T}_{\mathsf{PAKE}}||ESK$. If the validity does not hold then the session is aborted. Otherwise, C computes $\pi_2 = H_2(pw_C)$ and $sk_{ID_C} = \mathcal{D}_{\pi_2}(ESK)$. Using sk_{ID_C}, the client generates a signature, $\sigma_C \leftarrow \mathsf{Sign}(pp_{\mathsf{IBS}}, ID_C, sk_{ID_C}, M_C)$ on $M_C = ID_C||\mathcal{T}_{\mathsf{PAKE}}||CT_S$. Then the client generates $CT_C = \mathcal{E}_{ek}(\sigma_C)$ and sends $[ID_C,\ CT_C]$ to S.

Finally, the client computes a secret session key, $ssk = H_3(\mathsf{pid}_C||\mathsf{sid}_C||K)$ where $\mathsf{pid}_C = ID_C||ID_S$ and $\mathsf{sid}_C = \mathcal{T}_{\mathsf{PAKE}}||ID_S||CT_S||ID_C||CT_C$.

5. **Server.** Upon receiving $[ID_C, CT_C]$, the server computes $\sigma_C = \mathcal{D}_{ek}(CT_C)$ and checks if σ_C is valid, i.e., $1 = \mathsf{Vrfy}(pp_{\mathsf{IBS}}, ID_C, \sigma_C, M_C)$ for $M_C = ID_C ||\mathcal{T}_{\mathsf{PAKE}}||CT_S$. If it is not valid then the session is aborted. Otherwise, C computes a secret session key $ssk = H_3(\mathsf{pid}_S||\mathsf{sid}_S||K)$ where $\mathsf{pid}_S = ID_C||ID_S$ and $\mathsf{sid}_S = \mathcal{T}_{\mathsf{PAKE}}||ID_S||CT_S||ID_C||CT_C$.

In the above construction, different IBS schemes can be used for participants, to gain advantages. Assume that an IBS scheme has an efficient verifying algorithm and another IBS scheme has an efficient signing algorithm. A client's performance can be significantly improved if a server and a client use the first and the second IBS schemes, respectively.

The second generic method is a simplified version of the first method to omit executing a knowledge proof that a client is aware of the original password. In certain applications, a server can be managed systematically and sufficiently protected from a well-organized security architecture. For the situation, we can relax the security requirement on server compromise. An IBS-PAKE protocol is constructed by eliminating the third flow of the first method. The resulting protocol reduces the client's computation significantly. See Fig. 3.

In the above generic methods, a server authenticates himself to a client using two factors, i.e., a password of low entropy and an IBS of high entropy.

5.2 Instances

As instances resulting from the generic methods, we present two IBS-PAKE protocols using PAK [8] and the IBS scheme in Sect. 4. The instances are constructed not exactly following the generic methods but with a modification where a server use a single authentication factor, i.e., an IBS.[5] The first protocol is, for short called PWIBS-AKE. Each phase of PWIBS-AKE is given as follows.

Initialization Phase. Three processes Setup, Extract, and Registration are executed as follows. Let $pw_C \in \mathcal{D}_{PW}$ denote a password chosen by a client, C.

- Setup(λ): For a given security parameter λ, it generates a cyclic group, $\mathbb{G}$ of prime order q and two random generators, g and g_1 of $\mathbb{G}$. It generates $\theta \in \mathbb{Z}_q^*$ uniformly at random and computes $u = g^\theta$. It also generates independent cryptographic hash functions, $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$, $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_i : \{0,1\}^* \rightarrow \{0,1\}^{\ell_i}$ for $i = 2, 3$. It outputs public parameters $pp = (\mathbb{G}, g, g_1, u, H, H_{i=1,2,3})$ and the corresponding master secret key $\mathsf{msk} = \theta$.
- Extract(msk=θ, ID). For a given identity ID, it picks $r \in \mathbb{Z}_q^*$ uniformly at random. It computes $R = g^r$, $w = H(ID, R)$, and $v = r + \theta w \pmod q$. It then returns $sk_{ID} = (R, v)$. Assume that the private key is transmitted via a secure channel.

[5] It is not difficult to fix the instances to follow the generic methods.

– $\mathsf{Registration}(C, S)$. First, a client, C generates its password, pw_C according to a pre-defined password creation policy. Also, C obtains a signing key, $sk_{ID_C} = (v_C, R_C)$ from $\mathsf{Extract}$. Assume that a secure channel is established in advance between C and S. To register a service, C sends ($\mathtt{Register\text{-}Req}$, ID_C, $g_1^{-H_1(pw_C)}$, $\mathcal{E}_{H_2(pw_C)}(v_C), R_C$) to the server, S over the secure channel. The server appends $\pi_S[C] = (ID_C, g_1^{-H_1(pw_C)}, \mathcal{E}_{H_2(pw_C)}(v_C), R_C)$ to $\mathcal{PF}$.

Key Establishment Phase. A client, C and a server, S execute a run of $\mathsf{PWIBS\text{-}AKE}$ to agree on a temporal session key. The concrete protocol is described as follows (See Fig. 2).

