electronics

MDPI

Article

Scalable Wildcarded Identity-Based Encryption with Full Security

Jiwon Lee [1], Seunghwa Lee [2], Jihye Kim [2,*] and Hyunok Oh [1,*]

1 Department of Information System, Hanyang University, Seoul 04763, Korea; jiwonlee@hanyang.ac.kr
2 Department of Security Enhanced Smart Electric Vehicle, Kookmin University, Seoul 02707, Korea; ttyhgo@kookmin.ac.kr
* Correspondence: jihyek@kookmin.ac.kr (J.K.); hoh@hanyang.ac.kr (H.O.)

Received: 20 July 2020; Accepted: 3 September 2020; Published: 6 September 2020

Abstract: Wildcarded identity-based encryption (WIBE) is an encryption system where one can encrypt messages to multiple users by specifying a pattern, which is a set of identity strings or wildcards. It is a useful primitive for practical applications where users are defined with multiple attributes (or affiliations), such as organization networks or IoT firmware updates. However, the ciphertext size in traditional WIBE schemes are linear to the number of wildcards in the pattern; since the ciphertext size determines the payload in network systems, it degrades the practicality when deployed in transmission-sensitive systems. In this paper, we represent scalable wildcarded identity-based encryption (SWIBE), which achieves a constant-size ciphertext regardless of the number of wildcards (or depth of patterns). the SWIBE scheme also allows the wildcard usage key derivation as well as encryption: a user with wildcarded pattern can delegate keys for the fixed pattern. Compared to the existing WIBE schemes, the SWIBE scheme is the first approach to yield constant-size ciphertext. Moreover, SWIBE also improves encryption time and decryption time while maintaining a key size of $2L$, comparable to the key size of L in WIBE schemes (where L is a depth of the pattern). The experimental results show that the decryption time is 3 to 10 times faster than the existing WIBE schemes, and 650 times faster than the attribute-based encryption with constant-size ciphertext. For the security, we first propose the selective-CPA-secure SWIBE scheme in a prime order bilinear group and extend it to be selective-CCA-secure. Then we also propose a fully-secure SWIBE scheme which can overcome the selective security.

Keywords: wildcard identity based encryption; constant ciphertext; key delegation; pattern

1. Introduction

In modern IoT infrastructure, it is often essential to encrypt data to multiple devices at once on a group-basis, which let the data security require more advanced functionality of multi-encryption. Specifically, IoT devices are often defined and categorized by their own features, such as device types, manufacturers, affiliations, or locations; users (or devices) often need to send secure messages to a group of devices by specifying particular features. Some well-known examples are described as follows.

- In smart city networks, an occupant may need to encrypt command reservations to home devices (e.g., turn-off devices of type: "lamp" and location: "living room").
- In secure firmware updates, the firmware is encrypted to a group of devices which satisfies specific conditions (e.g., update devices of manufacturer: "Tesla").
- In military communications, tactical data are encrypted to classified groups based on specific authorities (e.g., send strategy to devices of affiliation: "Air Force").

When we apply simple encryption technique such as RSA encryption to the IoT data security, the result becomes inefficient: one should encrypt the message to each user one by one, which requires encryption process and different ciphertext as many as the number of receivers. Moreover, the RSA public key system needs to manage a public key infrastructure (PKI) to identify devices, which increases the communication and storage costs. Therefore, in the IoT data security, it is recommended to consider group-based encryption structure rather than a simple standard encryption.

Related works. Considering that the identities are often defined by attributes, attribute-based encryption (ABE) can provide a suitable functionality for secure communications. Ciphertext-policy attribute-based encryption (CP-ABE) [1–3] associates an access policy to each ciphertext, where the access policy is defined by logical combinations (i.e., AND and OR gates) of attribute values. In this setting, the secret key is issued for a set of attributes, and users can decrypt the message only when their attributes satisfy the access policy of the ciphertext. Unfortunately, ABE focuses on more complicated access policy based on variative attributes, while IoT devices are sufficient to handle with fixed identities. To support general access policy, existing ABE schemes either suffer from large ciphertext linearly with the number of attributes or suffer from exponential keys when the ciphertext is fixed as constant.

