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. Wildcard identity-based encryption (WIBE) 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 novel scalable wildcarded identity-based encryption, called SWIBE, which reduces the ciphertext size to be constant. To the best of our knowledge, SWIBE is the first wildcard identity-based encryption scheme that generates a constant size ciphertext regardless of the depth of the identities with fast decryption. The proposed scheme improves the decryption time. According to our experiment results, decryption of the SWIBE scheme is 3, 10, and 650 times faster than existing WIBE, WW-IBE, and CCP-ABE schemes. The SWIBE scheme 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.

* This paper is a full version of the conference paper with the same title that was published in the Euuropean Symposium on Research in Computer Security (ESORICS), 2018

- 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.

Related Works. Identity-based encryption (IBE) is one of most powerful building blocks to provide data confidentiality which can encrypt a message without retrieving and verifying the public key separated from the identity. The IBE scheme proposed by Shamir [Sha84] 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 [BF01, Wat05]. It has advanced to a hierarchical identity-based encryption (HIBE) scheme in [BBG05] 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. The technical intuition of the key delegation in the HIBE system has been adapted to other various cryptographic constructions, such as broadcast encryption [BGW05, LLWQ14, BH08].

Motivated by the fact that many email addresses correspond to groups of users rather than single individuals, Abdalla et al. [ACD+06] 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. WIBE engages a pattern as a public key for encryption, meaning that a single ciphertext encrypted on a pattern **(edu, univ, cs,** $*$**)** (corresponding to $*$**@cs.univ.edu**) can be simultaneously delivered to the multiple users with identities such as **alice@cs.univ.edu** and **bob@cs.univ.edu**. 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. [BDNS07] 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 [ACD+06]. However, the ciphertext size is still as large as that for the underlying WIBE schemes, i.e., at least $O(L)$ size ciphertext. In [ACP12], Abdalla et al. upgraded the WIBE notion to the WW-IBE notion by combining the generalized key delegation notion in [AKN07]. Although key delegation is useful to minimize the key management overhead in

the distributed setting, the non-scalable ciphertext size in [ACP12] 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 [BSW07] 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 **(2||Tesla** $\wedge$ **4||Model S)**. The device with a set of attributes **(2018, Tesla, Powertrain, Model S)** can decrypt the ciphertext. However, its ciphertext size grows linearly with the number of attributes in the access structure. The authors in [EMN$^+$09] proposed a CP-ABE scheme with constant ciphertext size, but, without supporting wildcards in its access policy. Later, the ABE scheme was upgraded to support wildcards in [ZH10], however, only with 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.

Ciphertext Size. 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 WIBE scheme [ACD$^+$06] produces 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 scheme. With the approach that the ciphertext contains all information required to conversion, it is not clear how to construct a WIBE with constant size ciphertext.

Using a generic construction described in [ACD$^+$06], WIBE with a constant size ciphertext can be constructed from a hierarchical identity based encryption scheme (HIBE) with a constant sized ciphertext. However, this WIBE scheme has a secret key size that is exponential in the depth of the pattern. Briefly, a user stores secret keys for every possible matching pattern such that the user directly decrypts a ciphertext with a suitable key. For example, given an identity pattern (ID_1, ID_2), a user stores keys for all four matching patterns: (ID_1, ID_2), ($*$, ID_2), (ID_1, $*$), ($*$, $*$). Thus, the secret key size is exploded up to 2^L which is impractical to be used in small embedded devices.

Intuitions. Our construction is based on the variant of the BBG-HIBE[BBG05]. The BBG-HIBE scheme has the advantage of constant-sized ciphertexts and our WIBE scheme keeps this advantage while allowing wildcards in a pattern in encryption. The WIBE scheme by Abdalla et al.[ACD$^+$06] also modifies a BBG-HIBE's ciphertext generation to include some extra data related to each

wildcard. This extra data is used for each user to convert the ciphertext to match their specific identity pattern. However, this approach seems to be difficult to avoid the message overhead required for each wildcard. In the WIBE scheme[ACD+06], a pattern with w wildcards leads to a ciphertext with $w + 3$ group elements. Our approach reverses the method of Abdalla et al.'s: each user converts the identity pattern of the secret key to match the ciphertext. In this method, each user must store an extra data 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 additional elements. The benefit of this approach is that the extra values for each wildcard do not have to be separately delivered as in [ACD+06] because they are not used for conversion of each corresponding identity string any more. From this observation, our WIBE scheme reduces the extra data values for each wildcard in the ciphertext as a single element. The resulting scheme is secure under the same cryptographic assumption as the BBG-HIBE scheme. The details of the construction are described in section 4.

Contributions. In this paper, we propose a new WIBE 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 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 [BBG05] that contains three group elements for its ciphertext.
- The proposed SWIBE also improves decryption performance of WIBE. Much of the decryption overhead in the existing WIBE is in the conversion operation of wildcards in a ciphertext into identity strings of a user's secret key. While the WIBE [ACD+06] converts a ciphertext to another ciphertext for a specific matching identity strings, the proposed SWIBE replaces any identity string by a wildcard; this method reduces point multiplications (i.e., exponentiations) required in the previous WIBE and speeds up the decryption.
- The proposed SWIBE allows wildcards in the key delegation process as well as in the encryption procedure, naturally subsuming the generalized key derivation of wicked-IBE [AKN07] and distributing the key management overhead.
- We formally prove the selective CPA-security of the proposed SWIBE scheme under the L-BDHE assumption. We also extend it to be a CCA secure SWIBE scheme.

Table 1 compares the HIBE scheme [BBG05] (that does not support wildcards as identities), the WIBE scheme [ACD+06], HIBE with the generalized key delegation (wicked-IBE) [AKN07], WIBE scheme with generalized key delegation (WW-IBE) [ACP12], constant-size ciphtertext policy attribute-based encryption (CCP-ABE) [ZH10], and the proposed SWIBE scheme subsuming wildcards as

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 group elements, Enc = Encryption, and Der = Key derivation.*

	HIBE[BBG05]	**WIBE[ACD+06]**	**wicked-IBE[AKN07]**	**WW-IBE[ACP12]**	**CCP-ABE**[ZH10]	**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

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 time overheads in WW-IBE and CCP-ABE are 10 times and 650 times compared with the proposed approach.

This paper is organized as follows: section 2 introduces the basic definitions and complexity assumptions. In section 3, we formally define the wildcard identity-based encryption as a universal primitive. In section 4 we explain the main idea of the proposed scheme and how to construct it in details, along with the security proof. Section 5 extends it to be CCA secure, and section 6 proposes a new version of our scheme in composite order to obtain full security instead of selective security. In section 7, we show the experimental results and in section 8, we conclude.

2 Background

2.1 Identity-based Encryption

An identity-based encryption (IBE) scheme is a tuple of algorithm $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ providing the following functionality. The trusted authority runs Setup to generate a key pair (pp, msk). It publishes public parameter pp and keeps master secret key msk privately. When a user with identity ID

wishes to become part of the system, the trusted authority generates a user decryption key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$, and sends this key over a secure and authenticated channel to the user. To send an encrypted message m to the user with identity ID, the sender computes the ciphertext $C \xleftarrow{\$} \mathsf{Enc}(pp, ID, \mathsf{m})$, which can be decrypted by the user as $\mathsf{m} \leftarrow \mathsf{Dec}(d_{ID}, C)$. We refer to [BF03] for details on the security definitions for IBE schemes.

2.2 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 $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ 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 $sk_{ID'} \xleftarrow{\$} \mathsf{KeyDer}(sk_{ID}, ID')$. Note that by interactively applying the KeyDer algorithm, user ID can derive secret keys for any of its descendants $ID' = (P_1, \ldots, P_{l+\delta})$, $\delta \geq 0$. We use the overloaded notation $sk_{ID'} \xleftarrow{\$} \mathsf{KeyDer}(sk_{ID}, (P_{l+1}, \ldots, P_{L+1}))$ to denote this process. 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 [BBG05] for details on the security definitions for HIBE schemes.

2.3 Bilinear Groups and Pairings

We briefly review the necessary facts about bilinear maps and bilinear map groups. We use the following standard notation [Jou04, BF03].

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. $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map.