1. C picks $x \in \mathbb{Z}_q^*$ uniformly at random and computes $W = g^x g_1^{H_1(pw_C)} \in \mathbb{G}$ using the password, pw_C. Then, C sends $[ID_C, W]$ to S.
2. Upon receiving $[ID_C, W]$, S picks $y \in \mathbb{Z}_q^*$ uniformly at random and computes $Y = g^y \in \mathbb{G}$, and also $X' = W g_1^{-H_1(pw_C)}$ and $K' = (X')^y$. It finds $\pi_S[C]$ corresponding to ID_C, i.e., $[ID_C, g_1^{-H_1(pw_C)}, ESK = \mathcal{E}_{H_2(pw_C)}(v_C), R_C]$ from a database. Using its signing key, $sk_{ID_S} = (R_S, v_S)$, the server generates a signature, $\sigma_S = (d_S, z_S, R_S)$ on $M_S = ID_S||W||Y||ESK$, where $E_S = g^{e_S}$, $z_S = H(M_S, ID_S, E_S)$ and $d_S = e_S - v_S z_S \pmod q$ for random $r_S, e_S \in \mathbb{Z}_q^*$. Also, using $ek = H_1(K')$ as an encryption key, S generates a ciphertext, $CT_S = \mathcal{E}_{ek}(ESK||\sigma_S)$. Then S sends $[ID_S,\ Y,\ CT_S]$ to C.
3. Upon receiving $[ID_S,\ Y,\ CT_S]$, the client C computes $K = Y^x$ and $ek = H_1(K)$, and $ESK||\sigma_S = \mathcal{D}_{ek}(CT_S)$. It checks if the signature, σ_S is valid, i.e., the equality of $z_S = H(M_S, ID_S, g^{d_S} \cdot (R_S^{z_S} \cdot u^{w_S \cdot z_S}))$ holds for $M_S = ID_S||W||Y||ESK$ and $w_S = H(ID_S, R_S)$. If the validity does not hold then the session is aborted. Otherwise, the client computes $\pi_1 = H_2(pw_C)$ and decrypts ESK to obtain $v_C = \mathcal{D}_{\pi_1}(\text{ESK})$. Using v_C, the client generates a signature share, $\sigma_C = (d_C, z_C)$ where $z_C = H(M_C, ID_C, E_C)$ and $d_C = e_C - v_C z_C \pmod q$ for random $r_C, e_C \in \mathbb{Z}_q^*$, $M_C = ID_C||W||Y||CT_S$. Let $\sigma'_C = (z_C, d_C)$. The client computes $CT_C = \mathcal{E}_{ek}(\sigma'_C)$. Finally, the client sends $[ID_C, CT_C]$ to S.
   Then the client computes a secret session key, $ssk = H_3(\mathsf{pid}||\mathsf{sid}_C||K)$ where $\mathsf{pid}_C{=}ID_C||ID_S$ and $\mathsf{sid}_C = ID_C||W||Y||CT_S||CT_C$.
4. Upon receiving $[ID_C,\ CT_C]$, the server decrypts CT_C to obtain $(z_C, d_C) = \sigma'_C = \mathcal{D}_{ek}(CT_C)$ and checks if the signature is valid, i.e., the equality of $z_C = H(M_C, ID_C, g^{d_C}(R_C u^{w_C})^{z_C})$ holds. Here R_C is the value stored at the database, and $M_C = ID_C||W||Y||CT_S$ and $w_C = H(ID_C, R_C)$. If the validity does not hold then the session is aborted. Otherwise, the server computes a secret session key, $ssk = H_3(\mathsf{pid}||\mathsf{sid}_S||K')$ where $\mathsf{pid}_S{=}ID_C||ID_S$ and $\mathsf{sid}_S = ID_C||W||Y||CT_S||CT_C$.

At Step 3, $\sigma'_C = (z_C, d_C)$ is generated by the client. It does not consist of a full IBS because R_C is not given, and thus nobody can check its validity. Instead of an encryption of σ'_C, we can send σ'_C to the server. However, since R_C is stored at the server, the server is able to verify it. Note that R_C is a global value included in all signatures generated by a client. This modification will alleviate the computation and communication cost on the client side.

A simplified IBS-PAKE protocol can be constructed from PWIBS-AKE by omitting a knowledge proof for an original password. The resulting protocol can run in two independent passes. For more details, refer to the appendix.

5.3 Security Proofs

In this section we prove the security of the proposed protocols in the model of Sect. 3. We prove that our first protocol provides AKE security and resilience to server compromise. That is, an adversary attacking the protocol cannot obtain useful information about session keys of fresh and semi-fresh instances with greater advantages than that of an on-line dictionary attack.

Theorem 1. *Assume that the IBS scheme, DL-IBS is used for PWIBS-AKE. Also, assume that the CDH assumption holds in $\mathbb{G}$. The proposed AKE protocol, PWIBS-AKE is AKE-secure in the random oracle model under the security model of Sect. 3.*

Proof. In the proof we consider a series of protocols, P_i $(i = 0, 1, .., 7)$ which are modified sequentially from the original protocol, P_0 = PWIBS-AKE. For each modification, we shall show that the advantage of an adversary increases with a negligible fraction. In the final protocol, P_7, the adversary will be able to get only an advantage from an on-line-guessing attack.

Assume that an adversary $\mathcal{A}$ can make at most q_{ex} and q_s queries to the Execute and the Send oracles. In the original protocol, P_0, we consider the random oracle model where hash functions are considered random functions. For each new hash query, a fresh random output is returned. To make consistent responses to hash queries, lists L_H and $L_{H_{i=1,2,3}}$ are maintained.

- Hash query. On a $\mathcal{H}(m)$-query for $\mathcal{H} = H, H_i$, returns h as follows. Let $h = \rho$ if (m, ρ) exists in $L_{\mathcal{H}}$, and otherwise, let $h = \rho' \overset{R}{\leftarrow} D$ where D is the domain of $\mathcal{H}$. $L_{\mathcal{H}}$ is updated with (m, ρ').

Next we describe P_i for $i = 0, 1, ..., 7$ concretely.

P_0: It is the original protocol, PWIBS-AKE defined in Subsect. 5.2 under the random oracle model.

P_1: It is modified from P_0 as follows. Let Rep denote the event that honest parties do not generate $W = g^x g_1^{H_1(pw_C)}$ or $Y = g^y$ twice. In P_1, we assume that Rep does not occur. Let q be the order of the group $\mathbb{G}$. We have $\Pr[\texttt{Rep}] \leq \frac{(q_s+q_{ex})^2}{q}$ by using a similar analysis of [37]. It is negligible because q is assumed to be sufficiently large. P_0 and P_1 are indistinguishable except the negligible probability.

P_2: It is modified from P_1 as follows. In P_2, we assume that the adversary cannot generate a valid server's signature without making a Corrupt query to a server. As shown in Sect. 4, the given IBS scheme is existentially unforgeable. It is obvious that by this assumption, P_1 and P_2 are indistinguishable except a negligible probability from existential unforgeability, i.e., $\mathsf{Adv}_{\mathcal{A},\mathsf{DL\text{-}IBS}}^{EUF\text{-}CMIDA}$. An adversary is able to use only (ID_S, Y, CT_S) which has been generated by a server, in order to make a Send query to a client instance.

In the protocol, a password-based authentication and an IBS works independently. Even if a password is revealed on a client side, a server's IBS key cannot be compromised.

P_3: It is modified from P_2 as follows. When an H_3 query is issued, P_3 does not check consistency against Execute queries, but returns a random output. The response to an Execute query is a collection of transcripts generated from an honest execution of the protocol. That is, it has the form, $[(ID_C, W)$, (ID, Y, CT_S), $(ID_C, CT_C)]$, where $CT_S = \mathcal{E}_{ek}(ESK||\sigma_S)$, $CT_C = \mathcal{E}_{ek}(\sigma'_C)$ and $ek = H_3(g^{ab})$.