Identity-based encryption (IBE) [4–6] is a useful primitive where the user's identity (e.g., alice@cs.univ.edu) can be utilized as a public key for encryption, which eliminates a requirement of public key certification such as public key infrastructure (PKI). The IBE can provide an interface for representing the groups or attributes, which is a building block for group-basis secure communications. Later, it advanced to hierarchical identity-based encryption (HIBE) [7], where the identity is defined by a hierarchical set of identity strings such that keys for the identity can be distributed in a hierarchical delegation, i.e., a user at level i can delegate keys for the user at level $i+1$. While traditional IBE requires the certificate authority (CA) to issue keys for the entire users, the HIBE allows each user to share the burden of key delegation to resolve the bottleneck problem. The intuition of key delegation technique in the HIBE system has been advanced to other various cryptographic primitives, such as broadcast encryption [8–10].

Considering the fact that many email addresses consist of affiliations, Abdalla et al. [11] extended HIBE to an advanced primitive called wildcarded identity-based encryption (WIBE) by introducing wildcard ($*$), which can be superceded by any identity string in a set of identity strings. A pattern (or an identity) is defined as a set of multiple identity strings and wildcards, and it efficiently determines a group of identities or a single identity. WIBE utilizes a pattern as a public key for encryption; a single ciphertext encrypted for a pattern (edu, univ, cs, $*$) (corresponding to $*$@cs.univ.edu) can be delivered to multiple identities such as alice@cs.univ.edu and bob@cs.univ.edu.

Abdalla's group proposed three different WIBE systems by extending the existing HIBE schemes. However, all of them suffer from large ciphertext size which is at least $O(L)$ where L is the maximum depth of pattern (i.e., the number of identity strings). Birkett et al. [12] devised compilation techniques called WIB-KEM, which can convert any CPA-secure WIBE into CCA-secure identity-based key encapsulation mechanisms with wildcards. They achieved more efficient CCA-secure WIBE systems by applying their techniques to the Abdalla's CPA-secure WIBE schemes [11]. However, the ciphertext size still remains as $O(L)$, which is as large as the underlying WIBE schemes. Later, Abdalla's group introduced IBE with wildcarded key derivation (WKD-IBE) [13], which can include wildcarded pattern in the key derivation of HIBE. They combined the WKD-IBE with their WIBE [11] construction, where wildcarded pattern can be included both in encryption and key derivation, and upgraded the notion of WIBE to WW-IBE [14]. Still, the ciphertext size in WW-IBE [14] is not improved from the original WIBE [11], which is not scalable in IoT systems.

Ciphertext size. The ciphertext size is an important factor, since ciphertext is an actual payload which is transmitted to the network in real applications. Unfortunately, ciphertext size in existing WIBE schemes [11,14] are not constant; it increases linear to the maximum depth of pattern. The main reason is that the WIBE construction let the decryptor transform the ciphertext to his own matching pattern. In brief, the ciphertext should include additional parameters for the wildcarded pattern so that the wildcard can be transformed to the matching key element, based on the key delegation technique from BBG-HIBE [7].

As an alternative approach, the authors of WIBE [11] describes a generic construction of WIBE from HIBE, which can maintain the constant-size ciphertext of the HIBE ingredient. However, this technique results in an excessive secret key size, exponential to the depth of pattern. The main idea in this generic construction is to let the user store secret keys for every possible cases for decryption, instead of transforming the ciphertext. For example, to decrypt the ciphertext encrypted for pattern (ID_1, ID_2), a user stores keys for every possible matching patterns: (ID_1, ID_2), $(*, ID_2)$, $(ID_1, *)$, $(*, *)$. This leads to the secret key size exploded up to 2^L which is not reasonable to be used, e.g., in small IoT devices.