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two groups as above. A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ 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 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ as above.

2.4 Computational Complexity Assumptions

Security of our system is based on a complexity assumption called the bilinear Diffie-Hellman Exponent assumption (BDHE) used in [BBG05], which is a natural extension of bilinear-DHI assumption previously used in [BB11]. Let $\mathbb{G}$ be a bilinear group of prime order p. The L-BDHE problem in $\mathbb{G}$ is stated as follows: 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

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

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 1. *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 Model

A scalable wildcarded identity based scheme (SWIBE) 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 [ACD+06].

Definition 2. *A pattern P is a vector $(P_1, \ldots, P_L) \in (Z_p^* \cup \{*\})^L$, where $*$ is a special wildcard symbol, p is a q-bit prime number, and L is the maximal depth of the SWIBE scheme.*[3]

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

Definition 3. *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 4. *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 complementary 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 5. $W(P)$ *is the set containing all wildcard indices in a pattern* P.

Definition 6. $\overline{W}(P)$ *is the complementary set containing all non-wildcard indices in a pattern* P.

A scalable wildcarded identity-based encryption $\mathcal{SWIBE}$ 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 for pattern P.

$\mathsf{Decrypt}(sk_P, C, P')$ takes as input user secret key sk_P for pattern $P = (P_1, \ldots, P_L)$ and ciphertext C for pattern P'. Any user in possession of the secret key for a pattern P that matches P' can decrypt the ciphertext using sk_P, and the algorithm outputs message m.

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

SECURITY. We define the security of a scalable wildcarded identity based encryption scheme similar to [ACD+06], but additionally considering the key delegation property. An adversary is allowed to choose an arbitrary pattern and query secret keys corresponding to the pattern. The adversary is not allowed to query 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.

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.

Key derivation queries:1: Adversary $\mathcal{A}$ issues key derivation queries $q_{K_1}, \ldots, q_{K_m}$ in which a key derivation query consists of a pattern $P' \in (Z_p^* \cup \{*\})^L$, and challenger $\mathcal{C}$ responds with $sk_{P'} \xleftarrow{\$} \mathsf{KeyDer}(sk_P, P')$.

Query phase1: Adversary $\mathcal{A}$ issues decryption queries $q_{D_1}, \ldots, q_{D_n}$ in which a decryption query consists of a pattern $P' \in (Z_p^* \cup \{*\})^L$ and ciphertext C, next challenger $\mathcal{C}$ responds with a message $M \xleftarrow{\$} \mathsf{Decrypt}(C, P')$.

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

Key derivation queries:2: Attacker $\mathcal{A}$ continues to issue key derivation queries $q_{K_{m+1}}, \ldots, q_{q_K}$ same as Key derivation queries:1 where query pattern $P' \not\approx P^*$. $\mathcal{C}$ responds as in key derivation query 1.

Query phase2: Attacker $\mathcal{A}$ continues to issue decryption queries $q_{D_{n+1}}, \ldots, q_{q_D}$ same as Query phase1. It should be satisfied that queried ciphertext $C \neq C^*$. $\mathcal{C}$ responds as in query phase 1.

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

Definition 7. *A $\mathcal{SWIBE}$ is $(t, q_K, q_D, \epsilon, L)$ IND-ID-CCA-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 described above.*

We also define IND-ID-CPA-secure similar as IND-ID-CCA game except forbidden all decryption queries.

Definition 8. *A $\mathcal{SWIBE}$ is $(t, q_K, 0, \epsilon, L)$ IND-ID-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-ID-CPA game described above.*

SELECTIVE-IDENTITY SECURITY. A weaker selective-identity (sID) security notion IND-sID-CPA is defined analogously to the 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.

4 The Proposed Scheme

In this section, we describe a new scalable wildcarded identity based encryption scheme (SWIBE). Since our SWIBE is based on the BBG-HIBE scheme proposed by Boneh et al.[BBG05], we first overview the BBG-HIBE protocol. And then we explain our idea to allow wildcards as identities in encryption at the cost of increasing the secret key size. Finally, we illustrate our SWIBE protocol.

4.1 Overview: BBG-HIBE and our idea

The BBG-HIBE scheme [BBG05] is described as follows:

$\mathsf{Setup}(L)$: L indicates the maximum hierarchy depth. 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 parameters and the master secret key are given by

$$pp = (g, g_1, g_2, g_3, h_1, h_2, \ldots, h_L), msk = g_2^\alpha.$$

$\mathsf{KeyDer}(sk_{P_{|l-1}}, P)$: To compute the secret key sk_P for an identity $P = (P_1, \ldots, P_l) \in (\mathbb{Z}_p^*)^l$ where $l \leq L$ from the master secret key, a random $r \xleftarrow{\$} \mathbb{Z}_p^*$ is chosen, then secret key $sk_P = (a_1, a_2, b_{l+1}, \cdots, b_L)$ for P is constructed as

$$a_1 = g_2^\alpha (g_3 \cdot \prod_{i \in [1,\cdots,l]} h_i^{P_i})^r, \quad a_2 = g^r, \{b_i = h_i^r\}_{i \in [l+1,\cdots,L]}$$

The private key for P can be generated incrementally, given a private key for the parent identity $P_{|l-1} = (P_1, \ldots, P_{l-1}) \in (\mathbb{Z}_p^*)^{l-1}$. Let $sk_{P_{|l-1}} = (a'_1, a'_2, b'_l, \cdots, b'_L)$ be the private key for $P_{|l-1}$. To generate sk_P, pick a random $t \in \mathbb{Z}_p^*$ and output

$$a_1 = a'_1 \cdot (b'_l)^{P_l} \cdot (g_3 \cdot \prod_{i \in [1,\cdots,l]} h_i^{P_i})^t, \quad a_2 = a'_2 \cdot g^t,$$
$$\{b_i = b'_i \cdot h_i^t\}_{i \in [l+1,\cdots,L]}$$

$\mathsf{Encrypt}(pp, P, M)$: To encrypt a message $m \in \mathbb{G}_1$ to pattern $P = (P_1, \ldots, P_l)$, choose $s \xleftarrow{\$} \mathbb{Z}_p^*$, and compute

$$C = (g^s, \quad (g_3 \cdot \prod_{i \in [1,\cdots,l]} h_i^{P_i})^s, \quad M \cdot e(g_1, g_2)^s)$$

$\mathsf{Decrypt}(sk_P, C)$: Consider an identity pattern $P = (P_1, \cdots, P_l)$. To decrypt a given ciphertext $C = (C_1, C_2, C_3)$ with private key $sk_P = (a_1, a_2, b_{l+1}, \cdots, b_L)$, output

$$C_3 \cdot \frac{e(a_2, C_2)}{e(C_1, a_1)} = M$$

The BBG-HIBE scheme [BBG05] has the advantage of constant-sized ciphertexts. The ciphertext consists of only three elements: $(g^s, (g_3 \cdot h_1^{P_1} \cdots h_l^{P_l})^s, M(g, g_2)^{s\alpha})$. The secret key is composed of two types of keys for its purposes: decryption and key delegation. To decrypt a ciphertext, two elements in the secret key are involved with: $(g_2^\alpha (g_3 \cdot h_1^{P_1} \cdots h_l^{P_l})^r, g^r)$. A pattern $P = (P_1, \cdots, P_l)$ is combined as a single element in the secret key. We observe that each identity string of a pattern in the secret key can be replaced by another identity by multiplication because it can be canceled by multiplying its inverse. For instance, it is possible to compute $g_2^\alpha (g_3 h_1^{P'_1} \cdots h_l^{P_l})^r$ by multiplying $h_1^{(P'_1 - P_1)r}$ to $g_2^\alpha (g_3 \cdot h_1^{P_1} \cdots h_l^{P_l})^r$.

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 $\mathbf{c_i} = (h_i^w / h_i^{P_i})^r$ 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. From this extra secret key c_i and the values w and P_i, it is possible to compute $h_i^r = \mathbf{c_i}^{1/(w-P_i)}$; this leads to extract the top level secret key $g_2^\alpha g_3^r$ from $g_2^\alpha (g_3 h_1^{P_1} \cdots h_l^{P_l})^r$ because $(h_1^{P_1} \cdots h_l^{P_l})^r$ can be generated from $h_1^r, \cdots, h_l^r$.