The way that an adversary know the inconsistency can be used to solve a CDH problem as follows. For a given CDH problem $(A = g^a, B = g^b)$, we plug in A and B for X and Y, respectively. We have $ek = H_3(g^{ab})$. The adversary would have made a H_3 query with g^{ab} to distinguish the distribution of ek. We can get g^{ab} for the solution to the CDH problem.

In other words, P_2 and P_3 are indistinguishable except the negligible probability to solve the CDH problem. Since a random output is used as an encryption key, ek, the ciphertext $\mathcal{E}_{ek}(\cdot)$ looks random from a viewpoint of an adversary. Also, in the above case, a session key is defined by a random value because it is an output of H_3. Hence, if Test query is asked to an instance which was initialized via an Execute query, $\mathsf{Adv}_{\mathcal{A},\mathsf{PWIBS\text{-}AKE}}^{IBS\text{-}PAKE}$ is upper bounded by a negligible probability.

P_4: It is modified from P_3 as follows. We assume that P_4 halts if a correct guess for a password is made against a client instance or a server instance before a Corrupt query. We can determine whether a password is correctly guessed or not, by an H_3 query using a correct input to compute ek and ssk. In this case, P_3 and P_4 are identical.

In the case of semi-freshness, the following assumption is added. If a correct password guess is made against a server instance before a Corrupt query to a client instance, P_4 halts. We can determine whether a password is correctly guessed or not, by an $H_{i=1,2}$ query with the correct password and a Corrupt query to a server. P_4 is identical to P_3 except that off-line dictionary attacks occurs.

P_5: It is modified from P_4 as follows. We assume that the adversary cannot make a password guess against client and server instances which are partnered. To argue the impossibility, we show that the capability to make a password guess can be used to solve a CDH problem. Let (ID_C, W) be the first protocol transcript which is generated via a Send query to a client instance or directly given by the adversary. Let $W = g^{\alpha}$. Then, to a Send query with $(ID_C, W = g^{\alpha})$ to a server instance, the response, which is also the second transcript, (ID, Y, CT_S) is generated as follows. Let $(g, A = g^a, B = g^b)$ be a given CDH problem. We plug in A for g_1 and B for Y and returns a random value for ek. Define $ek = H_3(g^{\alpha b} g_1^{-\gamma b})$ where $\gamma = H_2(pw_C)$. Let $CT_S = \mathcal{E}_{ek}(ESK||\sigma_S)$. We have $g_1^{-\gamma y} = (g^{ab})^{-\gamma}$. To guess a password, the adversary must know g^{ab} which is the CDH solution. Hence, a correct password guess from client and server instances which are partnered, is impossible under the CDH assumption.

P_6: It is modified from P_5 as follows. We assume that the adversary cannot generate a client's valid signature share, (z_C, d_C) without making H_2 query with the correct password and a Corrupt query to a server. It is easy to see that the adversary get no useful information about the secret key. Note that $R_C = g^{r_C}$ for $r_C \in \mathbb{Z}_q^*$ is used as a signature part but is stored by a server and so the adversary cannot be aware of it. Thus, the probability that the adversary generates (z_C, d_C) verified with R_C is upper bounded by $1/q$.

P_7: It is modified from P_6 as follows. We assume that a password guess can be checked by a *password* oracle, that is, whether it is correct or not. Let cguess denote an event that the adversary guesses a password correctly. P_7 accepts a Corrupt(U) query and returns $\mathcal{PF}_S = \{\pi_S[C]\}_{C \in \textbf{Clients}}$ if $U \in$ **Servers**, and otherwise pw_U. For freshness, there are at most q_s queries before a Corrupt query. Thus, we have $\Pr[\mathsf{cguess}] \leq \frac{q_s}{|\mathcal{D}_{\mathcal{PW}}|}$. For semifreshness, $q_{H_1} + q_{H_2}$ more queries are considered before a Corrupt query to a client. Here, let q_{H_i} denote the maximum number of H_i queries. Since these occur if there has been a Corrupt query to a server, we have $\Pr[\mathsf{cguess}] \leq \frac{q_s + q_{H_1} + q_{H_2}}{|\mathcal{D}_{\mathcal{PW}}|}$.

Overall, the success probability of the adversary can be evaluated by expanding with the event of cguess. That is, we have $\Pr[\mathsf{Succ}_{P_7}] = \Pr[\mathsf{cguess}] + \Pr[\mathsf{Succ}_{P_7}|\overline{\mathsf{cguess}}]$ where $\overline{\mathsf{cguess}}$ denotes the negation of cguess. As we shown in a series of the protocol modification above, $\Pr[\mathsf{Succ}_{P_7}|\overline{\mathsf{cguess}}]$ can be upper bounded by the negligible probability under the CDH assumption and existential unforgeability of the given IBS scheme. Therefore we obtain the desired results.

Based on the security of a given PAKE and an IBS scheme, we can prove that the first modular method (defined in Fig. 1) to yield an IBS-PAKE protocol is AKE secure and resilient to server compromise. That is, an adversary attacking the protocol cannot obtain useful information about session keys of fresh and semi-fresh instances with greater advantages than that of an on-line dictionary attack. The security proof for the modular method can be completed by following the security proof of PAKE with consideration for unforgeability of the underlying IBS scheme. As in the PWIBS-AKE, a client's signature, σ_C is encrypted and so not revealed to an adversary in the modular method. An adversary gains no meaningful advantage for a password guess from $CT_C = \mathcal{E}_{ek}(\sigma_C)$.

Also, a similar proof idea can be applied with a slight modification, to prove the AKE security of the simplified PWIBS-AKE and AKE protocols generated from our second modular method (defined in Fig. 3), equivalently that an adversary attacking the protocol cannot obtain useful information about session keys of fresh instances with greater advantages than that of an on-line dictionary attack. Actually, the proofs can be completed by simplifying the security proof of Theorem 1, i.e., P_4 because the protocols do not consider semi-fresh to capture 'server compromise'.

6 Performance Analysis

In this section, we compare performance between our protocols and other known AKE protocols using a combination of a password and an asymmetric techniques [53]. We also give experimental results for our protocols.

