Scalable Wildcarded Identity-Based Encryption

Jihye Kim[1], Seunghwa Lee[1], Jiwon Lee[2], and Hyunok Oh[2](✉)

[1] Kookmin University, Seoul, Korea
{jihyek,ttyhgo}@kookmin.ac.kr
[2] Hanyang University, Seoul, Korea
{jiwonlee,hoh}@hanyang.ac.kr

Abstract. Wildcarded identity-based encryption allows a sender to simultaneously encrypt messages to a group of users matching a certain pattern, defined as a sequence of identifiers and wildcards. We propose a new wildcarded identity-based encryption scheme with generalized key delegation, which reduces the ciphertext size to be constant. To the best of our knowledge, our proposal is the first wildcarded identity-based encryption scheme that generates a constant size ciphertext regardless of the depth of the identities. The proposed scheme also improves the decryption time by minimizing the wildcard conversion cost. According to our experiment results, decryption of the proposed scheme is 3, 10, and 650 times faster than existing WIBE, WW-IBE, and CCP-ABE schemes. The proposal also subsumes the generalized key derivation naturally by allowing wildcards in the key delegation process. We prove CPA security of the proposed scheme and extend it to be CCA secure.

Keywords: Wildcard identity based encryption
Constant ciphertext · Key delegation · Pattern

1 Introduction

The advanced information technology has increased the popularity and diversity of embedded systems (or IoT devices) in a variety of applications such as smart city, transport, smart grid, production control, medical, military, and so on. In these distributed settings, messages often need to be securely delivered to a specific group of devices or users for communication and management. Some examples are as follows:

- The official commands or monitoring messages from a commander or sensors deployed to jointly monitor malicious activity for city security must be securely communicated to a specific group or user determined by its region, role, class, function, etc.
- Secure firmware updates in many systems including vehicles are crucial to improve performance and provide fixes for defective software that can lead to

costly product recalls. The firmware, the intellectual property of a company, must be distributed securely to a distinct group specified by the brand, model, year, device type, version, etc.
- In the military, tactical communications such as real-time video and targeting data need to be securely transmitted according to the access structure determined by the receiver's class, mission, location, etc.

1.1 Related Work

Identity-based encryption (IBE) is one of most powerful building blocks to provide data confidentiality, which encrypts a message without retrieving and verifying the public key separated from the identity. The IBE scheme proposed by Shamir [12] uses an actual user identity (e.g., alice@cs.univ.edu) as a public key for encryption. The first practical IBE scheme construction was presented by using bilinear maps [7,13]. It has advanced to a hierarchical identity-based encryption (HIBE) scheme in [6] where an identity is defined by multiple identity strings in a hierarchy such that keys for each identity string can be generated in a hierarchically distributed way: users at level l can derive keys for their children at level $l + 1$. The advantage of HIBE is to reduce the burden of a trusted key distribution center by distributing key derivation and solving a bottleneck problem.

Motivated by the fact that many email addresses correspond to groups of users rather than single individuals, Abdalla et al. [2] extended HIBE to wildcarded identity-based encryption (WIBE) by combining a concept called wildcard ($*$), which can be replaced by any identity string in a sequence of identity strings. A pattern (or an identity) defined as a sequence of multiple identity strings and wildcards efficiently determines a group of identities as well as a single identity. Abdalla et al. proposed three different WIBE constructions by extending the previous HIBE schemes; however, all constructions suffer from comparatively larger ciphertext size which is at least $O(L)$ where L denotes the maximum depth of a pattern (i.e., the maximum number of identity strings). Later, Birkett et al. [5] presented compilation techniques that convert any L-level CPA-secure WIBE scheme into L-level CCA-secure identity-based key encapsulation mechanisms with wildcards (WIB-KEM). They constructed more efficient CCA-secure WIBE variants by applying their compilation techniques to the CPA-secure WIBE schemes from [2]. However, the ciphertext size is still as large as that for the underlying WIBE schemes, i.e., at least $O(L)$ size ciphertext. In [1], Abdalla et al. upgraded the WIBE notion to the WW-IBE notion by combining the generalized key delegation notion in [3] to provide the full security with pattern anonymity. They utilize bilinear groups of composite order to support the full security when the maximum hierarchy depth is a polynomial in the security parameter. Although key delegation is useful to minimize the key management overhead in the distributed setting, the non-scalable ciphertext size in [1] has not been improved and remained an obstacle so far.

There are attribute-based encryption schemes (ABE) that allow more expressive policies than WIBE. Ciphertext-policy attribute-based encryption (CP-

ABE) in [4] associates to each ciphertext an access structure consisting of a logical combination of attribute values using AND and OR gates. A decryption key is given for a set of attributes and can only decrypt a ciphertext whose access structure is satisfied by the set of its attributes. WIBE schemes are a special case of CP-ABE schemes by mapping the identity vector **(*, Tesla, *, Model S)** to the access structure ($\mathbf{2}||\mathbf{Tesla} \wedge \mathbf{4}||\mathbf{Model\ S}$) where an identity is concatenated with its position index. The ciphertext size in [4] grows linearly with the number of attributes in the access structure. The authors in [10] proposed a CP-ABE scheme with constant ciphertext size, but, without supporting wildcards in its access policy. Later, the ABE scheme proposed in [14] supports wildcards with a restricted setting of only binary identities. When it is converted into the string-based identity version, the number of attributes grows exponentially to cover all possible identities in a binary notation, which results in an exponential number of public parameters. Otherwise, each attribute should be denoted in a binary format, which increases the the maximum depth of a pattern by the binary string length times.

In general, the ciphertext size is an important issue because the ciphertext is the actual payload that is transmitted via network in real applications. However, the existing schemes [1,2] produce a non-constant size ciphertext linearly increasing by the maximum depth of a pattern. It is mainly because the ciphertext should include additional information for each wildcard such that wildcards in a pattern can be transformed for every matching key element in WIBE/WW-IBE scheme. With the approach that the ciphertext contains all information required to conversion, it is not clear how to construct a wildcarded identity based encryption scheme with constant size ciphertext.

In this paper, we devise a method to convert the key pattern into matching ciphertext patterns, contrary to the approach by Abdalla et al. [1,2]. In our method, each user stores a conversion key for each non-wildcard identity in order to replace the identity by a wildcard. A pattern with l specific identity strings leads to a secret key with l conversion keys. The number of the conversion keys in a secret key is bounded by the maximum depth L. The benefit of this approach is that the extra conversion keys do not have to be delivered in the ciphertext any more because the keys deal with conversion into matching patterns. The details of the construction are described in Sect. 3.

Contributions. In this paper, we propose a new wildcarded identity based encryption scheme with constant size ciphertext and with polynomial overhead in every parameter. Our main contributions are summarized as follows:

– We propose a novel scalable wildcarded identity based encryption scheme called SWIBE. To the best of our knowledge, the proposed scheme is the first WIBE (or WW-IBE) scheme that generates a constant size ciphertext regardless of the depth, i.e., the maximum number of attributes; the ciphertext consists of just four group elements, which is comparable even to the HIBE scheme [6] that contains three group elements for its ciphertext.
– The SWIBE scheme also improves decryption performance of WIBE (or WW-IBE). Much of the decryption overhead in the existing wildcarded schemes is

in the conversion operation of wildcards in a ciphertext into identity strings of a user's secret key. While the WIBE and WW-IBE schemes [1,2] convert a ciphertext to another ciphertext for a specific matching identity strings, our scheme replaces any identity string by a wildcard; this method reduces point multiplications (i.e., exponentiations) required in the previous WIBE/WW-IBE and speeds up the decryption.