To solve the problem, we randomize the wildcard part using an independent random value $t \in \mathbb{Z}_p$. Thus, the extra key value is revised as $\mathbf{c_i} = h_i^{wt} / h_i^{P_i r}$ 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 $\mathbf{c_1 c_3}$. 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 $(h_1^w h_3^w)^s$. This additional element is used to cancel out the wildcard part of the 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.

The key delegation is orthogonal to the wildcard support for 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 section 4.2 with $w = 1$ and we prove the security of the proposed scheme in section 4.3.

4.2 SWIBE Construction

We present the SWIBE scheme 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

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

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

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

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 $C = (C_1, C_2, C_3, C_4)$

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

Decrypt(sk_P, C, P'): Consider patterns P and $P' \in (Z_p^* \cup \{*\})^L$, where P is a key pattern and P' is a ciphertext pattern. To decrypt a given ciphertext $C = (C_1, C_2, C_3, C_4)$ with private key $sk_P = (a_1, a_2, a_3, b, c, d)$, 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

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

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 $d_i = \frac{h_i^t}{h_i^{P_i r}}$,

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

Since $\mathsf{W}_{P'\overline{P}} \cup \mathsf{W}_{\overline{P}'\overline{P}} = \overline{W}(P)$ and $\mathsf{W}_{P'\overline{P}} \cap W_{\overline{P}'\overline{P}} = \emptyset$, $\prod_{i \in \overline{W}(P)} h_i^{P_i} \cdot \prod_{i \in \mathsf{W}_{P'\overline{P}}} h_i^{-P_i}$ $= \prod_{i \in \mathsf{W}_{\overline{P}'\overline{P}}} h_i^{P_i}$. Similarly, since $\mathsf{W}_{P'P} \cup \mathsf{W}_{P'\overline{P}} = W(P')$ and $\mathsf{W}_{P'P} \cap \mathsf{W}_{P'\overline{P}} = \emptyset$, $\prod_{i \in \mathsf{W}_{P'P}} h_i^t \cdot \prod_{i \in \mathsf{W}_{P'\overline{P}}} h_i^t = \prod_{i \in W(P')} h_i^t$. Therefore,

$$a_1' = g_2^{\alpha}(g_3 \cdot \prod_{i \in \mathsf{W}_{\overline{P}'\overline{P}}} h_i^{P_i} \cdot \prod_{i \in \mathsf{W}_{\overline{P}'P}} h_i^{P_i'})^r \cdot \prod_{i \in W(P')} h_i^t.$$

Since $P \approx P'$, $P_i = P_i'$ for $i \in \mathsf{W}_{\overline{P}'\overline{P}}$.

$$a_1' = g_2^{\alpha}(g_3 \cdot \prod_{i \in \mathsf{W}_{\overline{P}'\overline{P}}} h_i^{P_i'} \cdot \prod_{i \in \mathsf{W}_{\overline{P}'P}} h_i^{P_i'})^r \cdot \prod_{i \in W(P')} h_i^t.$$

Since $\mathsf{W}_{\overline{P}'\overline{P}} \cup \mathsf{W}_{\overline{P}'P} = \overline{W}(P')$ and $\mathsf{W}_{\overline{P}'\overline{P}} \cap \mathsf{W}_{\overline{P}'P} = \emptyset$, $\prod_{i \in \mathsf{W}_{\overline{P}'\overline{P}}} h_i^{P_i'} \cdot \prod_{i \in \mathsf{W}_{\overline{P}'P}} h_i^{P_i'}$ $= \prod_{i \in \overline{W}(P')} h_i^{P_i'}$.

$$a_1' = g_2^{\alpha}(g_3 \cdot \prod_{i \in \overline{W}(P')} h_i^{P_i'})^r \cdot \prod_{i \in W(P')} h_i^t.$$

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

4.3 Security Proof

In this section, we show an IND-sID-CPA-security of the SWIBE scheme in the standard model.

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 $(t', q_K, 0, \epsilon, L)$ 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 $g_3 = g^\delta \prod_{i \in \overline{W}(P^*)}^{L} y_{L-i+1}^{P_i^*}$.

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 $P^* \notin_* P$. That means that there exists an index $k \in \overline{W}(P)$ 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

$$
\begin{aligned}
a_1 &= g_2^\alpha \cdot (g_3 \prod_{i \in \overline{W}(P) h_i^{P_i}})^r; a_2 = g^r; a_3 = g^t \\
b &= \{b_i = h_i^r\}_{i \in W(P)} \\
c &= \{c_i = h_i^t\}_{i \in W(P)} \\
d &= (d_i = h_i^t / h_i^{P_i^* r})_{i \in \overline{W}(P)}
\end{aligned}
$$

We have

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

We split this term up into two factors $A \cdot Z$, where $A = (y_{L-k+1}^{P_k^* - P_k})^r$ is the third product only. 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,

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

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 $i \in \overline{W}(P)$,

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

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.

5 Extension to CCA Security

We extend the semantically secure SWIBE scheme using the similar technique in [CHK04] to obtain chosen ciphertext security. Given a strong one-time signature scheme ($SigKeygen$, $Sign$, $Verify$) with a verification key which is a q-bit string, we enable construction of an L-level wildcard identity-based encryption with constant size ciphertext Π = (Setup, KeyDer, Encrypt, Decrypt) secure against chosen-ciphertext attacks using the $(L+1)$-level Π' = (Setup′, KeyDer′ Encrypt′, Decrypt′) semantically secure SWIBE. 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'. Thus, the secret key sk_P for P in Π is the secret key $sk_{P'}$ in Π'. Recall that $sk'_{P'}$ can generate secret keys of all descendants

of node P'. When encrypting a message m to $P_C = (P_{C_1}, \cdots, P_{C_L})$ in Π, the sender generates a q-bit verification key $V_{sig} \in \mathbb{Z}_p^*$ and then encrypts m to the $P'_C = (P_{C_1}, \cdots, P_{C_L}, V_{sig})$ using Π'. L-level Π is constructed using $(L+1)$-level Π' and a one-time signature scheme as follows:

$\mathsf{Setup}(L)$ 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 $pp \leftarrow (g, g_1, g_2, g_3, h_1, \cdots, h_L)$ and the master secret key is $msk \leftarrow msk'$.

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

$\mathsf{Encrypt}(pp, P, m)$ runs $SigKeyGen(1^L)$ 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 $C \leftarrow \mathsf{Encrypt}'(pp', P', m)$ and $\sigma \leftarrow Sign(K_{sig}, C)$, and output $CT = (C, \sigma, V_{sig})$

$\mathsf{Decrypt}(sk_P, CT, P_C)$: Let $CT = (C, \sigma, V_{sig})$.

1. Verify that σ is the valid signature of C under the key V_{sig}. If invalid, output $\perp$.
2. Otherwise, check $P \approx P_C$, generate $sk_{P'} \leftarrow \mathsf{KeyDer}'(sk_P, P')$ for $P' = (P_1, \cdots, P_L, V_{sig})$, and run $\mathsf{Decrypt}'$ $(sk_{P'}, C, P_C)$ to extract the message.

Theorem 2. *Let $\mathbb{G}$ be a bilinear group of prime order p. The above SWIBE Π is $(t, q_K, q_D, \epsilon_1 + \epsilon_2, L)$ CCA-secure assuming the SWIBE Π' is $(t', q'_K, 0, \epsilon_1, L+1)$ semantically secure in $\mathbb{G}$ and signature scheme is (t'', ϵ_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, $\mathcal{A}$, such that $|AdvBr_{\mathcal{A},\Pi} - 1/2| > \epsilon_1 + \epsilon_2$. We build an algorithm $\mathcal{B}$, that has advantage $|AdvBr_{B,\Pi'} - 1/2| > \epsilon_1$ in $\mathbb{G}$. Algorithm $\mathcal{B}$ proceeds as follows.

Setup: $\mathcal{B}$ gets the public parameter pp of Π' and also gets secret keys $sk'_{P'}$ for $P' \not\approx S^{**}$ from challenger $\mathcal{C}$.