6.1 Performance Comparison

The performance is analyzed in terms of communication and computation overhead. Our protocols work with discrete logarithm parameters. Let $\mathbb{G}$ be a group of prime order q. Let ℓ_q and $\ell_{\mathbb{G}}$ denote the bit-length of the order of $\mathbb{G}$ and an element of $\mathbb{G}$, respectively. Let ℓ_H denote the bit-length of a hash output. Let $\mathtt{Exp}_t$ denote simultaneous multi-exponentiation (or scalar multiplication) using t group elements. In the communication analysis, we exclude identifiers commonly required for every protocol.

In PWIBS-AKE, a client transmits ID_C, W in the first round, and ID_C, $CT_C = \mathcal{E}_{ek}(\sigma'_C)$ in the third round where $\sigma'_C = (z_C, d_C)$. Since W is an element of $\mathbb{G}$, and z_C and d_C are elements of $\mathbb{Z}_q$, a client transmits a $(\ell_{\mathbb{G}} + 2\ell_q)$-bit string. A client executes two $\mathtt{Exp}_1$, one $\mathtt{Exp}_2$, and one $\mathtt{Exp}_3$ and two decryption, i.e., $\mathcal{D}$ of a symmetric encryption scheme. A server transmits $ID_S, Y, CT_S = \mathcal{E}_{ek}(ESK||\sigma_S)$ where $\sigma_S = (z_S, d_S, R_S)$ and ESK is a ciphertext of v_C, i.e., $ESK = \mathcal{E}_{H_1(pw_C)}(v_C)$. Since v_C is an elements of $\mathbb{Z}_q$, we can assume that the bit length of CT_S is $2\ell_{\mathbb{G}} + 3\ell_q$. Note that W and R_S are elements of $\mathbb{G}$, and z_S and d_S are elements of $\mathbb{Z}_q$. Thus a server must transmit a $(2\ell_{\mathbb{G}} + 3\ell_q)$-bit string. Also, a server executes three $\mathtt{Exp}_1$, one $\mathtt{Exp}_3$ and one encryption, i.e., $\mathcal{E}$ of a symmetric encryption scheme. When an elliptic curve group with $\ell_p = 192$ and $\ell_{\mathbb{G}} = 192$ is considered, a client's transcript length is about 576 bits or 72 bytes. In the simplified PWIBS-AKE, reduced computation and smaller transcripts are required for a client and a server.

In Table 1, we summarizes comparison results among PAKE, IBE-PAKE, and our IBS-PAKE protocols, PWIBS-AKE and the simplified PWIBS-AKE (denoted by Sim-PWIBS in the table). For PAKE, we consider SPEKE [28,30], J-PAKE [1,27], SRP6 [51], AMP [34], and SK [49] which are presented in ISO/IEC 11770-4 [32] or IEEE P1363.2 [31]. For IBE-PAKE, we consider the protocols of [53] constructed from two different IBE schemes, i.e., the pairing-based Boneh-Franklin (BF) IBE [4] and TDL-based IBE [43] with the CCA-security. In the table,

Table 1. Comparison of AKE protocols

Protocol		RSI	RSC	Round (Pass)	Client		Server	
					Comm.	Comp.	Comm.	Comp.
PAKE (ISO/IEC) [32]	SPEKE	X	X	1(2)	$\ell_{\mathbb{G}}$	$2\mathtt{Exp}_1$	$\ell_{\mathbb{G}}$	$2\mathtt{Exp}_1$
	J-PAKE	X	X	2(4)	$6\ell_{\mathbb{G}}+3\ell_H$	$6\mathtt{Exp}_1+4\mathtt{Exp}_2$	$6\ell_{\mathbb{G}}+3\ell_H$	$6\mathtt{Exp}_1+4\mathtt{Exp}_2$
	SRP6	X	O	2(4)	$1\ell_{\mathbb{G}''}+1\ell_H$	$2\mathtt{Exp}_1$	$1\ell_{\mathbb{G}''}+1\ell_H$	$1\mathtt{Exp}_1+1\mathtt{Exp}_2$
	AMP	X	O	2(4)	$1\ell_{\mathbb{G}}+1\ell_H$	$2\mathtt{Exp}_1$	$1\ell_{\mathbb{G}}+1\ell_H$	$2\mathtt{Exp}_2$
	SK	X	O	2(4)	$1\ell_{\mathbb{G}}+1\ell_H$	$2\mathtt{Exp}_1$	$1\ell_{\mathbb{G}}+1\ell_H$	$1\mathtt{Exp}_1+1\mathtt{Exp}_2$
IBE-PAKE [53]	w/BF [4]	O	X	2(2)	$2\ell_{\mathbb{G}_1}+2\ell_H$	$1\mathtt{P}+6\mathtt{Exp}$	$2\ell_{\mathbb{G}_1}$	$1\mathtt{P}+4\mathtt{Exp}$
	w/TDL [43]	O	X	2(2)	$2\ell_{\mathbb{G}'}+2\ell_H$	$6\mathtt{Exp}$	$2\ell_{\mathbb{G}'}$	$5\mathtt{Exp}$
IBS-PAKE	PWIBS-AKE	O	O	3(3)	$1\ell_{\mathbb{G}}+2\ell_q$	$2\mathtt{Exp}_1+\mathtt{Exp}_2+\mathtt{Exp}_3+2\mathcal{D}+\mathcal{E}$	$2\ell_{\mathbb{G}}+3\ell_q$	$3\mathtt{Exp}_1+\mathtt{Exp}_3+\mathcal{D}+\mathcal{E}$
	Sim-PWIBS	O	X	1(2)	$1\ell_{\mathbb{G}}$	$\mathtt{Exp}_1+\mathtt{Exp}_2+\mathtt{Exp}_3$	$2\ell_{\mathbb{G}}+2\ell_q$	$3\mathtt{Exp}_1$

let 'RSI' and 'RSC' denote 'Robustness to Server Impersonation (when a password is revealed)' and 'Resilience to Server Compromise', respectively. Also, let 'Comm.' and 'Comp.' denote the communication length and the computation cost, respectively. Let $\mathbb{G}'$ and $\mathbb{G}''$ be a TDL group defined with RSA parameters and a multiplicative group of a finite field, respectively. Thus $\ell_{\mathbb{G}'}$ or $\ell_{\mathbb{G}''}$ should be larger than at least 1024. '$\mathtt{P}$' denotes a pairing operation and let $\mathbb{G}_1$ be a bilinear group. It is known that a pairing operation is more expensive than (or comparable to) an exponentiation or a scalar multiplication when a similar security level is assumed [5,6,20]. Our protocols can be efficiently performed without requiring any pairing operation. Similarly, our generic construction can be efficiently performed.