– The SWIBE scheme allows wildcards in the key delegation process as well as in the encryption procedure, naturally subsuming the generalized key derivation of wicked-IBE [3] and distributing the key management overhead. The SWIBE schemes with and without generalized key delegation correspond to WW-IBE [1] and WIBE [2], respectively.
– We formally prove the selective CPA-security of the proposed scheme under the L-BDHE assumption. We also extend it to be a CCA secure scheme.

Table 1. Comparison of HIBE, WIBE, wicked-IBE, WW-IBE, CCP-ABE, and proposed SWIBE schemes. *cf.* $\boldsymbol{e}$ *= time of scalar multiplication,* $\boldsymbol{p}$ *= time of pairing, and* L *= hierarchy depth,* ID_i *is represented using a q-bit string, size indicates the number of group elements, Enc = Encryption, and Der = Key derivation.*

	HIBE [6]	WIBE [2]	wicked-IBE [3]	WW-IBE [1]	CCP-ABE [14]	SWIBE
pp size	$L+4$	$L+4$	$L+2$	$2L+2$	$2L2^q+1$	$L+4$
SK size	$L+2$	$L+2$	$L+2$	$L+1$	$3L2^q+1$	$2L+3$
CT size	3	$L+3$	3	$3L+2$	2	4
Enc time	$(L+3)e+p$	$(L+3)e+p$	$(L+1)e+p$	$(3L+2)e$	$2L2^qe+p$	$(L+3)e+p$
Dec time	$2p$	$Le+2p$	$Le+2p$	$Le+(2L+1)p$	$2L2^qe+4L2^qp$	$Le+3p$
Wildcard use	None	Enc	Der	Enc & Der	Enc	Enc & Der

Table 1 compares the HIBE scheme [6] (that does not support wildcards as identities), the WIBE scheme [2], HIBE with the generalized key delegation (wicked-IBE) [3], WIBE scheme with generalized key delegation (WW-IBE) [1], constant-size ciphtertext policy attribute-based encryption (CCP-ABE) [14], and the proposed SWIBE scheme subsuming wildcards as identities as well as the generalized key delegation. The table shows the public parameter size (pp size), the user secret key size (SK size), the ciphertext size, the encryption time (Enc time), and the decryption time (Dec time) according to the maximum depth of the pattern (L) where e and p denote the numbers of scalar multiplications and pairings, respectively. It is assumed that each ID is represented using a q-bit string maximally. For the wildcard use, the table specifies whether wildcards are used in an encryption (Enc) algorithm or in an key derivation (Der) algorithm. Note that the SWIBE scheme has $O(L)$ size of the secret key, while it produces a constant-size ciphertext with allowing wildcards in a ciphertext pattern. Note that if each bit in ID representation is regarded as an attribute in CCP-ABE then pp size, SK size, the encryption, and the decryption time are $2Lq+1$, $3Lq+1$, $2Lqe+p$, and $2Lqe+4Lqp$, respectively. In WW-IBE and CCP-ABE, the decryption time is a major hurdle to be used in practical applications since the decryption requires pairing operations of which number is proportional to the maximum depth level L. Especially, in CCP-ABE, the number of pairing operations is dependent on the length of a bit string in each ID, in addition. In experiment, the decryption times in WW-IBE and CCP-IBE are 10 times and 650 times larger than the proposed approach.

This paper is organized as follows: Sect. 2 introduces the definitions and cryptographic assumptions. In Sect. 3, we explain the main idea of the proposed scheme and how to construct it in details. Section 4 formally proves the security of the proposed scheme and Sect. 5 extends it to be CCA secure. In Sect. 6, we show the experimental results and in Sect. 7, we conclude.

2 Definitions and Background

Wildcarded identity based encryption with generalized key delegation (WW-IBE) extends hierarchical identity based encryption (HIBE). In this section, we recall IBE, HIBE, WW-IBE, and the security definition of WW-IBE. The decryption of WIBE is omitted because WW-IBE subsumes the WIBE definition. We also describe mathematical background necessary to understand our proposal.

Identity-Based Encryption: An identity-based encryption (IBE) scheme is a tuple of algorithm $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$. A public/master key pair $(pp,\ msk)$ is generated from Setup by the trusted authority. A user decryption key with identity ID is computed as $d_{ID} \overset{\$}{\leftarrow} \mathsf{KeyDer}(msk, ID)$. To encrypt a message m for a user with identity ID, a ciphertext $C \overset{\$}{\leftarrow} \mathsf{Enc}(pp, ID, m)$ is computed, which can be decrypted by the user with ID as $\mathsf{m} \leftarrow \mathsf{Dec}(d_{ID}, C)$. We refer to [8] for details on the security definitions for IBE schemes.

Hierarchical IBE: In a hierarchical IBE (HIBE) scheme, users are organized in a tree of depth L, with the root being the master trusted authority. The identity of a user at level $0 \leq l \leq L$ in the tree is given by a vector $ID = (P_1, \ldots, P_l) \in (\{0,1\}^q)^l$. A HIBE scheme is a tuple of algorithms $\mathcal{HIBE} = (\mathsf{Setup}$, KeyDer, Enc, $\mathsf{Dec})$ providing the same functionality as in an IBE scheme, except that a user $ID = (P_1, \ldots, P_l)$ at level l can use its own secret key sk_{ID} to generate a secret key for any of its children $ID' = (P_1, \ldots, P_l, \ldots, P_L)$ via $sk_{ID'} \overset{\$}{\leftarrow} \mathsf{KeyDer}(sk_{ID}, ID')$. The secret key of the root identity at level 0 is $sk_\epsilon = msk$. Encryption and decryption are the same as for IBE, but with vectors of bit strings as identities instead of ordinary bit strings. We use the notation $P_{|l-1}$ to denote vector $(P_1, \ldots, P_{l-1})$. We refer to [6] for details on the security definitions for HIBE schemes.

Wildcarded Identity Based Encryption with Generalized Key Delegation: WW-IBE as a wildcarded identity based scheme allows general key delegation and encryption to a group that is denoted by multiple identity strings and wildcards. To make the further description simple and clear, we define the following notations similarly to [2].

Definition 1. *A pattern P is a vector $(P_1, \ldots, P_L) \in (\mathbb{Z}_p^* \cup \{*\})^L$, where $*$ is a special wildcard symbol, p is a q-bit prime number, and L is the maximal depth of the identity strings.*[1]

Definition 2. *A pattern $P' = (P'_1, \ldots, P'_L)$ belongs to P, denoted $P' \in_* P$, if and only if $\forall i \in \{1, \ldots, L\}, (P'_i = P_i) \vee (P_i = *)$.*

Definition 3. *A pattern $P' = (P'_1, \ldots, P'_L)$ matches P, denoted $P' \approx P$, if and only if $\forall i \in \{1, \ldots, L\}$, $(P'_i = P_i) \vee (P_i = *) \vee (P'_i = *)$.*

Notice that a set of matching patterns of P is a super set of belonging patterns of P. For a pattern $P = (P_1, \ldots, P_L)$, we define $W(P)$ is the set containing all wildcard indices in P, i.e. the indices $1 \leq i \leq L$ such that $P_i = *$, and $\overline{W}(P)$ is the set containing all non-wildcard indices. Clearly, $W(P) \cap \overline{W}(P) = \emptyset$ and $W(P) \cup \overline{W}(P) = \{1, \ldots, L\}$.

Definition 4. *$W(P)$ is the set containing all wildcard indices in a pattern P.*

Definition 5. *$\overline{W}(P)$ is the set containing all non-wildcard indices in a pattern P.*

Wildcarded identity-based encryption with generalized key delegation WW-IBE consists of four algorithms:

$\mathsf{Setup}(L)$ takes as input the maximal hierarchy depth L. It outputs a public parameter pp and master secret key msk.

$\mathsf{KeyDer}(sk_P, P_{new})$ takes as input a user secret key sk_P for a pattern $P = (P_1, \ldots, P_L)$ and can derive a secret key for any pattern $P_{new} \in_* P$. The secret key of the root identity is $msk = sk_{(*,\ldots,*)}$.