Since Π' can generate secret keys in a compressed way using $*$, wlog, P' can be categorized into the following two formats:

1. $P' = (P_1, \cdots, P_L, *)$ for $P \not\in_* P^*$
2. $P' = (P_1, \cdots, P_L, V_{sig})$ for $P \in_* P^*$ and $V_{sig} \neq V^*_{sig}$.

$\mathcal{B}$ responds with pp and secret keys $sk'_{P'}$ of the first type of P'. (Recall that the secret key $sk_P = sk'_{P'}$ where $P' = (P_1, \cdots, P_L, *.)$ The secret keys $sk'_{P'}$ of the second type of P' are used to respond to the decryption queries of $\mathcal{A}$ as described in the below.

$\mathsf{Query\ phase1}$: Algorithm $\mathcal{A}$ issues decryption queries. Let (P, CT) be a decryption query where $P \in_* P^*$ and $P \approx P_C$ where P_C is a pattern of ciphertext. Let $CT = (\ (C_1, C_2, C_3, C_4), \sigma, V_{sig})$. Algorithm $\mathcal{B}$ responds as follows:

1. Run $Verify$ to check the signature σ on (C_1, C_2, C_3, C_4) using verification key V_{sig}. If the signature is invalid $\mathcal{B}$ responds with $\perp$.
2. If $V_{sig} = V^*_{sig}$, a *forge* event happens, algorithm $\mathcal{B}$ outputs a random bit $b \xleftarrow{\$} \{0,1\}$, and aborts the simulation.
3. Otherwise, $\mathcal{B}$ decrypts the ciphertext CT using the second type of secret keys. Since $V_{sig} \neq V^*_{sig}$, $\mathcal{B}$ can query the key generation query for $P' = (P_1, \cdots, P_L, V_{sig})$ which is second type pattern. Using $sk_{P'}$, B can decrypt $m \leftarrow \mathsf{Decrypt}'(sk_{P'}, (C_1, C_2, C_3, C_4), P'_C)$ where $P'_C = (P_1, \cdots, P_L, V_{sig})$ since $P' = P'_C$.

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 (CT_b) from $\mathcal{C}$. To generate challenge for $\mathcal{A}$, algorithm $\mathcal{B}$ computes C^* as follows:

$$\sigma^* \leftarrow Sign(CT_b, K^*_{sig})$$
$$C^* \leftarrow (CT_b, \sigma^*, V^*_{sig})$$

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

Query phase2: Same as in query phase1 except can not decrypt query for C*.

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

We see that algorithm $\mathcal{B}$ can simulate all queries to run $\mathcal{A}$. $\mathcal{B}$'s success probability as follows:

$$|AdvBr_{B,\Pi'} - \frac{1}{2}| \geq |AdvBr_{\mathcal{A},\Pi} - \frac{1}{2}| - Pr[forge] > (\epsilon_1 + \epsilon_2) - Pr[forge]$$

To conclude the proof of Theorem 2 it remains to bound the probability that $\mathcal{B}$ aborts the simulation as a result of *forge*. We claim that $Pr[forge] < \epsilon_2$. Otherwise one can use $\mathcal{A}$ to forge signatures with probability at least ϵ_2. Briefly, we can construct another simulator that knows the private key, but receives K^*_{sig} as a challenge in an existential forgery game.

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.

6 Fully Secure Scheme

In this section, we present a new version of SWIBE based on composite order bilinear groups, to obtain full security (IND-ID-CPA) instead of selective security (IND-sID-CPA). In the original SWIBE scheme (section 4), the security proof was limited to the selective security, since the scheme was bound to a

single group. In this case, if the challenge pattern is not committed before the game, the reduction algorithm cannot simulate secret keys (without knowing the secret factor α) for key queries from the adversary. To resolve this issue, we now construct the scheme with parameters from multiple subgroups similar to [ACP12], using the fact that it is difficult to determine from which group the element came from. When using the subgroup assumptions, for each pattern, the reduction can both simulate the secret key in one subgroup and embed the problem in another subgroup. Therefore, the full security can be achieved. We first introduce the definition and features of composite order bilinear groups. Then we show the decisional subgroup assumptions from Lewko and Waters (LW.1-LW.3), and present our composite order (fully-secure) SWIBE which also has a constant-size ciphertext. Finally, we prove the full security of our composite order SWIBE based on LW assumptions.

6.1 Composite Order Bilinear Groups

We describe the composite order bilinear groups as in [ACP12]. We use groups whose order is a product of three primes and a generator G which takes as input security parameter λ and outputs a description $\mathcal{G} = (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$ where p_1, p_2, p_3 are distinct primes of $\Theta(\lambda)$ bits, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order N. For $a, b, c \in \{1, p_1, p_2, p_3\}$, we denote by $\mathbb{G}_{abc}$ the subgroup of order abc. From the fact that the group is cyclic, it is simple to verify that if g and h are group elements of different order (and thus belonging to different subgroups), then $e(g, h) = 1$.

6.2 Complexity Assumptions

We first restate the complexity assumptions we use, which was originally introduced by Lewko and Waters in [LW10] and used in [ACP12]. All these assumptions can be seen as variants of the decisional subgroup assumption.

Assumption LW.1: For a generator G of bilinear settings, first pick a bilinear setting $\mathcal{G} \xleftarrow{\$} \mathsf{G}(1^\lambda)$ and then pick $g_1, T_1 \xleftarrow{\$} \mathbb{G}_{p_1}, g_3 \xleftarrow{\$} \mathbb{G}_{p_3}, T_2 \xleftarrow{\$} \mathbb{G}_{p_1p_2}$ and set $D = (\mathcal{G}, g_1, g_3)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be: $\mathsf{Adv}^{\mathcal{A}}_{LW.1}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

Assumption LW.2: For a generator G of bilinear settings, first pick a bilinear setting $\mathcal{G} \xleftarrow{\$} \mathsf{G}(1^\lambda)$ and then pick $g_1, X_1 \xleftarrow{\$} \mathbb{G}_{p_1}, X_2, Y_2 \xleftarrow{\$} \mathbb{G}_{p_2}, g_3, X_3 \xleftarrow{\$} \mathbb{G}_{p_3}, T_1 \xleftarrow{\$} \mathbb{G}_{p_1p_3}, T_2 \xleftarrow{\$} \mathbb{G}_{p_1p_2p_3}$ and set $D = (\mathcal{G}, g_1, g_3, X_1X_2, Y_2X_3)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be: $\mathsf{Adv}^{\mathcal{A}}_{LW.2}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

Assumption LW.3: For a generator G of bilinear settings, first pick a bilinear setting $\mathcal{G} \xleftarrow{\$} \mathsf{G}(1^\lambda)$ and then pick $\alpha, \delta, s \xleftarrow{\$}$
$Z_N, g_1 \xleftarrow{\$} \mathbb{G}_{p_1}, g_2, X_2, Y_2 \xleftarrow{\$} \mathbb{G}_{p_2}, g_3 \xleftarrow{\$} \mathbb{G}_{p_3}, T_2 \xleftarrow{\$} \mathbb{G}_T$ and set $T_1 = e(g_1, g_1^\delta)^{\alpha s}$

and $D = (\mathcal{G}, g_1, g_2, g_3, g_1^{\alpha} X_2, g_1^{\delta}, g_1^{s} Y_2)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be: $\mathsf{Adv}^{\mathcal{A}}_{LW.1}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

6.3 Fully-Secure SWIBE Construction

We present the fully-secure SWIBE with constant size ciphertexts and $O(L)$ size keys, which is based on the composite order bilinear group ($N = p_1 p_2 p_3$). Note that W from subgroup $\mathbb{G}_{p_3}$ is only a blinding factor, to be eliminated when computed in a pairing operation (due to the orthogonal property of composite order bilinear groups).

$\mathsf{Setup}(L)$: L indicates the maximum hierarchy depth. Choose a description of a bilinear group $\mathcal{G} \xleftarrow{\$} \mathsf{G}(1^{\lambda})$ with known factorization and $g_1, k_1 \xleftarrow{\$} \mathbb{G}_{p_1}, g_3 \xleftarrow{\$} \mathbb{G}_{p_3}$. Choose $\alpha \xleftarrow{\$} \mathbb{Z}_N$ and $\{h_i \xleftarrow{\$} \mathbb{G}_{p_1}\}_{i \in [L]}$, and compute $\gamma = g_1^{\alpha}$.