6.2 Experimental Results

The test for our experimental results has been performed on an Intel Pentium model CPU clocked at 2.40GHz. The algorithms were written in Python 2.7 and based on Charm-Crypto [14] and PyCrypto [41] libraries.[6] Each result is the average of 1,000 tests.

For a mathematical group in the protocol, we use four elliptic curve groups, 'prime192v1', 'sect193r1', 'secp224r1', and 'sect163k1'. 'prime192v1' represents NIST/X9.62/SECG curve over a 192 bit prime field [48]. 'sect193r1' represents SECG curve over a 193 bit binary field [47]. 'secp224r1' represents NIST/SECG curve over a 224 bit prime field [48]. 'sect163k1' represents NIST/SECG/WTLS curve over a 163 bit binary field [48]. As a symmetric encryption scheme we use AES with the CBC mode. The bit sizes of a key used for AES are 128 and 256.

Table 2 shows the running time of PWIBS-AKE. According to the distinct tasks by a communication round, the protocol can be divided into four submodules, Client.s1, Server.s2, Client.s3, and Server.s4. Client.s1 represents the

[6] Even though Charm is not optimised, our results are enough to show feasible and efficient implementation of our protocols.

Table 2. Experimental results of PWIBS-AKE (time:msec)

EC Group	AES key(bit)	Client.s1	Server.s2	Client.s3	Server.s4	Total
prime192v1	256	1.28	2.21	3.80	2.24	9.53
	128	1.30	2.25	3.90	2.27	9.66
sect193r1	256	2.28	4.01	6.58	3.83	16.7
	128	2.51	4.21	6.86	4.01	17.59
secp224r1	256	1.64	3.01	5.03	2.92	12.6
	128	1.64	3.01	5.01	2.90	12.56
sect163k1	256	1.85	2.87	4.84	2.87	12.43
	128	1.74	2.71	4.54	2.71	11.7

generation of $W = g^x g_1^{H_1(pw_C)}$ by a client. Server.s2 represents the generation of $(Y = g^y, \sigma_S = (z_S, d_S, R_S))$ and $CT_S = \mathcal{E}_{ek}(ESK||\sigma_S)$ by a server. Client.s3 represents the verification of σ_S, the computation of $K = Y^x$, the generation of $\sigma_C = (z_C, d_C)$, and the computation of a session key by a client. Server.s4 represents the verification of $\sigma_C = (z_C, d_C)$ and the computation of a session key by a server.

Table 3 shows the running time of the simplified PWIBS-AKE. In the experiment, the protocol is divided into four submodules, Client.s1, Server.s2, Client.s3, and Server.s4. As a client does not generate a signature for the possession proof of a password, Client.s3 of the simplified PWIBS-AKE is faster than that of PWIBS-AKE.

In both of the tables, 'Total' represents the sum of running time of all submodules. As shown in the tables, the protocols give different experimental results according to elliptic curve groups used. The most time-consuming task occurs in Client.s3. However, the total running time is only at most 0.02 s.

7 Conclusion

We have proposed efficient AKE protocols based on a password and an IBS. A client is able to do an easy authentication using a human-memorable password and an ID-based signature as authentication means. The use of an IBS gives two security enhancements against party compromise, i.e., resistance to sever impersonation attacks from client compromise and resilience to client impersonation attacks from server compromise. The proposed protocols also give good performance compared to known AKE protocols. They can be applied for various applications.

A Bilinear Maps [21,39]

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be additive groups and $\mathbb{G}_T$ a multiplicative group. Assume that the groups have the same prime order, q. We say that e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an admissible bilinear map (or a pairing) if the following properties are satisfied: (1) Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_q^*$. (2) Non-degeneracy: There exist $P \in \mathbb{G}_1$ and $R \in \mathbb{G}_2$ such that $e(P, R) \neq 1$. (3) Computability: There exists an efficient algorithm to compute $e(P', Q')$ for all $P' \in \mathbb{G}_1$ and $Q' \in \mathbb{G}_2$.

Bilinear maps can be classified in three types, i.e., Type I, II, III according to the existence of morphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. Type I pairings, called '*symmetric*', have $\mathbb{G}_1 = \mathbb{G}_2$. Type II pairings have an efficiently computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$ but none in the reverse direction. Type III pairings have no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Type II and III pairings are called '*asymmetric*'. For more details, refer to [21,39].

B Computational Assumptions

Discrete Logarithm (DL) Assumption. Assume that a group $\mathbb{G}$ of order q and a generator g of $\mathbb{G}$ are given. To define *Discrete Logarithm (DL) problem*, we consider the following game:

$$\begin{gathered} \textit{Initialization}: x \overset{R}{\leftarrow} \mathbb{Z}_q,\ g_1 = g^x, \\ \textit{Output}: r \leftarrow \mathcal{A}(\mathbb{G},\ g,\ g_1 = g^x) \end{gathered}$$

We define $\mathsf{Adv}_{\mathcal{A},\mathbb{G}}^{DL}(t)= \Pr[r = x]$, where $\mathcal{A}$ runs in time t. We define that $\mathsf{Adv}_{\mathbb{G}}^{DL}(t)$ $= \max_{\mathcal{A}}[\mathsf{Adv}_{\mathbb{G}}^{DL}(t)]$ where the maximum is taken over all $\mathcal{A}$. We say that DL assumption holds for $\mathbb{G}$ if $\mathsf{Adv}_{\mathbb{G}}^{DL}(t)$ is negligible.

Decisional Diffie-Hellman (DDH) Assumption. Assume that a group $\mathbb{G}$ of order q and a generator g of $\mathbb{G}$ are given. To define *Decisional Diffie-Hellman (DDH) problem*, we consider the following distinguishability game:

$$\begin{gathered} \textit{Initialization}: x, y, r \overset{R}{\leftarrow} \mathbb{Z}_q,\ e_0 = xy,\ e_1 = r, \\ b \overset{R}{\leftarrow} \{0, 1\},\ h = g^{e_b} \\ \textit{Guess}: b' \leftarrow \mathcal{A}(\mathbb{G},\ g,\ g^x,\ g^y,\ h) \end{gathered}$$

We define $\mathsf{Adv}_{\mathcal{A},\mathbb{G}}^{DDH}(t)= |\Pr[b = b'] - 1/2|$, where $\mathcal{A}$ runs in time t. We define that $\mathsf{Adv}_{\mathbb{G}}^{DDH}(t) = \max_{\mathcal{A}}[\mathsf{Adv}_{\mathbb{G}}^{DDH}(t)]$ where the maximum is taken over all $\mathcal{A}$. We say that DDH assumption holds for $\mathbb{G}$ if $\mathsf{Adv}_{\mathbb{G}}^{DDH}(t)$ is negligible.