$\mathsf{Encrypt}(pp, P, m)$ takes as input pattern $P = (P_1, \ldots, P_L)$, message $m \in \{0,1\}^*$ and public parameter pp. It outputs ciphertext C_P for pattern P.

$\mathsf{Decrypt}(sk_P, C_{P'})$ takes as input user secret key sk_P for pattern P and ciphertext C for pattern P'. Any user in possession of the secret key such that $P' \approx P$ decrypts the ciphertext using sk_P, outputting message m. Otherwise, it outputs $\perp$.

Correctness requires that for all key pairs (pp, msk) output by Setup, all messages $m \in \{0,1\}^*$, and all patterns $P, P' \in (\mathbb{Z}_p^* \cup \{*\})^L$ such that $P \approx P'$, $\mathsf{Decrypt}(\mathsf{KeyDer}(msk, P), \mathsf{Encrypt}(pp, P', m)) = m$.

Security: We define the security notion of WW-IBE similarly to [1,2]. An adversary is allowed to choose an arbitrary pattern and query its secret key, except the query to the key derivation oracle for any pattern matching with a challenge pattern. The security is defined by an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ via the following game. Both $\mathcal{C}$ and $\mathcal{A}$ are given the hierarchy depth L and the identity bit-length q as input.

[1] We denote pattern P as in $(\mathbb{Z}_p^* \cup \{*\})^L$ instead of $(\{0,1\}^q \cup \{*\})^L$, since $\{0,1\}^q$ can be easily mapped to $\mathbb{Z}_p^*$ with a hash function.

Setup:Challenger $\mathcal{C}$ runs $\mathsf{Setup}(L)$ to obtain public parameter pp and master secret key msk. $\mathcal{C}$ gives $\mathcal{A}$ public parameter pp.

Query phase 1:

- $\mathcal{A}$ issues key derivation queries $q_{K_1}, \ldots, q_{K_m}$ in which a key derivation query consists of a pattern $P' \in (\mathbb{Z}_p^* \cup \{*\})^L$, and challenger $\mathcal{C}$ responds with $sk_{P'} \xleftarrow{\$} \mathsf{KeyDer}(msk, P')$.
- $\mathcal{A}$ issues decryption queries $q_{D_1}, \ldots, q_{D_n}$ in which a decryption query consists of pattern P for sk_P, ciphertext C, and pattern P' for C, next challenger $\mathcal{C}$ responds with $\mathsf{Decrypt}(sk_P, C_{P'})$.

Challenge:$\mathcal{A}$ outputs two equal-length challenge messages $m_0^*, m_1^* \in \{0,1\}^*$ and a challenge identity $P^* = (P_1^*, \ldots, P_{L^*}^*)$ s.t. $P^* \not\approx P'$ for all queried P'. $\mathcal{C}$ runs algorithm $C^* \xleftarrow{\$} \mathsf{Encrypt}(pp, P^*, m_b^*)$ for random bit b and gives C^* to $\mathcal{A}$.

Query phase 2:

- $\mathcal{A}$ continues to issue key derivation queries $q_{K_{m+1}}, \ldots, q_{q_K}$ as in Query phase 1, except for pattern $P' \not\approx P^*$.
- $\mathcal{A}$ continues to issue decryption queries $q_{D_{n+1}}, \ldots, q_{q_D}$ as in Query phase 1, except for C^*.

Guess:$\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

We also define the IND-ID-CPA game similarly with the IND-ID-CCA game, without allowing any decryption query.

Definition 6. *A WW-IBE is $(t, q_K, q_D, \epsilon, L)$ IND-ID-CCA (or IND-ID-CPA) secure if all t-time adversaries making at most q_K queries to the key derivation oracle and at most q_D queries to the decryption oracle have at most advantage ϵ in the IND-ID-CCA game (or the IND-ID-CPA game) described above.*

Selective Security. A selective-identity (sID) security notion IND-sID-CCA (or IND-sID-CPA) is defined analogously to the IND-ID-CCA (IND-ID-CPA) one: every procedure is the same except that the adversary has to commit to the challenge identity at the beginning of the game, before the public parameter is made available.

Definition 7. *A WW-IBE is $(t, q_K, 0, \epsilon, L)$ IND-sID-CCA (IND-sID-CPA) secure if all t-time adversaries making at most q_K queries to the key derivation oracle have at most advantage ϵ in the IND-sID-CCA (or IND-sID-CPA) game.*

Bilinear Groups and Pairings: We review the necessary facts about bilinear maps and bilinear map groups, following the standard notation [8,11].

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of prime order p.
2. g is a generator of $\mathbb{G}$.
3. $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is a bilinear map.

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two groups as above. A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ with the following properties:

1. Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$
2. Non-degenerate: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exist a group $\mathbb{G}_1$ and an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ as above.

BDHE Assumption [6]: Let $\mathbb{G}$ be a bilinear group of prime order p. Given a vector of $2L + 1$ elements $(h, g, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^L)}, g^{(\alpha^{L+2})}, \ldots, g^{(\alpha^{2L})} \in \mathbb{G}^{2L+1})$ as input, output $e(g, h)^{\alpha^{L+1}} \in \mathbb{G}_1$. As shorthand, once g and α are specified, we use y_i to denote $y_i = g^{\alpha^i} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving L-BDHE in $\mathbb{G}$ if

$$Pr[\mathcal{A}(h, g, y_1, \ldots, y_L, y_{L+2}, \ldots, y_{2L}) = e(y_{L+1}, h)] \geq \epsilon$$

where the probability is over the random choice of generators g, h in $\mathbb{G}$, the random choice of α in $\mathbb{Z}_p$, and the random bits used by $\mathcal{A}$.

The decisional version of the L-BDHE problem in $\mathbb{G}$ is defined analogously. Let $\boldsymbol{y}_{g,\alpha,L} = (y_1, \ldots, y_L, y_{L+2}, \ldots, y_{2L})$. An algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving decisional L-BDHE in $\mathbb{G}$ if

$$|Pr[\mathcal{B}(g, h, \boldsymbol{y}_{g,\alpha,L}, e(y_{L+1}, h)) = 0] - \quad Pr[\mathcal{B}(g, h, \boldsymbol{y}_{g,\alpha,L}, T) = 0]| \geq \epsilon$$

where the probability is over the random choice of generators g, h in $\mathbb{G}$, the random choice of α in $\mathbb{Z}_p$, the random choice of $T \in \mathbb{G}_1$, and the random bits consumed by $\mathcal{B}$.

Definition 8. *We say that the (decisional) (t, ϵ, L)-BDHE assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the (decisional) L-BDHE problem in $\mathbb{G}$.*

Occasionally we omit the t and ϵ, and refer to the (decisional) L-BDHE in $\mathbb{G}$.

3 The Proposed Scheme

In this section, we describe a scalable WW-IBE scheme called SWIBE. Since our SWIBE is based on the BBG-HIBE scheme proposed by Boneh et al. [6], we briefly overview the BBG-HIBE protocol and explain our idea to allow wildcards as identities in encryption. And then we illustrate our SWIBE protocol.