Solution. The main barrier of ciphertext size in WIBE is that the decryptor needs to delegate the ciphertext to his own matching pattern. For instance, if a decryptor with an identity (ID_1, ID_2) decrypts a ciphertext for $(ID_1, *)$, he transforms the $*$ to ID_2 by the help of additional parameters in order to match his own pattern (ID_1, ID_2). This lets the ciphertext include extra parameters related to each wildcard, which cannot be compressed to a constant value.

The problem can be resolved by reversing the transformation method: instead of delegating the wildcard to the matching pattern, the decryptor may prune his own pattern to match the wildcard. Specifically, the wildcard is considered as a distinct pattern, and additional pruning keys corresponding to the user's patterns can be provided so that any pattern can be transformed to the wildcard pattern. For example, if a decryptor with an identity (ID_1, ID_2) decrypts a ciphertext for $(ID_1, *)$, he transforms his ID_2 to $*$ by using his own pruning key, in order to match the ciphertext pattern $(ID_1, *)$. Note that the pruning key is only required for the depth of pattern L, which just doubles the secret key. As a result, the ciphertext does not require any additional parameters for delegation and the size can remain constant.

Scalable WIBE. Based on the idea, Kim et al. [15] proposed scalable wildcarded identity-based encryption (SWIBE), which achieves a constant-size ciphertext while maintaining the order of other parameters. SWIBE can efficiently cover the multi-encryption requirements in the IoT examples: with a single encryption and a constant ciphertext, one can encrypt to any group of multiple users by specifying the attributes of receivers.

SWIBE scheme also allows wildcarded patterns both in encryption and key derivation as in WW-IBE [14], i.e., one can encrypt a message to multiple users by using wildcarded patterns, or delegate keys to other users if the identity includes wildcards. Compared to the WW-IBE, SWIBE also improves decryption time, since a user in WW-IBE requires a delegation of the given ciphertext to his own pattern while SWIBE allows a user to decrypt the ciphertext immediately. The practicality of SWIBE is justified by implementing the protocol on a real IoT device, and comparing experimental results with the existing works.

Table 1 shows an overall comparison between hierarchical IBE (HIBE) [7], wildcarded IBE (WIBE) [11], wildcarded key derivation IBE (WKD-IBE) [13], WW-IBE [14], ciphertext policy ABE (CP-ABE) [3], and our proposed scalable WIBE (SWIBE) scheme. The comparison focuses on the size of public parameter (pp), secret key (SK), ciphertext (CT) and execution time of encryption (Enc), decryption (Dec), while the identity (pattern) depth is fixed as L and each identity is a q-bit string.

For the wildcard use, HIBE does not support wildcard, WIBE and CP-ABE supports wildcard only in encryption, WKD-IBE supports wildcard only in key derivation, while WW-IBE and SWIBE allow wildcard both in encryption and key derivation. For the ciphertext policy in CP-ABE, we assumed each bit in the ID as an attribute. Among the WIBE schemes which allow wildcards in encryption (WIBE, WW-IBE, SWIBE), only SWIBE has a constant-size ciphertext of 4 group elements regardless of the depth L. While supporting wildcarded pattern both in key derivation and encryption, SWIBE improves the decryption time compared to the WW-IBE.

Table 1. Comparison of parameters between related works. ref. e = scalar multiplication, p = pairing, and L = hierarchy depth, q = ID bits, size indicates group elements, Enc = Encryption, and Der = Key derivation.

	HIBE [7]	**WIBE** [11]	**WKD-IBE** [13]	**WW-IBE** [14]	**CP-ABE** [3]	**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	None	Enc	Der	Enc&Der	Enc	Enc&Der

Selective security. Despite the novel advantages, one remaining challenge is that the security of SWIBE relies on the selective security. In the selective security, the security model assumes that the target devices (i.e., devices which adversary desires to attack) are fixed before the security game. However, in real-life situation, we often cannot predict which devices are vulnerable, and which devices the attackers are planning to exploit. Therefore, for more robust security, it is recommended to advance the scheme to achieve full-security, i.e., IND-CPA secure without selective security.