Computational Diffie-Hellman (CDH) Assumption. Assume that a group $\mathbb{G}$ of order q and a generator g of $\mathbb{G}$ are given. To define *Computational Diffie-Hellman (CDH) problem*, we consider the following game:

$$\begin{gathered} \textit{Initialization}: x, y \overset{R}{\leftarrow} \mathbb{Z}_q,\ g_1 = g^x,\ g_2 = g^y, \\ \textit{Output}: g^z \leftarrow \mathcal{A}(\mathbb{G},\ g,\ g^x,\ g^y) \end{gathered}$$

We define $\mathsf{Adv}_{\mathcal{A},\mathbb{G}}^{CDH}(t)= \Pr[z = xy]$, where $\mathcal{A}$ runs in time t. We define that $\mathsf{Adv}_{\mathbb{G}}^{CDH}(t) = \max_{\mathcal{A}}[\mathsf{Adv}_{\mathbb{G}}^{CDH}(t)]$ where the maximum is taken over all $\mathcal{A}$. We say that CDH assumption holds for $\mathbb{G}$ if $\mathsf{Adv}_{\mathbb{G}}^{CDH}(t)$ is negligible.

Table 3. Experimental results of our simplified PWIBS-AKE (time:msec)

EC Group	Client.s1	Server.s2	Client.s3	Server.s4	Total
prime192v1	1.33	1.52	3.01	0.84	6.7
sect193r1	2.36	2.58	5.11	1.36	11.41
secp224r1	1.66	2.01	3.82	1.09	8.58
sect163k1	1.75	1.76	3.53	0.92	7.96

C Simplified IBS-PAKE Protocols

Initialization Phase. Three processes Setup, Extract, and Registration are executed as follows.

- Setup and Extract are the same to those of PWIBS-AKE.
- Registration(C, S). First, a client, C generates his or her password, pw_C according to a pre-defined password creation policy. To register a service, C sends (`Register-Req`, ID_C, $g_1^{-H_1(pw_C)}$) to the server, S over a secure channel. The server appends $\pi_S[C] = (ID_C, g_1^{-H_1(pw_C)})$ to $\mathcal{PF}$.

Key Establishment Phase. A client, C and a server, S execute the protocol to agree on a temporal key to be used for a session. The concrete protocol is described as follows (See Fig. 4).

1. C picks $x \in \mathbb{Z}_q^*$ uniformly at random and computes $W = g^x g_1^{H_1(pw_C)} \in \mathbb{G}$ using the password, pw_C. Then, C sends $[ID_C, W]$ to S.
2. S picks $y \in \mathbb{Z}_q^*$ uniformly at random and computes $Y = g^y \in \mathbb{G}$. Also, using its signing key, $sk_{ID_S} = (R_S, v_S)$, the server generates a signature, $\sigma_S = (d_S, z_S, R_S)$ on $M_S = ID_S||Y$, where $E_S = g^{e_S}$, $z = H(m, ID_S, E_S)$ and $d_S = e_S - v_S z_S \pmod q$ for random $r_S, e_S \in \mathbb{Z}_q^*$. Then S sends $[ID_S, Y, \sigma_S]$ to C. From the receipt message $[ID_C, W]$, the server finds authentication information corresponding to ID_C, i.e., $[ID_C,\ g_1^{-H_1(pw_C)}]$ from a database. It then computes $X' = W g_1^{-H_1(pw_C)}$ and $K' = (X')^y$. Finally, S computes $ssk = H_3(\mathsf{pid}_S||\mathsf{sid}_S||K')$, where $\mathsf{sid}_S = ID_C||W||Y||\sigma_S$.
3. Upon receiving $[ID_S, Y, \sigma_S]$, the client C checks if the signature, σ_S is valid, i.e., the equality of $z_S = H(M_S, ID_S, g^{d_S} \cdot (R_S \cdot u^{w_S})^{z_S})$ holds. Here $M_S = ID_S||Y$ and $w_S = H(ID_S, R_S)$. If the validity does not hold then the session is aborted. Otherwise, the client computes $K = Y^x$. Finally, S computes $ssk = H_3(\mathsf{pid}_S||\mathsf{sid}_S||K')$, where $\mathsf{sid}_S = ID_C||W||Y||\sigma_S$.

Client C		Server S
$pp = \{\mathbb{G}, g, g_1, u, H, H_1, H_2, H_3\}$		$pp = \{\mathbb{G}, g, g_1, u, H, H_1, H_2, H_3\}$
$[ID_C, pw_C]$		$[ID_S, sk_{ID_S} = (v_S, R_S)]$
		$\pi_S[C] = (ID_C, g_1^{-H_1(pw_C)},$
		$ESK = \mathcal{E}_{H_2(pw_C)}(v_C), R_C))$
$\pi_1 = H_1(pw_C)$		
$x \overset{R}{\leftarrow} \mathbb{Z}_q^*,\ W = g^x g_1^{\pi_1}$	$\xrightarrow{ID_C, W}$	$y \overset{R}{\leftarrow} \mathbb{Z}_q^*,\ Y = g^y$
		$X' = W g_1^{-H_1(pw_C)},\ K' = (X')^y$
		$e_S \overset{R}{\leftarrow} \mathbb{Z}_q^*,\ E_S = g^{e_S}$
		$z_S = H(M_S, ID_S, E_S)$
		$d_S = e_S - v_S z_S \pmod q$
		where $M_S = ID_S \Vert W \Vert Y \Vert ESK$
		$\sigma_S = (z_S, d_S, R_S)$
$K = Y^x,\ ek = H_3(K)$	$\xleftarrow{ID_S, Y, CT_S}$	$ek = H_3(K'),\ CT_S = \mathcal{E}_{ek}(ESK \Vert \sigma_S)$
$ESK \Vert \sigma_S = \mathcal{D}_{ek}(CT_S)$		
$M_S = ID_S \Vert W \Vert Y \Vert ESK$		
$h_S = g^{d_S}(R_S u^{w_S})^{z_S}$		
If $z_S \neq H(M_S, ID_S, h_S)$, abort		
Otherwise, proceed as follows		
$v_C = \mathcal{D}_{H_2(pw_C)}(ESK)$		
$e_C \overset{R}{\leftarrow} \mathbb{Z}_q^*,\ E_C = g^{e_C}$		
$z_C = H(M_C, ID_C, E_C)$		
$d_C = e_C - v_C z_C \pmod q$		
where $M_C = ID_C \Vert W \Vert Y \Vert CT_S$		
$\sigma'_C = (z_C, d_C)$		
$CT_C = \mathcal{E}_{ek}(\sigma'_C)$	$\xrightarrow{ID_C, CT_C}$	$\sigma'_C = \mathcal{D}_{ek}(CT_C)$
		$M_C = ID_C \Vert W \Vert Y \Vert CT_S$
		$h_C = g^{d_C}(R_C u^{w_C})^{z_C}$
		If $z_C \neq H(M_C, ID_C, h_C)$, abort
		Otherwise, proceed as follows
$\mathsf{pid}_C = ID_C \Vert ID_S$		$\mathsf{pid}_S = ID_C \Vert ID_S$
$\mathsf{sid}_C = ID_C \Vert W \Vert Y \Vert CT_S \Vert CT_C$		$\mathsf{sid}_S = ID_C \Vert W \Vert Y \Vert CT_S \Vert CT_C$
$ssk = H_3(\mathsf{pid}_C \Vert \mathsf{sid}_C \Vert K)$		$ssk = H_3(\mathsf{pid}_S \Vert \mathsf{sid}_S \Vert K')$