3.1 Overview

BBG-HIBE [6]: In the BBG-HIBE scheme, the secret key is composed of two types of keys for its purposes: decryption and key delegation. Given $pp = (g, g_1, g_2, g_3, h_1, h_2, \ldots, h_L)$ and $msk = g_2^{\alpha}$, a secret key for pattern

$P = \{P_1, \cdots, P_l\}$ consists of elements $(a_1 = g_2^\alpha(g_3 \cdot h_1^{P_1} \cdots h_l^{P_l})^r, a_2 = g^r, b = \{b_i = h_i^r\}_{i \in [l+1,\cdots,L]})$. The ciphertext encrypted for $P = \{P_1, \cdots, P_l\}$ is composed of $C_P = (g^s, (g_3 \cdot \prod_{i \in [1,\cdots,l]} h_i^{P_i})^s, M \cdot e(g_1, g_2)^s)$. To decrypt a given ciphertext $C_P = (C_1, C_2, C_3)$ with private key $sk_P = (a_1, a_2, b_{l+1}, \cdots, b_L)$, we compute $C_3 \cdot \frac{e(a_2, C_2)}{e(C_1, a_1)} = M$. Notice that the first two elements a_1 and a_2 are used for decryption, while the remaining elements $\{b_i = h_i^r\}_{i \in [l+1,\cdots,L]}$ are used for key delegation. For the easy delivery of description, we first describe our idea focusing on the decryption key without delegation, assuming the maximum depth is l.

Adding Gadgets for Wildcard Conversion: Multiple identity strings in a pattern $P = (P_1, \cdots, P_l)$ are merged into a single element in the decryption key: $a_1 = g_2^\alpha(g_3 \cdot h_1^{P_1} \cdots h_l^{P_l})^r$. We observe that given extra gadget values, each identity string of a pattern in a_1 can be replaced by another identity by multiplication. For instance, given $h_1^{(P_1' - P_1)r}$, it is possible to change $a_1 = g_2^\alpha(g_3 \cdot h_1^{P_1} \cdots h_l^{P_l})^r$ into $a_1' = g_2^\alpha(g_3 \cdot h_1^{P_1'} \cdots h_l^{P_l})^r$.

Step 1: Assume that a wildcard $*$ is mapped to some element $w \in \mathbb{Z}_p^*$. In order to allow a pattern to include a wildcard, we consider a way to include gadgets $\mathbf{d_i} = (h_i^w / h_i^{P_i})^r$ for every identity string P_i in a secret key. Then each identity part $(h_i^{P_i})^r$ can be substituted by $(h_i^w)^r$. The method, however, is not secure yet. From this extra secret key d_i and the values w and P_i, it is possible to compute $h_i^r = \mathbf{d_i}^{1/(w-P_i)}$; this leads to extract the top level secret key: $g_2^\alpha g_3^r = a_1/(h_1^r)^{P_1} \cdots (h_l^r)^{P_l}$. To avoid this attack, the gadgets need to be randomized.

Step 2: We randomize the gadget using an independent random value $t \in \mathbb{Z}_p$. Thus, the extra gadget value is revised as $\mathbf{d_i} = h_i^{wt}/h_i^{P_i r}$ and g^t is additionally appended so that they can be canceled out correctly in decryption. For example, the key $g_2^\alpha(g_3 \cdot h_1^{P_1} h_2^{P_2} h_3^{P_3})^r$ for (P_1, P_2, P_3) is changed to $g_2^\alpha(g_3 \cdot h_2^{P_2})^r \cdot (h_1^w h_3^w)^t$ for $(*, P_2, *)$ by multiplying $\mathbf{d_1 d_3}$. The encryption needs to be slightly changed to be compatible with this modification. To encrypt a message to $(*, P_2, *)$, the pattern must be divided into non-wildcard and wildcard identity groups so that they can be treated as follows: the encryption for the non-wildcard identities $(\cdot, P_2, \cdot)$ is the same as the BBG-HIBE which generates three elements: $(g^s, (g_3 \cdot h_2^{P_2})^s, M \cdot e(g_1, g_2)^s)$. The encryption for the wildcard identities $(*, \cdot, *)$ is computed as $(h_1^w h_3^w)^s$, which is used to cancel out the gadget part of user's secret key in decryption. As a result, the ciphertext size increases by a single group element to support wildcards in the proposed scheme. The key size increases linearly to the number of identity strings, which is still polynomial to the maximum depth of a pattern.

Generalized Key Delegation: Finally, the key delegation can be subsumed independently to the wildcard support in encryption and our scheme follows the key delegation method of BBG-HIBE. Only difference is that our key delegation is more flexible because it does not have to follow the hierarchical order as in BBG-HIBE.

The complete scheme is described in the following Sect. 3.2 with $w = 1$ and we prove the security of the proposed scheme in Sect. 4.

3.2 Construction

We propose a new WW-IBE scheme called SWIBE with constant size ciphertexts and $O(L)$ size keys.

$\mathsf{Setup}(L)$: L indicates the maximum hierarchy depth. The generation of a random initial set of keys proceeds as follows. Select a random integer $\alpha \in \mathbb{Z}_p^*$, and $O(L)$ random group elements $g, g_2, g_3, h_1, h_2, \ldots, h_L \in \mathbb{G}$, and compute $g_1 = g^\alpha$. The public parameter is given by $pp \leftarrow (g, g_1, g_2, g_3, h_1, h_2, \ldots, h_L)$. A master secret key is defined as $msk = g_2^\alpha$.

$\mathsf{KeyDer}(pp,\ sk_P,\ P')$: To compute the secret key $sk_{P'}$ for a pattern $P' = (P'_1, \ldots, P'_L) \in (\mathbb{Z}_p^* \cup \{*\})^L$ from the master secret key, first two randoms $r, t \xleftarrow{\$} \mathbb{Z}_p^*$ are chosen, then secret key $sk_{P'} = (a'_1, a'_2, a'_3, b', c', d')$ for P' is constructed as

$$a'_1 = msk(g_3 \cdot \prod_{i \in \overline{W}(P')} h_i^{P'_i})^r, a'_2 = g^r, a'_3 = g^t, b' = \{b'_i = h_i^r\}_{i \in W(P')},$$

$$c' = \{c'_i = h_i^t\}_{i \in W(P')}, d' = \{d'_i = h_i^t / h_i^{P'_i r}\}_{i \in \overline{W}(P')}$$

In order to generate secret key $sk_{P'}$ for a pattern P' from secret key $sk_P = (a_1, a_2, a_3, b, c)$ for a pattern P such that $P' \in_* P$, simply choose two randoms $r', t' \xleftarrow{\$} \mathbb{Z}_q^*$ and output $sk_{P'} = (a'_1, a'_2, a'_3, b', c', d')$, where

$$a'_1 = a_1 \cdot (\prod_{i \in \overline{W}(P') \cap W(P)} b_i^{P'_i}) \cdot (g_3 \prod_{i \in \overline{W}(P')} h_i^{P'_i})^{r'}, a'_2 = a_2 \cdot g^{r'}, a'_3 = a_3 \cdot g^{t'},$$

$$b' = \{b'_i = b_i \cdot h_i^{r'}\}_{i \in W(P')}, c' = \{c'_i = c_i \cdot h_i^{t'}\}_{i \in W(P')}$$

$$d' = \{d'_i = d_i \cdot \frac{h_i^{t'}}{h_i^{P'_i r'}}\}_{i \in \overline{W}(P') \cap \overline{W}(P)} \cup \{d'_i = \frac{c_i}{b_i^{P'_i}} \cdot \frac{h_i^{t'}}{h_i^{P'_i r'}}\}_{i \in \overline{W}(P') \cap W(P)}$$

$\mathsf{Encrypt}(pp,\ P,\ m)$: To encrypt a message $m \in \mathbb{G}_1$ to pattern $P = (P_1, \ldots, P_L)$ under pp, choose $s \xleftarrow{\$} \mathbb{Z}_p^*$, and compute $C_P = (C_1, C_2, C_3, C_4)$

$$C_1 = g^s, \quad C_2 = (g_3 \cdot \prod_{i \in \overline{W}(P)} h_i^{P_i})^s, C_3 = m \cdot e(g_1, g_2)^s, C_4 = (\prod_{i \in W(P)} h_i)^s$$

$\mathsf{Decrypt}(sk_P,\ C_{P'})$: Set $C = (C_1, C_2, C_3, C_4)$ and $sk_P = (a_1, a_2, a_3, b, c, d)$. If $P' \approx P$ then compute $a'_1 = a_1 \cdot \prod_{i \in \overline{W}(P') \cap W(P)} b_i^{P'_i} \cdot \prod_{i \in W(P') \cap W(P)} c_i \cdot \prod_{i \in W(P') \cap \overline{W}(P)} d_i$ and output

$$C_3 \cdot \frac{e(a_2, C_2) \cdot e(a_3, C_4)}{e(C_1, a'_1)} = m.$$

Otherwise, output $\perp$.