Fully-secure SWIBE. In this paper, we extend selectively-secure SWIBE [15] to a fully-secure version, by modifying the SWIBE construction based on the composite order group. The composite order group sets the size of the cyclic group N as a composite number instead of a prime, to create different subgroups and let them cancel each other when computed in the pairing operation. This technique can disguise public elements while maintaining the correctness of the protocol, which can achieve the full security instead of the selective security. Based on the idea, we propose a new version of SWIBE protocol which satisfies full security, and formally prove the security under the standard assumptions.

We now summarize the main contributions of SWIBE, and SWIBE with full security.

- Scalable wildcarded identity-based encryption (SWIBE) achieves constant-size ciphertext among the WIBE systems, without sacrificing the order of other parameters.
- SWIBE is formally proved to satisfy IND-sID-CPA security, and it is extended to satisfy IND-sID-CCA security.
- We provide practical experimental results based on a small IoT device with 500 MHz Atom processor.
- This paper additionally proposes a fully secure (IND-ID-CPA-secure) SWIBE construction to overcome the selective security, and we formally prove the IND-ID-CPA security.

The rest of 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. Section 4 explains the main idea of the SWIBE 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 SWIBE 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

Identity-based encryption (IBE) [16] is defined as a tuple of algorithm $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ which provides the following functionality. The authority runs Setup to generate a public parameter pp and a master secret key msk. It publishes pp and keeps msk in private. When a user with an identity ID needs to join the system, the authority creates a decryption key

$d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$, and sends the key to the user via secure channel. To encrypt a message m to the user with an identity ID, the sender computes a ciphertext $C \xleftarrow{\$} \mathsf{Enc}(pp, ID, \mathsf{m})$, which is decrypted by the user as $\mathsf{m} \leftarrow \mathsf{Dec}(d_{ID}, C)$.

2.2. Hierarchical IBE

In hierarchical IBE (HIBE) [7], users are organized in an L-level binary tree, where the root is considered as a master authority. The user's identity at level $0 \leq l \leq L$ is given by a vector $ID = (P_1, \ldots, P_l) \in (\{0,1\}^q)^l$. HIBE is defined as a tuple of algorithms $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ providing same functionalities as in IBE, except that a user $ID = (P_1, \ldots, P_l)$ at level l can use his own secret key sk_{ID} to generate a secret key for its children $ID' = (P_1, \ldots, P_l, \ldots, P_L)$ via $sk_{ID'} \xleftarrow{\$} \mathsf{KeyDer}(sk_{ID}, ID')$. By applying the KeyDer algorithm interactively, user ID can derive secret keys for any of its descendants $ID' = (P_1, \ldots, P_{l+\delta})$, $\delta \geq 0$. We denote this process as $sk_{ID'} \xleftarrow{\$} \mathsf{KeyDer}(sk_{ID}, (P_{l+1}, \ldots, P_{L+1}))$. The secret key of the root at level 0 is $sk_{\epsilon} = msk$. Encryption and decryption are the same as IBE, but the identity is a vector of strings instead of ordinary strings.

2.3. Bilinear Groups and Pairings

In references [16,17], the bilinear maps and bilinear map groups are defined as follows:

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

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

- The map is bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$
- The map is non-degenerate: $e(g, g) \neq 1$.

$\mathbb{G}$ is a bilinear group if the operation in $\mathbb{G}$ is efficient and there is a group $\mathbb{G}_1$ with an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$.

2.4. Computational Complexity Assumptions

The security of our scheme is based on bilinear Diffie–Hellman Exponent (BDHE) assumption used in [7], which is an extension of bilinear Diffie–Hellman Inversion (BDHI) assumption previously used in [18]. Let $\mathbb{G}$ be a bilinear group of prime order p. The L-BDHE problem in $\mathbb{G}$ is defined as follows:

$$(h, g, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^L)}, g^{(\alpha^{L+2})}, \ldots, g^{(\alpha^{2L})} \in \mathbb{G}^{2L+1})$$

Given a vector of $2L + 1$ elements above as input, output $e(g, h)^{\alpha^{L+1}} \in \mathbb{G}_1$.