The public parameter is given by

$$pp \leftarrow (N, g_1, \gamma, k_1, g_3, h_1, h_2, \ldots, h_L).$$

A master secret key is defined as $msk = k_1^{\alpha}$.

$\mathsf{KeyDer}(pp, sk_P, P')$: 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 $r, t \xleftarrow{\$} \mathbb{Z}_N$ are chosen. Then random blinding factors from different subgroup $W_1, W_2, W_3 \xleftarrow{\$} \mathbb{G}_{p_3}$, $\{W_{a,i}, W_{d,i} \xleftarrow{\$} \mathbb{G}_{p_3}\}_{i \in \overline{W}(P)}$, and $\{W_{b,i}, W_{c,i} \xleftarrow{\$} \mathbb{G}_{p_3}\}_{i \in W(P)}$ are chosen. Secret key $sk_{P'} = (a'_1, a'_2, a'_3, b', c', d')$ for P' is constructed as

$$\begin{aligned}
a'_1 &= msk \cdot \prod_{i \in \overline{W}(P')} (h_i^{P'_i} \cdot W_{a,i})^r \cdot W_1 \\
a'_2 &= g_1^r \cdot W_2 \\
a'_3 &= g_1^t \cdot W_3 \\
b' &= \{b'_i = h_i^r \cdot W_{b,i}\}_{i \in W(P')} \\
c' &= \{c'_i = h_i^t \cdot W_{c,i}\}_{i \in W(P')} \\
d' &= \{d'_i = h_i^t / h_i^{P'_i r} \cdot W_{d,i}\}_{i \in \overline{W}(P')}
\end{aligned}$$

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 $r', t' \xleftarrow{\$} \mathbb{Z}_N$, and blinding randoms $W'_1, W'_2, W'_3 \xleftarrow{\$} \mathbb{G}_{p_3}$, $\{W'_{a,i}, W'_{d,i}\}_{i \in \overline{W}(P')}$, and $\{W'_{b,i}, W'_{c,i}\}_{i \in W(P')}$ and output $sk_{P'} = (a'_1, a'_2, a'_3, b', c', d')$, where

$$
\begin{aligned}
a'_1 &= a_1 \cdot (\prod_{i \in \overline{W}(P') \cap W(P)} b_i^{P'_i}) \cdot \prod_{i \in \overline{W}(P')} (h_i^{P'_i} \cdot W'_{a,i})^{r'}, \\
a'_2 &= a_2 \cdot g_1^{r'} \\
a'_3 &= a_3 \cdot g_1^{t'} \\
b' &= \{b'_i = b_i \cdot h_i^{r'} \cdot W'_{b,i}\}_{i \in W(P')} \\
c' &= \{c'_i = c_i \cdot h_i^{t'} \cdot W'_{c,i}\}_{i \in W(P')} \\
d' &= \{d'_i = d_i \cdot \frac{h_i^{t'}}{h_i^{P'_i r'}} \cdot W'_{d,i}\}_{i \in \overline{W}(P') \cap \overline{W}(P)} \\
&\quad \cup \{d'_i = \frac{c_i}{b_i^{P'_i}} \cdot \frac{h_i^{t'}}{h_i^{P'_i r'}} \cdot W'_{d,i}\}_{i \in \overline{W}(P') \cap W(P)}
\end{aligned}
$$

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

$$
\begin{aligned}
C_1 &= g_1^s, & C_2 &= (\prod_{i \in \overline{W}(P)} h_i^{P_i})^s \\
C_3 &= m \cdot e(\gamma, k_1)^s, & C_4 &= (\prod_{i \in W(P)} h_i)^s
\end{aligned}
$$

$\mathsf{Decrypt}(sk_P,\ C,\ P')$: The decryption is similar to the decryption in section 4.2, since the elements in $\mathbb{G}_{p_1}$ will work in an equivalent way compared to the original SWIBE construction. Consider patterns P and $P' \in (Z_p^* \cup \{*\})^L$, where P is a key pattern and P' is a ciphertext pattern. To decrypt a given ciphertext $C = (C_1, C_2, C_3, C_4)$ with private key $sk_P = (a_1, a_2, a_3, b, c, d)$, 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

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

The fact that decryption works is similar to the decryption of original SWIBE construction in section 4.2. Note that the blinding factors W from subgroup $\mathbb{G}_{p_3}$ make no difference, since they will be eliminated in the pairing operations due to the orthogonal property in 6.1.

6.4 Security Proof

In this section, we prove that our fully-secure SWIBE scheme in 6.3 is IND-ID-CPA-secure under the Lewko and Waters assumptions (LW.1-LW.3) in 6.2. The security proof is based on the hybrid games.

Before presenting hybrid games, we define two additional structures used in the proofs of security similar to [LW10, ACP12]:

Semi-Functional (SF) Ciphertext. Let g_2 denote a generator of $\mathbb{G}_{p_2}$. An SF ciphertext is created as follows: first, we use the encryption algorithm to form a normal ciphertext $C' \leftarrow [C'_1, C'_2, C'_3, C'_4]$. We choose random exponents $x \xleftarrow{\$} \mathbb{Z}_N$ and $zc_2, zc_4 \xleftarrow{\$} \mathbb{Z}_N$. Then we set: $C_1 = C'_1 \cdot g_2^x, C_2 = C'_2 \cdot g_2^{x \cdot zc_2}, C_3 = C'_3, C_4 = C'_4 \cdot g_2^{x \cdot zc_4}$.

Semi-Functional (SF) Keys. To create a SF key, a normal secret key $(a'_1, a'_2, a'_3, b', c', d')$ is first created, using the key derivation algorithm with pp, msk, and pattern P. Then randoms are chosen: $y, zk_1, zk_2, zk_3 \xleftarrow{\$} \mathbb{Z}_N$, $(zk_{b,i}, zk_{c,i} \xleftarrow{\$} \mathbb{Z}_N)_{i \in W(P)}$, and $(zk_{d,i} \xleftarrow{\$} \mathbb{Z}_N)_{i \in \overline{W}(P)}$. Then the SF secret key is constructed as follows.

$$
\begin{aligned}
a_1 &= a'_1 \cdot g_2^{y \cdot zk_1}, \quad b = \{b_i = b'_i \cdot g_2^{y \cdot zk_{b,i}}\}_{i \in W(P)} \\
a_2 &= a'_2 \cdot g_2^{y \cdot zk_2}, \quad c = \{c_i = c'_i \cdot g_2^{y \cdot zk_{c,i}}\}_{i \in W(P)} \\
a_3 &= a'_3 \cdot g_2^{y \cdot zk_3}, \quad d = \{d_i = d'_i \cdot g_2^{y \cdot zk_{d,i}}\}_{i \in \overline{W}(P)}
\end{aligned}
$$

Notice that SF ciphertexts can still be decrypted by normal secret keys and SF secret keys can still be used to decrypt normal ciphertext. But an SF ciphertext, for pattern P', and an SF secret key, for pattern P, lead to a random decryption with high probability. In fact the decryption algorithm will compute the blinding factor multiplied by the additional term $e(g_2, g_2)^{xy \cdot (zk_2 \cdot zc_2 + zk_3 \cdot zc_4 - zk_1)}$. If $zk_2 \cdot zc_2 + zk_3 \cdot zc_4 = zk_1$, decryption will still work. Such ciphertexts and secret keys are defined as *nominally* SF. This concept is used during the security proofs, and the nominality is hidden even to an unbounded adversary under the constraints of the security game.

Theorem 3. *If Assumptions LW.1, LW.2, and LW.3 hold, then our fully-secure SWIBE scheme is IND-ID-CPA (or IND-WWID-CPA [ACP12]) secure.*

Proof. We prove the security via sets of hybrid games, and by showing that two games are computationally indistinguishable in each step. For the sketch of our proof, we first define the series of games. Let us assume PPT adversary $\mathcal{A}$ with q key queries. Then we define $q+5$ games between $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows.