The fact that decryption works can be seen as follows. We denote $\mathsf{W}_{P'P} = W(P') \cap W(P)$, $\mathsf{W}_{\overline{P'}P} = \overline{W}(P') \cap W(P)$, $\mathsf{W}_{P'\overline{P}} = W(P') \cap \overline{W}(P)$, and $\mathsf{W}_{\overline{P'}\overline{P}} = \overline{W}(P') \cap \overline{W}(P)$ to simplify notations.

Since $a_1 = g_2^\alpha(g_3 \prod_{i \in \overline{W}(P)} h_i^{P_i})^r$, $b_i = h_i^r$, $c_i = h_i^t$, and $d_i = \frac{h_i^t}{h_i^{P_i r}}$,

$$\begin{aligned}
a_1' =& a_1 \cdot \prod_{i \in \mathsf{W}_{\overline{P'}P}} b_i^{P_i'} \cdot \prod_{i \in \mathsf{W}_{P'P}} c_i \cdot \prod_{i \in \mathsf{W}_{P'\overline{P}}} d_i \\
=& g_2^\alpha (g_3 \cdot \prod_{i \in \overline{W}(P)} h_i^{P_i})^r \cdot \prod_{i \in \mathsf{W}_{\overline{P'}P}} h_i^{P_i' r} \cdot \prod_{i \in \mathsf{W}_{P'P}} h_i^t \cdot \prod_{i \in \mathsf{W}_{P'\overline{P}}} \frac{h_i^t}{h_i^{P_i r}} \\
=& g_2^\alpha (g_3 \cdot \prod_{i \in \overline{W}(P)} h_i^{P_i} \cdot \prod_{i \in \mathsf{W}_{\overline{P'}P}} h_i^{P_i'} \cdot \prod_{i \in \mathsf{W}_{P'\overline{P}}} h_i^{-P_i})^r \cdot \prod_{i \in \mathsf{W}_{P'\overline{P}}} h_i^t \cdot \prod_{i \in \mathsf{W}_{P'P}} h_i^t \\
=& g_2^\alpha (g_3 \cdot \prod_{i \in \mathsf{W}_{\overline{P'}\overline{P}}} h_i^{P_i} \cdot \prod_{i \in \mathsf{W}_{\overline{P'}P}} h_i^{P_i'})^r \cdot \prod_{i \in W(P')} h_i^t \\
=& g_2^\alpha (g_3 \cdot \prod_{i \in \mathsf{W}_{\overline{P'}\overline{P}}} h_i^{P_i'} \cdot \prod_{i \in \mathsf{W}_{\overline{P'}P}} h_i^{P_i'})^r \cdot \prod_{i \in W(P')} h_i^t \quad (\because P' \approx P) \\
=& g_2^\alpha (g_3 \cdot \prod_{i \in \overline{W}(P')} h_i^{P_i'})^r \cdot \prod_{i \in W(P')} h_i^t.
\end{aligned}$$

$$\begin{aligned}
\frac{e(a_2, C_2) \cdot e(a_3, C_4)}{e(C_1, a_1')} =& \frac{e(g^r, (g_3 \cdot \prod_{i \in \overline{W}(P')} h_i^{P_i'})^s) \cdot e(g^t, (\prod_{i \in W(P')} h_i)^s)}{e(g^s, g_2^\alpha (g_3 \cdot \prod_{i \in \overline{W}(P')} h_i^{P_i'})^r \cdot \prod_{i \in W(P')} h_i^t)} \\
=& \frac{1}{e(g, g_2)^{s\alpha}} = \frac{1}{e(g_1, g_2)^s}.
\end{aligned}$$

4 Security Proof

We show an IND-sID-CPA-security of the SWIBE scheme in the standard model and then transform the scheme to achieve IND-ID-CPA-security in the random oracle model.

4.1 Selective Security

Theorem 1. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the decisional (t, ϵ, L)-BDHE assumption holds in $\mathbb{G}$. Then our SWIBE is $(t', q_K, 0, \epsilon, L)$ IND-sID-CPA secure for arbitrary L, and $t' < t - O(L\boldsymbol{e} + \boldsymbol{p})$, where $\boldsymbol{e}$ is a time of scalar multiplication and $\boldsymbol{p}$ is a time of pairing in $\mathbb{G}$.*

Proof is available in Appendix.

4.2 Full Security

Theorem 1 demonstrates that the SWIBE scheme is IND-sID-CPA secure. Therefore, the SWIBE scheme is secure when the attacker in advance commits to the pattern to attempt to attack.

Any HIBE or WIBE scheme that is IND-sID-CPA secure can be transformed into a HIBE or WIBE scheme that is IND-ID-CPA secure in the random oracle model, as described in [2,6] for the case of HIBE schemes and the case of WIBE schemes, respectively, with losing a factor $O({q_H}^L)$ in reduction tightness. The transformation is as follows:

Let $H : \{0,1\}^* \rightarrow \{0,1\}^d$ be a hash function and let $SWIBE_H$ be a $SWIBE$ scheme where a pattern $P = (P_1, \cdots, P_L)$ is replaced by $P' = (P'_1, \cdots, P'_L)$ with $P'_i = H(P_i)$ if $P_i \neq *$ and $P'_i = *$ if $P_i = *$ before it is used in key generation, encryption and decryption algorithms. Then, if H is collision resistant, $SWIBE_H$ becomes fully secure, but the reduction introduces a loss factor of $O({q_H}^{dL})$. In the random oracle model, $SWIBE_H$ is fully secure with a reduction loss factor of $O({q_H}^L)$. Thus, this transformation only works when the hierarchy depth is small enough.

Theorem 2. *[2] Suppose the SWIBE scheme is $(t, q_K, 0, \epsilon, L)$ IND-sID-CPA secure for arbitrary L with a pattern space $|P|$. The $SWIBE_H$ scheme described above is $(t', q'_K, q'_H, 0, \epsilon', L)$ IND-ID-CPA secure in the random oracle model for all*

$$t' \leq t, \; q'_K \leq q_K \text{ and } \epsilon' \geq (L+1)(q'_H + 1)^L \cdot \epsilon + q'^2_H/|P|.$$

5 Extension to CCA Security

We extend the semantically secure scheme to obtain chosen ciphertext security using the similar technique in [9]. Given a strong one-time signature scheme ($SigKeyGen$, $Sign$, $Verify$), we enable construction of an L-level IND-sID-CCA secure scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ from the $(L+1)$-level IND-sID-CPA scheme $\Pi' = (\mathsf{Setup}', \mathsf{KeyDer}'\ \mathsf{Encrypt}', \mathsf{Decrypt}')$. The intuition is that $P = (P_1, \cdots, P_L) \in \{\mathbb{Z}_p^* \cup \{*\}\}^L$ in Π is mapped to $P' = (P_1, \cdots, P_L, *) \in \{\mathbb{Z}_p^* \cup \{*\}\}^{L+1}$ in Π' and the $(L+1)$-th identity string is determined by the verification key of one-time signature scheme. When encrypting a message m with $P = (P_1, \cdots, P_L)$ in Π, the sender generates a one-time signature key (K_{sig}, V_{sig}) such that $V_{sig} \in \mathbb{Z}_p^*$ and then encrypts m with $P' = (P_1, \cdots, P_L, V_{sig})$ using $\mathsf{Encrypt}'$ in Π'. We describe how to construct L-level Π with $(L+1)$-level Π' and a one-time signature scheme in the following:

$\mathsf{Setup}(L)$ runs $\mathsf{Setup}'(L+1)$ to obtain (pp', msk'). Given $pp' \leftarrow (g, g_1, g_2, g_3, h_1, \cdots, h_{L+1})$ and msk', the public parameter is $pp \leftarrow pp'$ and the master secret key is $msk \leftarrow msk'$.