Once we specify g and α, we use y_i to represent $y_i = g^{\alpha^i} \in \mathbb{G}$. The adversary $\mathcal{A}$ has an 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 bits used by $\mathcal{A}$, the random choice of generators g, h in $\mathbb{G}$, and the random choice of α in $\mathbb{Z}_p$. The decisional version of the L-BDHE problem in $\mathbb{G}$ is defined analogously. Let $\vec{y}_{g,\alpha,L} = (y_1, \ldots, y_L, y_{L+2}, \ldots, y_{2L})$. An algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ϵ in solving decisional L-BDHE in $\mathbb{G}$ if

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

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

Definition 1. *The (decisional) (t, ϵ, L)-BDHE assumption holds in $\mathbb{G}$ if no t-time algorithm has at least ϵ advantage to solve the (decisional) L-BDHE problem in $\mathbb{G}$.*

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

3. Model

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 [11].

Definition 2. *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 SWIBE scheme. 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.*

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

Definition 5. *$W(P)$ denotes a set including all wildcard indices in a pattern P.*

Definition 6. *$\overline{W}(P)$ represents a 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 with the secret key for a pattern P which 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 $m \in \mathbb{Z}_p^*$, and all patterns $P, P' \in (\mathbb{Z}_p^* \cup \{*\})^L$ such that $P \approx P'$, $\mathsf{Decrypt}(\mathsf{KeyDer}\ (msk, P), \mathsf{Encrypt}(pp, P', m), P') = m$.

SECURITY. We define the security of SWIBE scheme similar to the original WIBE [11], except we also consider the key delegation property. We allow an adversary to select an arbitrary pattern and query secret keys for the corresponding pattern. The adversary is disallowed to ask a key derivation query for any pattern matching the challenge pattern. The security of SWIBE 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 inputs.

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 $P' \in (\mathbb{Z}_p^* \cup \{*\})^L$, 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 $m_0^*, m_1^* \in \mathbb{Z}_p^*$ 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 must 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 the proposed scalable wildcarded identity based encryption scheme (SWIBE). Before presenting the formal scheme, we first describe the BBG-HIBE construction [7], which is a main building block for WIBE systems including SWIBE, along with our main idea to achieve constant-size ciphertext in Section 4.1. Then we provide SWIBE construction in Section 4.2, and show the formal security proof in Section 4.3.

4.1. Overview: BBG-HIBE and Our Idea

The BBG-HIBE scheme [7] 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 [7] has the advantage of constant-sized ciphertexts. The ciphertext consists of only three elements: $(g^s, (g_3 \cdot h_1^{P_1} \cdots h_l^{P_l})^s, M(g, g_2)^{s\alpha})$. When observing the secret key, a_1 and a_2 are responsible for the decryption, while b_i stands for further key delegations. The pattern $P = (P_1, \cdots, P_l)$ is combined as a single element a_1, and we can delegate a_1 to another extended pattern $P = (P_1, \cdots, P_l, P_{l+1})$ by additionally combining the b_{l+1} into the a_1. Since the united element a_1 directly involves the pattern and the element can be canceled by multiplying its inverse, we observe that secret key for some pattern can be replaced to another secret key for a different pattern by multiplication. 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 $g_2^{\alpha}(g_3 \cdot h_1^{P_1} \cdots h_l^{P_l})^r$.

The wildcard pattern can work in a similar way: when we consider the wildcard $*$ as a fixed identity, i.e., a mapped element $w \in \mathbb{Z}_p^*$, we may change an identity to the wildcard by including $\mathbf{c_i} = (h_i^w / h_i^{P_i})^r$ for every identity string P_i in a secret key and replacing $(h_i^{P_i})^r$ with $(h_i^w)^r$. However, this approach has a security problem; the extra c_i allows the adversary to compute $h_i^r = \mathbf{c_i}^{1/(w-P_i)}$ by combining c_i, w, P_i, which reveals the top level secret key $g_2^{\alpha} g_3^r$ by canceling $(h_1^{P_1} \cdots h_l^{P_l})^r$ from $g_2^{\alpha}(g_3 h_1^{P_1} \cdots h_l^{P_l})^r$. With the top level secret key, the adversary can generate any secret keys as he desires, which is a significant problem.