- $\mathbf{Game_{real}}$: It is the real IND-ID-CPA (or IND-WWID-CPA [ACP12]) security game.
- $\mathbf{Game_{gen}}$: This game is same as $Game_{real}$, except the fact that all key queries from the adversary is answered by fresh key generation algorithm. In the security proof, we use the term *key generation* for the key delegation from msk, which is equivalent.
- $\mathbf{Game_{pre}}$: It is a same game as $Game_{gen}$, except the fact that the adversary cannot query for keys if the pattern is a *prefix* of the challenge pattern mod p_2. We say that the pattern P is a prefix of P^* mod p_2 if there exists i s.t. $P_i \neq P_i^*$ mod N and $P_i = P_i^*$ mod p_2.
- $\mathbf{Game_0}$: It is a same game as $Game_{pre}$, except the fact that the challenge ciphertext $C_1 \sim C_4$ is given as a SF ciphertext.

- $\mathbf{Game_{1\sim q}}$: For k from 1 to q, $Game_k$ is same as $Game_0$ except the fact that the first k keys are given as SF keys. The other keys are given as normal keys from the key generation algorithm.
- $\mathbf{Game_{rand}}$: This game is same as $Game_q$, but where C_3 of challenge ciphertext has a random element in $\mathbb{G}_T$. Since C_3 is distorted with a random element, the ciphertext is now independent from the message and the adversary can have no advantage.

With the defined games above, we now introduce the outline of our hybrid games as follows.

1. $\mathbf{Game_{real}} = \mathbf{Game_{gen}}$. Since the key generation algorithm has an identical distribution, it is exactly the same to call key delegation or fresh key generation from the adversary's view.
2. $\mathbf{Game_{gen}} \approx \mathbf{Game_{pre}}$. We require that the adversary cannot obtain the key for prefix pattern of challenge ciphertext, to prevent the adversary to find the nominality in SF randoms. We prove the hybrid game by showing that prefix query itself can break the Assumption LW.2.
3. $\mathbf{Game_{pre}} \approx \mathbf{Game_0}$. We construct algorithm $\mathcal{B}$ which breaks Assumption LW.1 with a help of SWIBE CPA distinguisher $\mathcal{A}$. The nature of the challenge ciphertext (normal or SF) is decided by T given from the Assumption LW.1, which indicates that two games are indistinguishable from the adversary's view.
4. $\mathbf{Game_{k-1}} \approx \mathbf{Game_k}$. We construct algorithm $\mathcal{B}$ which breaks Assumption LW.2 with a help of SWIBE CPA distinguisher $\mathcal{A}$. The nature of the k-th key (normal or SF) is decided by T given from the Assumption LW.2, which indicates that two games are indistinguishable from the adversary's view.
5. $\mathbf{Game_q} \approx \mathbf{Game_{rand}}$. We construct algorithm $\mathcal{B}$ which breaks Assumption LW.3 with a help of SWIBE CPA distinguisher $\mathcal{A}$. The nature of the ciphertext (SF or random) is decided by T given from the Assumption LW.3, which indicates that two games are indistinguishable from the adversary's view. This concludes the proof, since the adversary can have no advantage in the final game.

We now give formal proofs for each hybrid game stated above.

1) $\mathbf{Game_{real}} = \mathbf{Game_{gen}}$.
The delegated key from the key delegation algorithm has an identical distribution as the freshly generated key from the key generation algorithm. Therefore, it is straightforward that it is exactly the same whether to provide the key with delegating the previous key or to provide the key from a fresh call to the key generation algorithm.

2) $\mathbf{Game_{gen}} \approx \mathbf{Game_{pre}}$.
We define that pattern $P = (P_1, \cdots, P_k)$ is a prefix of pattern $P^* = (P_1^*, \cdots, P_j^*)$ modulo p_2, if there exists P_i and P_i^* such that $P_i \neq P_i^* \bmod N$ and $P_i = P_i^* \bmod p_2$. In this case, p_2 divides $P_i - P_i^*$, and therefore $a = gcd(P_i - P_i^*, N)$ is a

nontrivial factor of N. Since p_2 divides a, let us set $b = N/a$. There are two cases: (1) b is in $ord(g_1)$, or (2) b is not in $ord(g_1)$ but in $ord(g_3)$. Suppose that case 1 has probability of at least $\epsilon/2$. We describe algorithm $\mathcal{B}$ that breaks Assumption LW.2. $\mathcal{B}$ receives $(\mathcal{G}, g_1, g_3, X_1X_2, Y_2X_3)$ and T and constructs pp and msk by choosing $\alpha, \delta \xleftarrow{\$} \mathbb{Z}_N$ and $\eta_i \xleftarrow{\$} \mathbb{Z}_{N\,i\in[L]}$, and setting $pp \leftarrow (N, g_1, \gamma = g_1^{\alpha}, k_1 = g_1^{\delta}, g_3, \{h_i = g_1^{\eta_i}\}_{i\in[L]})$ and $msk \leftarrow k_1^{\alpha}$. Then $\mathcal{B}$ runs $\mathcal{A}$ on input pp and uses knowledge of msk to answer $\mathcal{A}$'s queries. At the end of the game, $\mathcal{B}$ computes $a = gcd(P_i - P_i^*, N)$, for all Ps which are patterns for the key queries that $\mathcal{A}$ has asked, and for P^* which is the challenge pattern. If $e((X_1X_2)^a, Y_2X_3)$ is the identity element of $\mathbb{G}_T$, $\mathcal{B}$ tests if $e(T^b, X_1x_2)$ is also the identity element of $\mathbb{G}_T$. When the second test is successful, $\mathcal{B}$ declares $T \in \mathbb{G}_{p_1p_3}$. Otherwise, $\mathcal{B}$ declares $T \in \mathbb{G}_{p_1p_2p_3}$. It is from the simple fact that p_2 divides a and $p_1 = ord(g_1)$ divides b. For case 2, $\mathcal{B}$ can break Assumption LW.2 in the same way by exchanging the roles of $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_3}$, i.e., p_1 and p_3.

3) $\mathbf{Game_{pre}} \approx \mathbf{Game_0}$.
We construct algorithm $\mathcal{B}$ that breaks Assumption LW.1 with the help of SWIBE CPA distinguisher $\mathcal{A}$. $\mathcal{B}$ first receives $(\mathcal{G}, g_1, g_3)$ and T from LW.1. Then $\mathcal{B}$ starts the IND-ID-CPA game with $\mathcal{A}$ and simulates $Game_{pre}$ or $Game_0$ depending on the nature of T.

Setup: $\mathcal{B}$ chooses $\alpha, \delta \xleftarrow{\$} \mathbb{Z}_N$, $(\eta_i \xleftarrow{\$} \mathbb{Z}_N)_{i\in[L]}$, then set $pp \leftarrow (g_1, \gamma = g_1^{\alpha}, k_1 = g_1^{\delta}, g_3, \{h_i = g_1^{\eta_i}\}_{i\in[L]})$ and $msk \leftarrow k_1^{\alpha}$.

Key derivation queries: $\mathcal{B}$ can answer all key generation queries from $\mathcal{A}$ since $\mathcal{B}$ knows msk.

Challenge: $\mathcal{A}$ sends $\mathcal{B}$ challenge message $m_0, m_1 \in \{0,1\}^*$ and a challenge pattern $P = (P_1, \cdots, P_l)$ where $0 \le l \le L$. $\mathcal{B}$ flips a coin $\zeta \leftarrow \{0,1\}$ and computes the challenge ciphertext C as follows:

$$C_1 = T, \qquad C_2 = \prod_{i\in\overline{W}(P)} T^{\eta_i \cdot P_i}$$

$$C_3 = m \cdot e(\gamma, T^{\delta}), \quad C_4 = \prod_{i\in W(P)} T^{\eta_i}$$

Notice that if $T \in \mathbb{G}_{p_1}$, then T can be written as g_1^s and C is a normal ciphertext with randomness s. Instead, if $T \in \mathbb{G}_{p_1p_2}$, then T can be written as $g_1^s g_2^x$ and C is SF with randomness $s, x, zc_2 = \eta_i \cdot P_i, zc_4 = \eta_i$.