Fig. 2. PWIBS-AKE: AKE from a combination of PAK and a Schnorr-based IBS

Client C		Server S
$pp = \{\mathbb{G}, g, pp_{\mathsf{IBS}}, H_{i=1,2,3}, SE\}$		$pp = \{\mathbb{G}, g, pp_{\mathsf{IBS}}, H_{i=1,2,3}, SE\}$
$[ID_C, pw_C]$		$[ID_S, sk_{ID_S}]$
		$\pi_S[C] = (ID_C, \pi_1 = H_1(pw_C))$
Using $\pi_1 = H_1(pw_C)$ instead of pw_C, perform its part in PAKE with the following modification: Whenever $(\overline{m}_S, \overline{\sigma}_S)$ is received, if $0 = \mathsf{Vrfy}(pp_{\mathsf{IBS}}, ID_S, \overline{\sigma}_S, \overline{m}_S)$, abort. Otherwise, perform the client's part for given $\overline{m}_S$ in PAKE.	Modified Execution of PAKE with$H_1(pw_C)$ $\xrightarrow{\quad}$ $\xleftarrow{\quad}$	perform its part in PAKE with the following modification: For each $\overline{m}_S$ to be sent to C, $\overline{\sigma}_S \leftarrow \mathsf{Sign}(pp_{\mathsf{IBS}}, ID_S, sk_{ID_S}, \overline{m}_S)$, and then send $(\overline{m}_S, \overline{\sigma}_S)$.
Output K		Output K
$\mathsf{pid}_C = ID_C\|\|ID_S$, $\mathsf{sid}_C = \mathcal{T}_{\mathsf{PAKE}}$		$\mathsf{pid}_S = ID_C\|\|ID_S$, $\mathsf{sid}_S = \mathcal{T}_{\mathsf{PAKE}}$
$ssk = H_3(\mathsf{pid}_C\|\|\mathsf{sid}_C\|\|K)$		$ssk = H_3(\mathsf{pid}_S\|\|\mathsf{sid}_S\|\|K)$

Fig. 3. Generic construction of a simplified IBS-PAKE protocol

Client C		Server S
$pp = \{\mathbb{G}, g, g_1, u, H, H_1, H_2\}$		$pp = \{\mathbb{G}, g, g_1, u, H, H_1, H_2\}$
$[ID_C, pw_C]$		$[ID_S, sk_{ID_S} = (d_S, z_S, R_S)]$
		$\pi_S[C] = (ID_C, g_1^{-H_1(pw_C)})$
$x \overset{R}{\leftarrow} \mathbb{Z}_q^*$, $W = g^x g_1^{H_1(pw_C)}$	$\xrightarrow{ID_C, W}$	$y \overset{R}{\leftarrow} \mathbb{Z}_q^*$, $Y = g^y$
If $0 = \mathsf{Vrfy}(pp, ID_S, \sigma_S, ID_S\|\|Y)$, abort	$\xleftarrow{ID_S, Y, \sigma_S}$	$\sigma_S \leftarrow \mathsf{Sign}(pp, ID_S, sk_{ID_S}, M_S)$ where $M_S = ID_S\|\|Y$
Otherwise, $K = Y^x$		$X' = Wg_1^{-H_1(pw_C)}$, $K' = (X')^y$
$\mathsf{pid}_C = ID_C\|\|ID_S$		$\mathsf{pid}_S = ID_C\|\|ID_S$
$\mathsf{sid}_C = ID_C\|\|W\|\|Y\|\|\sigma_S$		$\mathsf{sid}_S = ID_C\|\|W\|\|Y\|\|\sigma_S$
$ssk = H_2(\mathsf{pid}_C\|\|\mathsf{sid}_C\|\|K)$		$ssk = H_2(\mathsf{pid}_S\|\|\mathsf{sid}_S\|\|K')$

Fig. 4. Simplified PWIBS-AKE

References

1. Abdalla, M., Benhamouda, F., Mackenzie, P.: Security of the J-PAKE password-authenticated key exchange protocol. In: IEEE Symposium on Security and Privacy 2015, pp. 571–587. IEEE Computer Society (2015)
2. Boyarsky, M.K.: Public-key cryptography and password protocols: the multi-user case. In: ACMCCS 1999, pp. 63–72. ACM, New York (1999)
3. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer, Heidelberg (2013)
4. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