We resolve this issue by randomizing the wildcard part, by using another random $t \in \mathbb{Z}_p$ independent from r. Specifically, we revise the extra key c_i as $\mathbf{c_i} = h_i^{wt} / h_i^{P_i r}$, and also provide g^t with g^r, to let the decryption work correctly. For instance, 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 remaining issue is that the encryption requires some adjustments, to be compatible with the modification of random t. For example, to encrypt for the pattern $(*, P_2, *)$, the pattern is divided into wildcard part and non-wildcard part. The non-wildcard identity part $(\cdot, P_2, \cdot)$ is encrypted as $(g^s, (g_3 \cdot h_2^{P_2})^s, M \cdot e(g_1, g_2)^s)$, same as the BBG-HIBE. Then the wildcard identity part $(*, \cdot, *)$ is encrypted as $(h_1^w h_3^w)^s$, which can later be used in decryption to cancel out the wildcard part of user secret key. In summary, the ciphertext can remain as constant with an additional single group element to the original BBG-HIBE ciphertext which is three group elements, and the key size also remains efficient, linear to the number of patterns, within the same order compared with the BBG-HIBE.

The complete scheme is represented in the following Section 4.2 with assuming the value $w = 1$. Then we formally prove the security of the scheme in Section 4.3.

4.2. SWIBE Construction

We now describe the SWIBE scheme which achieves constant size ciphertext and $O(L)$ size keys.

$\mathsf{Setup}(L)$: Let L indicate the maximum depth of the pattern. The initial set of keys are generated as follows. Sample $\alpha \in \mathbb{Z}_p^*$, and $O(L)$ group elements $g, g_2, g_3, h_1, h_2, \ldots, h_L \in \mathbb{G}$ randomly, 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 set as $msk = g_2^\alpha$.

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

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

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 $r', t' \xleftarrow{\$} \mathbb{Z}_q^*$ and output $sk_{P'} = (a'_1, a'_2, a'_3, b', c', d')$, where

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

$\mathsf{Encrypt}(pp,\ P,\ m)$: 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 $C = (C_1, C_2, C_3, C_4)$

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

$\mathsf{Decrypt}(sk_P,\ C,\ P')$: 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 correctness of the decryption is described 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 $d_i = \frac{h_i^t}{h_i^{P_i r}}$,

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

Since $\mathsf{W}_{P'\overline{P}} \cup \mathsf{W}_{\overline{P}'\overline{P}} = \overline{W}(P)$ and $\mathsf{W}_{P'\overline{P}} \cap \mathsf{W}_{\overline{P}'\overline{P}} = \emptyset$, $\prod_{i \in \overline{W}(P)} h_i^{P_i} \cdot \prod_{i \in \mathsf{W}_{P'\overline{P}}} h_i^{-P_i} = \prod_{i \in \mathsf{W}_{\overline{P}'\overline{P}}} h_i^{P_i}$. 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 the formal proof for IND-sID-CPA-security of 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 $\boldsymbol{e}$ and $\boldsymbol{p}$ denote the execution times of scalar multiplication and pairing in $\mathbb{G}$, respectively.*

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 chosen from P_{BDHE} where $T = e(g,h)^{(\alpha^{L+1})}$ or from R_{BDHE} where T is uniformly distributed in $\mathbb{G}_1$. The goal of algorithm $\mathcal{B}$ is to output 1 when the input is sampled from P_{BDHE} and 0 otherwise. Algorithm $\mathcal{B}$ interacts 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 the 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 that adversary $\mathcal{B}$ queries a key derivation query for pattern $P = (P_1, \ldots, P_L) \in_* (\mathbb{Z}_p^* \cup \{*\})^L$. The definition of the security experiment says that $P^* \notin_* P$. Therefore, 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