4) $\mathbf{Game_{k-1}} \approx \mathbf{Game_k}$.
We construct algorithm $\mathcal{B}$ that breaks Assumption LW.2 with the help of SWIBE CPA distinguisher $\mathcal{A}$. $\mathcal{B}$ first receives $(\mathcal{G}, g_1, g_3, X_1X_2, Y_2X_3)$ and T from LW.2. Then $\mathcal{B}$ starts the IND-ID-CPA game with $\mathcal{A}$ and simulates $Game_{k-1}$ or $Game_k$ depending on the nature of T.

Setup: $\mathcal{B}$ chooses $\alpha, \delta \xleftarrow{\$} \mathbb{Z}_N$, $(\eta_i \xleftarrow{\$} \mathbb{Z}_N)_{i\in[L]}$, then set $pp \leftarrow (g_1, \gamma = g_1^{\alpha}, k_1 = g_1^{\delta}, g_3, \{h_i = g_1^{\eta_i}\}_{i\in[L]})$ and $msk \leftarrow k_1^{\alpha}$.

Key derivation queries: There are three cases for the i-th key query with pattern P.

i) case 1: $i < k$. Choose SF randoms $zk_1, zk_2, zk_3 \xleftarrow{\$} \mathbb{Z}_N$, and $(zk_{b,i}, zk_{c,i} \xleftarrow{\$} \mathbb{Z}_N)_{i \in W(P)}$, and $(zk_{d,i} \xleftarrow{\$} \mathbb{Z}_N)_{i \in \overline{W}(P)}$. Then choose random $r, t \xleftarrow{\$} \mathbb{Z}_N, W_1, W_2, W_3 \xleftarrow{\$} \mathbb{G}_{p_3}, \{W_{a,i}, W_{d,i} \xleftarrow{\$} \mathbb{G}_{p_3}\}_{i \in \overline{W}(P)}$, and $\{W_{b,i}, W_{c,i} \xleftarrow{\$} \mathbb{G}_{p_3}\}_{i \in W(P)}$. Then we set the sk_P as follows:

$$
\begin{aligned}
a_1 &= msk \cdot \prod_{i \in \overline{W}(P')} (h_i^{P_i} \cdot W_{a,i})^r \cdot (Y_2 X_3)^{zk_1} \cdot W_1 \\
a_2 &= g_1^r \cdot (Y_2 X_3)^{zk_2} \cdot W_2 \\
a_3 &= g_1^t \cdot (Y_2 X_3)^{zk_3} \cdot W_3 \\
b &= \{b_i = h_i^r \cdot (Y_2 X_3)^{zk_{b,i}} \cdot W_{b,i}\}_{i \in W(P)} \\
c &= \{c_i = h_i^t \cdot (Y_2 X_3)^{zk_{c,i}} \cdot W_{c,i}\}_{i \in W(P)} \\
d &= \{d_i = h_i^t / h_i^{P'_i r} \cdot (Y_2 X_3)^{zk_{d,i}} \cdot W_{d,i}\}_{i \in \overline{W}(P)}
\end{aligned}
$$

By writing Y_2 as g_2^y, we have that this is a properly distributed SF key with randomness y, $zk_{1\sim 3}$, $zk_{b\sim d}$.

ii) case 2: $i > k$. $\mathcal{B}$ runs key generation algorithm using msk.

iii) case 3: $i = k$. To answer the k-th key query, $\mathcal{B}$ chooses $r', t' \xleftarrow{\$} \mathbb{Z}_N$, $W_1, W_2, W_3 \xleftarrow{\$} \mathbb{G}_{p_3}$, $\{W_{a,i}, W_{d,i} \xleftarrow{\$} \mathbb{G}_{p_3}\}_{i \in \overline{W}(P)}$, and $\{W_{b,i}, W_{c,i} \xleftarrow{\$} \mathbb{G}_{p_3}\}_{i \in W(P)}$. Then the sk_P is constructed as follows:

$$
\begin{aligned}
a_1 &= msk \cdot \prod_{i \in \overline{W}(P')} (T^{\eta_i \cdot P_i} \cdot W_{a,i})^{r'} \cdot W_1 \\
a_2 &= T^{r'} \cdot W_2 \\
a_3 &= T^{t'} \cdot W_3 \\
b &= \{b_i = T^{\eta_i \cdot r'} \cdot W_{b,i}\}_{i \in W(P)} \\
c &= \{c_i = T^{\eta_i \cdot t'} \cdot W_{c,i}\}_{i \in W(P)} \\
d &= \{d_i = T^{\eta_i (t' - P_i r')} \cdot W_{d,i}\}_{i \in \overline{W}(P)}
\end{aligned}
$$

Notice that, if $T \in \mathbb{G}_{p_1 p_3}$, T can be written as $g_1^r g_3^w$ and the k-th secret key is a normal key with randomness rr', rt'. Otherwise, if $T \in \mathbb{G}_{p_1 p_2 p_3}$, T can be written as $g_1^r g_2^y g_3^w$ and the k-th secret key is SF with randomness $rr', rt', y, zk_1 = \sum_{i \in \overline{W}(P)} \eta_i P_i, zk_2 = r', zk_3 = t', zk_{b,i} = \eta_i r', zk_{c,i} = \eta_i t', zk_{d,i} = \eta_i (t' - P_i r')$.

Challenge: $\mathcal{A}$ sends $\mathcal{B}$ challenge message $m_0, m_1 \in \{0,1\}^*$ and a challenge pattern $P = (P_1, \cdots, P_l)$ where $0 \le l \le L$. $\mathcal{B}$ flips a coin $\zeta \leftarrow \{0,1\}$ and computes the challenge ciphertext C as follows:

$$C_1 = X_1X_2, \qquad C_2 = \prod_{i\in\overline{W}(P)} (X_1X_2)^{\eta_i \cdot P_i}$$
$$C_3 = m \cdot e(\gamma, (X_1X_2)^{\delta}), \quad C_4 = \prod_{i\in W(P)} (X_1X_2)^{\eta_i}$$

Notice that T can be written as g_1^s, and therefore this is a proper SF ciphertext with randomness $s, x, zc_2 = \sum_{i\in\overline{W}(P)} x\eta_i P_i, zc_4 = \sum_{i\in\overline{W}(P)} x\eta_i$.

Since the k-th secret key pattern is not a prefix of the challenge pattern mod p_2, we have that zk set and zc set are independent and randomly distributed. If $\mathcal{B}$ attempts to test whether the k-th key is SF by using the above procedure (to create an SF ciphertext for k-th secret key pattern), then we will have $zk_2 \cdot zc_2 + zk_3 \cdot zc_4 = zk_1$, which is nominally SF, and thus decryption always works (independently of T).

5) $\mathbf{Game_q} \approx \mathbf{Game_{rand}}$.
We construct algorithm $\mathcal{B}$ that breaks Assumption LW.3 with the help of SWIBE CPA distinguisher $\mathcal{A}$. $\mathcal{B}$ first receives $(\mathcal{G}, g_1, g_2, g_3, g_1^{\alpha}X_2, g_1^{\delta}, g_1^{s}Y_2)$ and T from LW.3. Then $\mathcal{B}$ starts the IND-ID-CPA game with $\mathcal{A}$ and simulates $Game_q$ or $Game_{rand}$ depending on the nature of T.

Setup: $\mathcal{B}$ chooses $(\eta_i \xleftarrow{\$} \mathbb{Z}_N)_{i\in[L]}$, then set $pp \leftarrow (g_1, \gamma = g_1^{\alpha}X_2, k_1 = g_1^{\delta}, g_3, \{h_i = g_1^{\eta_i}\}_{i\in[L]})$.