$\mathsf{KeyDer}(pp, sk_P, P')$ is the same as the KeyDer' algorithm.

$\mathsf{Encrypt}(pp, P, m)$ runs $SigKeyGen(1^\lambda)$ algorithm to obtain a signature signing key K_{sig} and a verification key V_{sig}. For a given pattern $P = (P_1, \cdots, P_L)$, encode P to $P' = (P_1, \cdots, P_L, V_{sig})$, compute $C \xleftarrow{\$} \mathsf{Encrypt}'(pp', P', m)$ and $\sigma \xleftarrow{\$} Sign(K_{sig}, C)$, and output $CT = (C, \sigma, V_{sig})$

$\mathsf{Decrypt}(sk_P, C_{P'})$: Let $C_{P'} = (C, \sigma, V_{sig})$.

1. Verify that σ is the valid signature of C under the key V_{sig}. If invalid, output $\perp$.
2. If $P \approx P'$ then run $\mathsf{Decrypt}'$ $(sk_P,\ C_{P'})$ to extract the message. Otherwise, output $\perp$.

Theorem 3. *Let $\mathbb{G}$ be a bilinear group of prime order p. The above SWIBE Π is $(t, q_K, q_D,\ \epsilon_1 + \epsilon_2,\ L)$ IND-sID-CCA secure assuming the SWIBE Π' is $(t',\ q'_K,\ 0,\ \epsilon_1,\ L+1)$ IND-sID-CPA secure in $\mathbb{G}$ and signature scheme is $(t_s,\ \epsilon_2)$ strongly existentially unforgeable with $q_K < q'_K$, $t < t' - (L\boldsymbol{e} + 3\boldsymbol{p})q_D - t_s$, where $\boldsymbol{e}$ is exponential time, $\boldsymbol{p}$ is pairing time, and t_s is sum of SigKeyGen, Sign and $Verify$ computation time.*

Proof is available in Appendix.

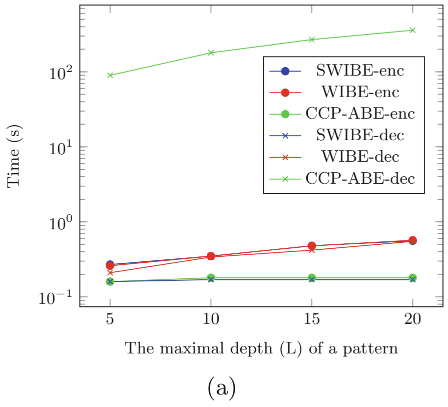
6 Experiment

In this section, we measure the execution times of encryption and decryption of the proposed SWIBE, WIBE [2], wicked-IBE [3], WW-IBE [1], and CCP-ABE [14]. We have implemented the algorithms based on the PBC (pairing based cryptography) library with *a_param* and executed them on Intel Edison with a 32-bit Intel Atom processor 500 MHz and ublinux 3.10.17.

Figure 1a illustrates encryption and decryption times of SWIBE, WIBE, and CCP-ABE by varying the maximal hierarchy depth (L) from 5 to 20. Note that in WIBE and CCP-ABE, only a ciphertext can include wildcards, while the proposed SWIBE allows wildcards in both key and ciphertext. While WIBE performs point multiplications to convert a ciphertext to another ciphertext for a specific matching ID, SWIBE computes point additions to replace any ID by a wildcard. In CCP-ABE, each bit in an ID is regarded as an attribute where each pattern (ID) is 32 bit. Since the decryption requires pairing operations of which number is proportional to the number of attributes in CCP-ABE, the decryption is very slow. On the other hand, since point additions is negligible compared with a pairing operation, decryption time of SWIBE remains as constant. SWIBE improves decryption performance by up to 3 times and 650 times compared with WIBE and CCP-ABE.

Figure 1b compares encryption and decryption performance between SWIBE and wicked-IBE. In this case, a private key may include wildcards but no wildcard is allowed in a ciphertext in wicked-IBE. Since a point multiplication is required to decrypt a ciphertext in both SWIBE and wicked-IBE, both schemes show similar encryption and decryption performance even though SWIBE allows wildcards in a ciphertext which is prohibited in wicked-IBE.

Figure 1c compares encryption and decryption performance between SWIBE and WW-IBE. Both SWIBE and WW-IBE allow wildcards in a key and a ciphertext. While a point multiplication is required to decrypt a ciphertext in SWIBE, $2L$ number of pairing operations are required in WW-IBE. SWIBE improves decryption performance by 10 times compared with WW-IBE.



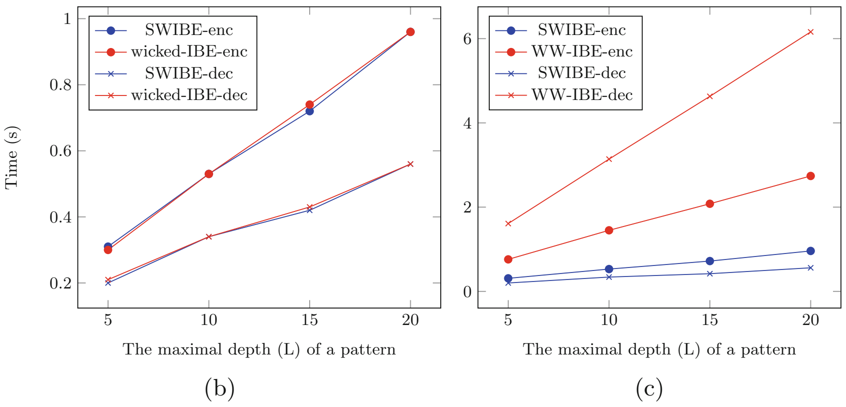


Fig. 1. Encryption and decryption time in (a) SWIBE, WIBE, and CCP-ABE, (b) in SWIBE and wicked-IBE, and (c) in SWIBE and WW-IBE

7 Conclusion

In this paper, we propose a new wildcard identity-based encryption called SWIBE, define appropriate security notions for SWIBE, and provide an efficient provably secure SWIBE construction with constant size ciphertext. Our SWIBE scheme allows wildcards for both key derivation and encryption, and it is the first success on constructing a constant-size ciphertext in a wildcarded identity-based encryption (WIBE) with fast decryption. We prove that our scheme is semantically secure based on L-BDHE assumption. In addition, we extend it to be CCA secure. Experimental results show that the proposed SWIBE improves the decryption performance by 3, 10, and 650 times compared with WIBE, WW-IBE, and CCP-ABE, respectively. It is our future work to construct a fully secure efficient scheme with a decent reduction loss factor in the standard model by considering a different setting such as a composite order group.

Acknowledgement. This work was supported by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-6-00599, A Study on Functional Signature and Its Applications and No. 2017-0-00661, Prevention of video image privacy infringement and authentication technique), by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2017R1A2B4009903 and No. 2016R1D1A1B03934545), and by Basic Research Laboratory Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (MSIP) (No. 2017R1A4A1015498).

A Appendix

Theorem 1. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the decisional (t, ϵ, L)-BDHE assumption holds in $\mathbb{G}$. Then our SWIBE is $(t', q_K, 0, \epsilon, L)$ IND-sID-CPA secure for arbitrary L, and $t' < t - O(L\boldsymbol{e} + \boldsymbol{p})$, where $\boldsymbol{e}$ is a time of scalar multiplication and $\boldsymbol{p}$ is a time of pairing in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the SWIBE scheme. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the (decisional) L-BDHE problem in $\mathbb{G}$.