5. Barreto, P.S.L.M., Galbraith, S.D., hÉigeartaigh, C.Ó., Scott, M.: Efficient pairing computation on supersingular abelian varieties. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
6. Barreto, P.S.L.M., Lynn, B., Scott, M.: Efficient implementation of pairing based cryptosystems. J. Cryptol. **17**, 321–334 (2004). Springer-Verlag
7. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocol secure against dictionary attack. In: IEEE Symposium on Research in Security and Privacy, pp. 72–84 (1992)
8. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
9. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 268–286. Springer, Heidelberg (2004)
10. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
11. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
12. Chen, L., Harrison, K., Soldera, D., Smart, N.P.: Applications of multiple trust authorities in pairing based cryptosystems. In: Davida, G.I., Frankel, Y., Rees, O. (eds.) InfraSec 2002. LNCS, vol. 2437, pp. 260–275. Springer, Heidelberg (2002)
13. Clancy, T.: Eap password authenticated exchange, draft archive (2005). http://www.cs.umd.edu/clancy/eap-pax/
14. Akinyele, J.A., et al.: Charm: a framework for rapidly prototyping cryptosystems. J. Crypt. Eng. **3**(2), 111–128 (2013)
15. Choi, K.Y., Hwang, J.Y., Cho, J., Kwon, T.: Constructing efficient PAKE protocols from identity-based KEM/DEM, Cryptology ePrint Archive, Report 2015/606 (2015). http://eprint.iacr.org/2015/606. (To appear in WISA 2015)
16. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
17. Choi, K.Y., Hwang, J.Y., Lee, D.-H.: Efficient ID-based group key agreement with bilinear maps. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 130–144. Springer, Heidelberg (2004)
18. Dent, A.W., Galbraith, S.D.: Hidden pairings and trapdoor DDH groups. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 436–451. Springer, Heidelberg (2006)
19. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
20. Elashry, I., Mu, Y., Susilo, W.: Jhanwar-Barua's identity-based encryption revisited. In: Au, M.H., Carminati, B., Kuo, C.-C.J. (eds.) NSS 2014. LNCS, vol. 8792, pp. 271–284. Springer, Heidelberg (2014)
21. Gallbraith, S.: Pairings, Advances in Elliptic Curve Cryptography, vol. 317, Chapter IX, pp. 183–213. Cambridge University Press (2005)
22. Galindo, D., Garcia, F.D.: A schnorr-like lightweight identity-based signature scheme. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 135–148. Springer, Heidelberg (2009)
23. Gong, L.A., Lomas, T.M., Needham, R., Saltzwe, J.: Protecting poorly chosen secrets from guessing attacks. IEEE J. Sel. Areas Commun. **11**(5), 648–656 (1993)

24. Gentry, C., MacKenzie, P.D., Ramzan, Z.: A method for making password-based key exchange resilient to server compromise. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 142–159. Springer, Heidelberg (2006)
25. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. ACM Trans. Inf. Syst. Secur. **2**(3), 230–268 (1999)
26. Housley, R., Polk, T.: Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure. Wiley, Chichester (2001)
27. Hao, F., Ryan, P.Y.A.: Password authenticated key exchange by juggling. In: Christianson, B., Malcolm, J.A., Matyas, V., Roe, M. (eds.) Security Protocols 2008. LNCS, vol. 6615, pp. 159–171. Springer, Heidelberg (2011)
28. Hao, F., Shahandashti, S.F.: The SPEKE protocol revisited. In: Chen, L., Mitchell, C. (eds.) SSR 2014. LNCS, vol. 8893, pp. 26–38. Springer, Heidelberg (2014). Cryptology ePrint Archive, Report 2014/585. http://eprint.iacr.org/2014/585
29. Internet Engineering Task Forces, Eap password authenticated exchange (2005). http://www.ietf.org/internet-drafts/draft-clancy-eap-pax-03.txt
30. Jablon, D.: Strong password-only authenticated key exchange. ACM SIGCOMM Comput. Commun. Rev. **26**(5), 5–26 (1996)
31. IEEE 1363.2:2008 Specification For Password-based Public-key Cryptographic Techniques
32. ISO/IEC 11770–4:2006 Information technology - Security techniques - Key management - Part 4: Mechanisms based on weak secrets
33. ITU-T Recommendation X. 1035: Password-Authenticated Key Exchange (PAK) Protocol. https://www.itu.int/rec/T-REC-X.1035/en
34. Kwon, T.: Addendum to Summary of AMP, In Submission to the IEEE P1363 study group for future PKC standards (2003)
35. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
36. Kolesnikov, V., Rackoff, C.: Key exchange using passwords and long keys. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 100–119. Springer, Heidelberg (2006)
37. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
38. Lee, H.T., Cheon, J.H., Hong, J.: Accelerating ID-based Encryption Based on Trapdoor DL Using Pre-computation. Cryptology ePrint Archive, Report 2011/187 (2011). http://eprint.iacr.org/2011/187
39. Paterson, K.: Cryptography from pairings, Advances in Elliptic Curve Cryptography, vol. 317, Chap. X, pp. 215–251. Cambridge University Press, Cambridge (2005)
40. Pointcheval, D.: Password-based authenticated key exchange. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 390–397. Springer, Heidelberg (2012)
41. Litzenberger, D.C.: Pycrypto-the python cryptography toolkit (2014). https://www.dlitz.net/software/pycrypto
42. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000)
43. Paterson, K.G., Srinivasan, S.: On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. Des. Codes Crypt. **52**(2), 219–241 (2009)

44. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1976)
45. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
46. Schnorr, C.-P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991)
47. Certicom, S.E.C.: SEC 2: Recommended elliptic curve domain parameters. In: Proceeding of Standards for Efficient Cryptography, Version 1 (2000)
48. Brown, D.: SEC 2: Recommended Elliptic Curve Domain Parameters, Version 2 (2010). http://www.secg.org/sec2-v2.pdf
49. Shin, S., Kobara, K.: Efficient Augumented Password-only Authentication and Key Exchange for IKEv2, RFC 6628, ISSN 2070–1721, IETF (2012)
50. Sakai, R., Kasahara, M.: ID Based Cryptosystems with Pairing over Elliptic Curve, Cryptology ePrint Archive, Report 2003/054. http://eprint.iacr.org/2003/054
51. Wu, T.: SRP-6: Improvements and Refinements to the Secure Remote Password Protocol, In Submission to the IEEE P1363 Working Group (2002)
52. Yi, X., Tso, R., Okamoto, E.: ID-based group password-authenticated key exchange. In: Takagi, T., Mambo, M. (eds.) IWSEC 2009. LNCS, vol. 5824, pp. 192–211. Springer, Heidelberg (2009)
53. Yi, X., Tso, R., Okamoto, E.: Identity-based password-authenticated key exchange for client/server model. In: SECRYPT 2012, pp. 45–54 (2012)
54. Yi, X., Hao, F., Bertino, E.: ID-based two-server password-authenticated key exchange. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 257–276. Springer, Heidelberg (2014)