Key derivation queries: For pattern $P = (P_1, \cdots, P_l)$ where $0 \le l \le L$, $\mathcal{B}$ chooses $zk'_1, zk'_2, zk'_3 \xleftarrow{\$} \mathbb{Z}_N$, and $(zk'_{b,i}, zk'_{c,i} \xleftarrow{\$} \mathbb{Z}_N)_{i\in W(P)}$, and $(zk'_{d,i} \xleftarrow{\$} \mathbb{Z}_N)_{i\in\overline{W}(P)}$. Next it chooses $r, t \xleftarrow{\$} \mathbb{Z}_N$, $W_1, W_2, W_3 \xleftarrow{\$} \mathbb{G}_{p_3}$, $\{W_{a,i}, W_{d,i} \xleftarrow{\$} \mathbb{G}_{p_3}\}_{i\in\overline{W}(P)}$, and $\{W_{b,i}, W_{c,i} \xleftarrow{\$} \mathbb{G}_{p_3}\}_{i\in W(P)}$. Then it creates a SF secret key by setting:

$$a_1 = (g_1^{\alpha}X_2) \cdot \prod_{i\in\overline{W}(P)} (h_i^{P_i} \cdot W_{a,i})^r \cdot g_2^{zk'_1} \cdot W_1$$
$$a_2 = g_1^r \cdot g_2^{zk'_2} \cdot W_2$$
$$a_3 = g_1^t \cdot g_2^{zk'_3} \cdot W_3$$
$$b = \{b_i = h_i^r \cdot g_2^{zk'_{b,i}} \cdot W_{b,i}\}_{i\in W(P)}$$
$$c = \{c_i = h_i^t \cdot g_2^{zk'_{c,i}} \cdot W_{c,i}\}_{i\in W(P)}$$
$$d = \{d_i = h_i^t / h_i^{P'_i r} \cdot g_2^{zk'_{d,i}} \cdot W_{d,i}\}_{i\in\overline{W}(P)}$$

Challenge: $\mathcal{A}$ sends $\mathcal{B}$ challenge message $m_0, m_1 \in \{0,1\}^*$ and a challenge pattern $P = (P_1, \cdots, P_l)$ where $0 \le l \le L$. $\mathcal{B}$ flips a coin $\zeta \leftarrow \{0,1\}$ and computes the challenge ciphertext C as follows:

$$C_1 = g_1^s Y_2, \quad C_2 = \prod_{i \in \overline{W}(P)} (g_1^s Y_2)^{\eta_i \cdot P_i}$$
$$C_3 = m \cdot T, \quad C_4 = \prod_{i \in W(P)} (g_1^s Y_2)^{\eta_i}$$

This sets $zc_2 = \sum_{i \in \overline{W}(P)} \eta_i P_i, zc_4 = \sum_{i \in W(P)} \eta_i$. We note that $g_1^{\eta_i}$ are elements of $\mathbb{G}_{p_1}$, so when η_i are randomly chosen from $\mathbb{Z}_N$, their value mod p_1 and mod p_2 are random and independent. We observe that, if $T = e(g_1, g_1^{\delta})^{\alpha s}$, then the challenge ciphertext is a properly distributed SF with message m_ζ. Otherwise, if $T \xleftarrow{\$} \mathbb{G}_T$, then the ciphertext is an SF with a random message.

Since the adversary has no advantage in $Game_{rand}$ (where ζ is information-theoretically hidden), and by hybrid game $Game_{rand}$ is indistinguishable from $Game_{Real}$, we conclude that the adversary in $Game_{Real}$ has a negligible advantage.

7 Experiment

In this section, we measure the execution times of encryption and decryption of the proposed SWIBE, WIBE [ACD+06], wicked-IBE [AKN07], WW-IBE [ACP12], and CCP-ABE [ZH10]. We have implemented the algorithms based on the PBC (pairing based cryptography) library *a_param* and executed them on Intel Edison with a 32-bit Atom processor 500 MHz and ublinux 3.10.17.

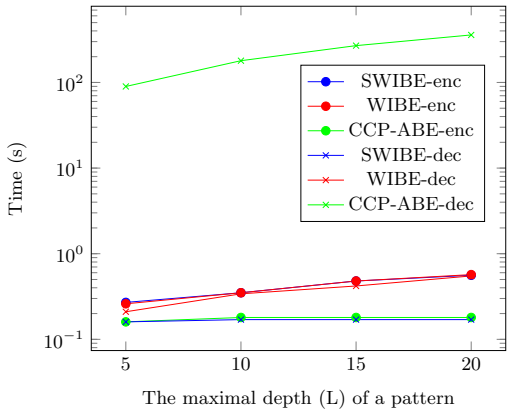

Fig. 1: Encryption and decryption time in SWIBE, WIBE, and CCP-ABE

Figure 1 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.

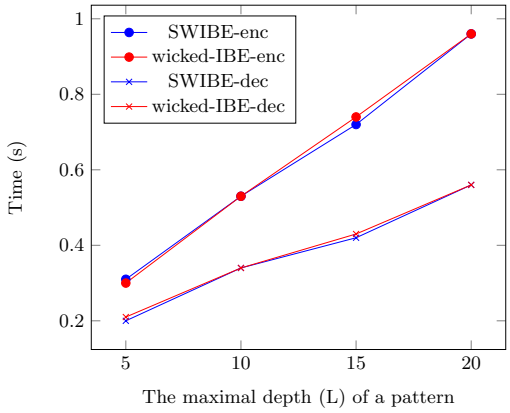

Fig. 2: Encryption and decryption time in SWIBE and wicked-IBE

Figure 2 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 3 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.

8 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

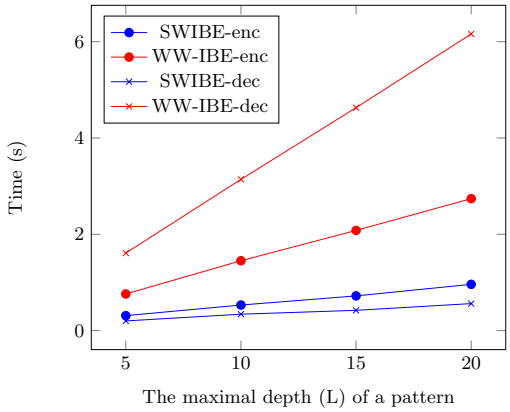

Fig. 3: Encryption and decryption time in SWIBE and WW-IBE

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.

References

ACD+06. Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. Identity-based encryption gone wild. In *International Colloquium on Automata, Languages, and Programming*, pages 300–311. Springer, 2006.

ACP12. Michel Abdalla, Angelo D. Caro, and Duong H. Phan. Generalized key delegation for wildcarded Identity-Based and Inner-Product encryption. *IEEE Trans. Information Forensics and Security*, 7(6):1695–1706, 2012.

AKN07. Michel Abdalla, Eike Kiltz, and Gregory Neven. Generalized key delegation for hierarchical identity-based encryption. In *European Symposium on Research in Computer Security*, pages 139–154. Springer, 2007.

BB11. Dan Boneh and Xavier Boyen. Efficient selective Identity-Based encryption without random oracles. *J. Cryptology*, 24(4):659–693, 2011.

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 440–456. Springer, 2005.

BDNS07. James Birkett, Alexander W. Dent, Gregory Neven, and Jacob C. N. Schuldt. Efficient chosen-ciphertext secure identity-based encryption with

wildcards. In *Australasian Conference on Information Security and Privacy*, pages 274–292. Springer, 2007.

BF01. Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual International Cryptology Conference*, pages 213–229. Springer, 2001.

BF03. Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM journal on computing*, 32(3):586–615, 2003.

BGW05. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Annual International Cryptology Conference*, pages 258–275. Springer, 2005.

BH08. Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 455–470. Springer, 2008.

BSW07. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-Policy Attribute-Based Encryption. pages 321–334. IEEE Computer Society, 2007.

CHK04. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 207–222. Springer, 2004.

EMN+09. Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A Ciphertext-Policy Attribute-Based encryption scheme with constant ciphertext length. In *Information Security Practice and Experience, 5th International Conference, ISPEC*, pages 13–23, 2009.

Jou04. Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *Annual International Cryptology Conference*, pages 306–316. Springer, 2004.

LLWQ14. Weiran Liu, Jianwei Liu, Qianhong Wu, and Bo Qin. Hierarchical Identity-Based broadcast encryption. In *ACISP*, volume 14, pages 242–257. Springer, 2014.

LW10. Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *Theory of Cryptography Conference*, pages 455–479. Springer, 2010.

Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 47–53. Springer, 1984.

Wat05. Brent Waters. Efficient identity-based encryption without random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 114–127. Springer, 2005.

ZH10. Zhibin Zhou and Dijiang Huang. 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*, pages 753–755, 2010.