For a generator $g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p^*$, let $y_i = g^{\alpha^i} \in \mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random tuple $(g, h, y_1, \ldots, y_L, y_{L+2}, \ldots, y_{2L}, T)$ that is either sampled from P_{BDHE} (where $T = e(g, h)^{(\alpha^{L+1})}$) or from R_{BDHE} (where T is uniform and independent in $\mathbb{G}_1$). Algorithm $\mathcal{B}$'s goal is to output 1 when the input tuple is sampled from P_{BDHE} and 0 otherwise. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective subset game as follows:

Init: The game begins with $\mathcal{A}$ first outputting an identity vector $P^* = (P_1^*, \ldots, P_L^*) \in_* (\mathbb{Z}_p^* \cup \{*\})^L$.

Setup: To generate a public parameter, algorithm $\mathcal{B}$ picks a random γ in $\mathbb{Z}_p$ and sets $g_1 = y_1 = g^\alpha$ and $g_2 = y_L g^\gamma = g^{\gamma + (\alpha^L)}$. Next, $\mathcal{B}$ picks random $\gamma_i \in \mathbb{Z}_p^*$ for $i = 1, \ldots, L$, and sets $h_i = g^{\gamma_i}/y_{L-i+1}$ for $i \in \overline{W}(P^*)$ and $h_i = g^{\gamma_i}$ for $i \in W(P^*)$. Algorithm $\mathcal{B}$ also picks a random δ in $\mathbb{Z}_p^*$ and sets $g_3 = g^\delta \prod_{i \in \overline{W}(P^*)} y_{L-i+1}^{P_i^*}$.

Key derivation queries: Suppose adversary $\mathcal{B}$ makes a key derivation query for pattern $P = (P_1, \ldots, P_L) \in_* (\mathbb{Z}_p^* \cup \{*\})^L$. By the definition of the security experiment, we know that $P^* \not\in_* P$. That means that there exists an index $k \in \overline{W}(P^*) \cap \overline{W}(P)$ such that $P_k \neq P_k^*$. We define k to be the smallest one among all possible indices. $\mathcal{B}$ picks two random $\tilde{r}, \tilde{t} \in Z_p^*$ and (implicitly) sets $r \leftarrow -\frac{\alpha^k}{P_k^* - P_k} + \tilde{r}$ and $t \leftarrow r \cdot P_k^* + \tilde{t}$. Secret key $sk_P = (a_1, a_2, a_3, b, c, d)$ for P is constructed as

$$a_1 = g_2^\alpha \cdot (g_3 \prod_{i \in \overline{W}(P) h_i^{P_i}})^r ; a_2 = g^r ; a_3 = g^t,$$

$$b = \{b_i = h_i^r\}_{i \in W(P)}, c = \{c_i = h_i^t\}_{i \in W(P)}, d = (d_i = h_i^t / h_i^{P_i^* r})_{i \in \overline{W}(P)}$$

We have

$$\begin{aligned}(g_3 \prod_{i \in \overline{W}(P)} h_i^{P_i})^r &= (g^\delta \prod_{i \in \overline{W}(P^*)} y_{L-i+1}^{P_i^*} \prod_{i \in \overline{W}(P)} g^{\gamma_i P_i} y_{L-i+1}^{-P_i})^r \\ =&(g^{\delta + \sum_{i \in \overline{W}(P)} P_i \gamma_i} \cdot \prod_{i \in \{1,\ldots,k-1,k+1,\ldots,L\}} y_{L-i+1}^{P_i^* - P_i} \cdot y_{L-k+1}^{P_k^* - P_k})^r\end{aligned}$$

where let $P_j^* = 0$ for $j \in W(P^*)$ and $P_j = 0$ for $j \in W(P)$.

We split this term up into two factors $A \cdot Z$, where $A = (y_{L-k+1}^{P_k^* - P_k})^r$. It can be checked that Z can be computed by $\mathcal{A}$, i.e. the terms y_i only appear with indices $i \in \{1, \ldots, L\}$. Term A can be expressed as

$$A = g^{\alpha^{L-k+1}(P_k^* - P_k)(-\frac{\alpha^k}{P_k^* - P_k} + \tilde{r})} = y_{L+1}^{-1} \cdot y_{L-k+1}^{(P_k^* - P_k)\tilde{r}}$$

Hence,

$$a_1 = g_2^\alpha \cdot A \cdot Z = y_{L+1} y_1^\gamma \cdot y_{L+1}^{-1} y_{L-k+1}^{(P_k^* - P_k)\tilde{r}} \cdot Z = \quad y_1^\gamma \cdot y_{L-k+1}^{(P_k^* - P_k)\tilde{r}} \cdot Z$$

can be computed by $\mathcal{A}$. Furthermore,

$$g^r = g^{-\frac{\alpha^k}{P_k^* - P_k} + \tilde{r}} = y_k^{-\frac{1}{P_k^* - P_k}} \cdot g^{\tilde{r}}$$

and for each $i \in W(P)$,

$$\begin{aligned} h_i^r &= (g^{\gamma_i} / y_{L-i+1})^{-\frac{\alpha^k}{P_k^* - P_k} + \tilde{r}} = y_k^{-\frac{\gamma_i}{P_k^* - P_k}} y_{L+k-i+1}^{\frac{1}{P_k^* - P_k}} \cdot g^{\gamma_i \tilde{r}} \cdot y_{L-i+1}^{-\tilde{r}} \\ h_i^t &= (h_i)^{r \cdot P_k^* + \tilde{t}} = h_i^{P_k^* r} \cdot h_i^{\tilde{t}} = y_k^{-\frac{\gamma_i P_k^*}{P_k^* - P_k}} y_{L+k-i+1}^{\frac{P_k^*}{P_k^* - P_k}} \cdot g^{\gamma_i(\tilde{r} P_k^* + \tilde{t})} \cdot y_{L-i+1}^{-(\tilde{r} P_k^* + \tilde{t})} \end{aligned}$$

can be computed since $k \notin W(P)$.
And for each $i \in \overline{W}(P)$,

$$\begin{aligned} h_i^t / h_i^{P_i^* r} &= h_i^{r P_k^* + \tilde{t}} / h_i^{P_i^* r} = h_i^{(P_k^* - P_i^*) r + \tilde{t}} = (g^{\gamma_i} / y_{L-i+1})^{(P_k^* - P_i^*)(-\frac{\alpha^k}{P_k^* - P_k} + \tilde{r}) + \tilde{t}} \\ &= (y_k^{-\frac{\gamma_i}{P_k^* - P_k}} y_{L+k-i+1}^{\frac{1}{P_k^* - P_k}} \cdot g^{\gamma_i \tilde{r}} \cdot y_{L-i+1}^{-\tilde{r}})^{(P_k^* - P_i^*)} \cdot (g^{\gamma_i} / y_{L-i+1})^{\tilde{t}}. \end{aligned}$$

If $i = k$, $P_k^* - P_i^* = 0$. So $\mathcal{A}$ can compute it. Otherwise also $\mathcal{A}$ can compute it since $i \neq k$ and y_{L+1} does not appear in the equation.

Challenge: To generate a challenge, $\mathcal{B}$ computes C_1, C_2, and C_4 as h, $h^{\delta+\sum_{i\in\overline{W}(P^*)}(\gamma_i P_i^*)}$, and $h^{\sum_{i\in W(P^*)}\gamma_i}$, respectively. It then randomly chooses a bit $b \in \{0,1\}$ and sets $C_3 = m_b \cdot T \cdot e(y_1,h)^{\gamma}$. It gives $C = (C_1, C_2, C_3, C_4)$ as a challenge to $\mathcal{A}$. We claim that when $T = e(g,h)^{(\alpha^{L+1})}$(i.e. the input to $\mathcal{B}$ is an L-BDHE tuple) then (C_1, C_2, C_3, C_4) is a valid challenge to $\mathcal{A}$ as in a real attack. To see this, write $h = g^c$ for some (unknown) $c \in \mathbb{Z}_p^*$. Then

$$h^{\delta+\sum_{i\in\overline{W}(P^*)}(\gamma_i P_i^*)} = (g^{\delta+\sum_{i\in\overline{W}(P^*)}(\gamma_i P_i^*)})^c$$

$$=(g^{\delta}\cdot \prod_{i\in\overline{W}(P^*)} y_{L-i+1}^{P_i^*} \prod_{i\in\overline{W}(P^*)} (\frac{g^{\gamma_i}}{y_{L-i+1}})^{P_i^*})^c = (g_3 \prod_{i\in\overline{W}(P^*)} h_i^{P_i^*})^c,$$

$$h^{\sum_{i\in W(P^*)}\gamma_i} = (g^{\sum_{i\in W(P^*)}\gamma_i})^c = (\prod_{i\in W(P^*)} g^{\gamma_i})^c$$

and

$$e(g,h)^{(\alpha^{L+1})} \cdot e(y_1,h)^{\gamma} = e(y_1,y_L)^c \cdot e(y_1,g)^{\gamma\cdot c} = e(y_1, y_L g^{\gamma})^c = e(g_1,g_2)^c.$$

Therefore, by definition, $e(y_{L+1},g)^c = e(g,h)^{(\alpha^{L+1})} = T$ and hence $C = (C_1, C_2, C_3, C_4)$ is a valid challenge to $\mathcal{A}$. On the other hand, when T is random in $\mathbb{G}_1$ (i.e. the input to $\mathcal{B}$ is a random tuple) then C_3 is just a random independent element in $\mathbb{G}_1$ to $\mathcal{A}$.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $T = e(g,h)^{(\alpha^{L+1})}$. Otherwise, it outputs 0 meaning T is random in $\mathbb{G}_1$.

When the input tuple is sampled from P_{BDHE} (where $T = e(g,h)^{(\alpha^{L+1})}$) then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ satisfies $|Pr[b=b'] - 1/2| \geq \epsilon$. When the input tuple is sampled from R_{BDHE} (where T is uniform in $\mathbb{G}_1$) then $Pr[b = b'] = 1/2$. Therefore, with g, h uniform in $\mathbb{G}$, α uniform in $\mathbb{Z}_p$, and T uniform in $\mathbb{G}_1$ we have that $|Pr[B(g,h,\boldsymbol{y}_{g,\alpha,L}, e(g,h)^{(\alpha^{L+1})}) = 0] - Pr[B(g,h,\boldsymbol{y}_{g,\alpha,L},T) = 0]| \geq |(1/2+\epsilon) - 1/2| = \epsilon$ as required, which completes the proof of the theorem.

Theorem 3. *Let $\mathbb{G}$ be a bilinear group of prime order p. The above SWIBE Π is $(t, q_K, q_D,\ \epsilon_1 + \epsilon_2,\ L)$ IND-sID-CCA secure assuming the SWIBE Π' is $(t',\ q_K',\ 0,\ \epsilon_1,\ L+1)$ IND-sID-CPA secure in $\mathbb{G}$ and signature scheme is $(t_s,\ \epsilon_2)$ strongly existentially unforgeable with $q_K < q_K'$, $t < t' - (L\boldsymbol{e} + 3\boldsymbol{p})q_D - t_s$, where $\boldsymbol{e}$ is exponential time, $\boldsymbol{p}$ is pairing time, and t_s is sum of SigKeyGen, Sign and Verify computation time.*

Proof. Suppose there exists a t-time adversary $\mathcal{A}$ breaking IND-sID-CCA security. We build an algorithm $\mathcal{B}$ breaking IND-sID-CPA. Algorithm $\mathcal{B}$ proceeds as follows:

$\mathcal{A}$ announces $P^* = (P_1^*, \cdots, P_L^*)$. $\mathcal{B}$ runs $SigKeyGen(1^{\lambda})$ algorithm to obtain a signature signing key K_{sig}^* and a verification key V_{sig}^*, and announces $P^* = (P_1^*, \cdots, P_L^*, V_{sig}^*)$.

Setup: $\mathcal{B}$ gets the public parameter pp from the challenger and forwards it to $\mathcal{A}$.

Key derivation queries: For a query on $P \not\approx P^*$ from $\mathcal{A}$, $\mathcal{B}$ responds with KeyDer(pp, msk, P).

Decryption queries: Algorithm $\mathcal{A}$ issues decryption queries on $(sk_P, C_{P'})$. Let $C_{P'} = (\ (C_1, C_2, C_3, C_4),\ \sigma,\ V_{sig})$. If $P \not\approx P^*$ then output $\perp$. If $P' = (P_1^*, \cdots, P_L^*, V_{sig}^*)$ then $\mathcal{B}$ aborts. (A *forge* event occurs.) Otherwise, $\mathcal{B}$ queries KeyDer(pp, msk, P), gets sk_P, and decrypts $C_{P'}$ using sk_P.

Challenge: $\mathcal{A}$ gives the challenge (m_0, m_1) to $\mathcal{B}$. $\mathcal{B}$ gives the challenge (m_0, m_1) to $\mathcal{C}$ and gets the challenge (C_b) from $\mathcal{C}$. To generate challenge for $\mathcal{A}$, $\mathcal{B}$ computes C^* as follows:

$$\sigma^* \xleftarrow{\$} Sign(C_b, K_{sig}^*), \quad C^* \xleftarrow{\$} (C_b, \sigma^*, V_{sig}^*)$$

$\mathcal{B}$ replies with C^* to $\mathcal{A}$.

Query phase2: Same as in query phase 1 except decryption query for C* is not allowed.

Guess: The $\mathcal{A}$ outputs a guess $b \in \{0, 1\}$. $\mathcal{B}$ outputs b.

In the above experiment, $\mathcal{A}$ causes an abort by submitting a query that includes an existential forgery under K_{sig}^* on some ciphertexts. Our simulator is able to use this forgery to win the existential forgery game. Note that during the game the adversary makes only one chosen message query to generate the signature needed for the challenge ciphertext. Thus, $Pr[forge] < \epsilon_2$. It now follows that $\mathcal{B}$'s advantage is at least ϵ_1 as required.

References

1. Abdalla, M., Caro, A.D., Phan, D.H.: Generalized key delegation for wildcarded identity-based and inner-product encryption. IEEE Trans. Inf. Forensics Secur. **7**(6), 1695–1706 (2012). https://doi.org/10.1109/TIFS.2012.2213594
2. Abdalla, M., Catalano, D., Dent, A.W., Malone-Lee, J., Neven, G., Smart, N.P.: Identity-based encryption gone wild. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 300–311. Springer, Heidelberg (2006). https://doi.org/10.1007/11787006_26
3. Abdalla, M., Kiltz, E., Neven, G.: Generalized key delegation for hierarchical identity-based encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 139–154. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74835-9_10
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption, pp. 321–334. IEEE Computer Society (2007)
5. Birkett, J., Dent, A.W., Neven, G., Schuldt, J.C.N.: Efficient chosen-ciphertext secure identity-based encryption with wildcards. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 274–292. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73458-1_21

6. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_26
7. 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). https://doi.org/10.1007/3-540-44647-8_13
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)
9. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_13
10. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 13–23. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00843-6_2
11. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_19
12. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
13. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_7
14. Zhou, Z., Huang, D.: On efficient ciphertext-policy attribute based encryption and broadcast encryption: extended abstract. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS, pp. 753–755 (2010)