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

We have

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

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. We can check 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,

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

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 $i \in W(P)$,

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

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: For the 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 $h^{\sum_{i \in W(P^*)} \gamma_i}$. Then, it selects a random bit $b \in \{0, 1\}$ and sets $C_3 = m_b \cdot T \cdot e(y_1, h)^{\gamma}$. It returns $C = (C_1, C_2, C_3, C_4)$ as a challenge to $\mathcal{A}$. We claim that if $T = e(g, h)^{(\alpha^{L+1})}$ (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

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

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 randomly chosen 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 $\mathcal{B}$ terminates 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 denoting that T is random in $\mathbb{G}_1$.

When the input tuple is chosen from P_{BDHE} (where $T = e(g, h)^{(\alpha^{L+1})}$) then $\mathcal{A}$ satisfies $|Pr[b = b'] - 1/2| \geq \epsilon$. When the input tuple is selected 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

$$
|Pr[B(g, h, \vec{y}_{g,\alpha,L}, e(g, h)^{(\alpha^{L+1})}) = 0] - Pr[B(g, h, \vec{y}_{g,\alpha,L}, T) = 0]| \geq |(1/2 + \epsilon) - 1/2| = \epsilon
$$

□

5. Extension to CCA Security

In this section, we show the extension of CPA-secure SWIBE to satisfy CCA-security, by applying the similar technique from the extension in [19]. Given any one-time signature scheme $(SigKeygen, Sign, Verify)$ with a verification key as a q-bit string, we construct an L-level SWIBE Π = (Setup, KeyDer, Encrypt, Decrypt) secure against chosen-ciphertext attacks using the $(L+1)$-level Π' = (Setup$'$, KeyDer$'$ Encrypt$'$, Decrypt$'$) semantically secure SWIBE. The idea 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 for 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 encrypts m to $P'_C = (P_{C_1}, \cdots, P_{C_L}, V_{sig})$ using Π'.

The construction of L-level Π utilizing $(L+1)$-level Π' and a one-time signature scheme is 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 $(t'',\ \epsilon_2)$ 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. Assume there exists a t-time adversary, $\mathcal{A}$, such that $|AdvBr_{\mathcal{A},\Pi} - 1/2| > \epsilon_1 + \epsilon_2$. We build an algorithm $\mathcal{B}$, which 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' generates secret keys in a compressed way using wildcard $*$, P' can be categorized into the following two cases:

1. $P' = (P_1, \cdots, P_L, *)$ for $P \notin_* P^*$
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 utilized to respond to the decryption queries of $\mathcal{A}$ as described in the below.

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 following:

1. Execute $Verify$ to verify the signature σ on (C_1, C_2, C_3, C_4) using verification key V_{sig}. If the signature is invalid, then $\mathcal{B}$ responds with $\perp$.
2. If $V_{sig} = V^*_{sig}$, a *forge* event occurs, algorithm $\mathcal{B}$ outputs a random bit $b \overset{\$}{\leftarrow} \{0,1\}$ 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, the remaining process is to bound the probability that $\mathcal{B}$ aborts the simulation as a result of *forge*, which we claim as $Pr[forge] < \epsilon_2$. Otherwise, one can use $\mathcal{A}$ to forge signatures with a probability of at least ϵ_2. In brief, we can construct another simulator which knows the private key, but receives K^*_{sig} as a challenge in an existential forgery game.

In the experiment above, $\mathcal{A}$ aborts by submitting a query which includes an existential forgery under K^*_{sig} on some ciphertexts. Our simulator utilizes the forgery to win the existential forgery game. Note that a single chosen message query is asked by adversary $\mathcal{A}$ to generate a signature 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 [14], 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 [14]. We use groups where the order is a product of three primes and a generator G which takes security parameter λ as input and generates 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 the subgroup of order abc as $\mathbb{G}_{abc}$. Considering the fact that the group is cyclic, it is easy to verify that if g and h are group elements of different order (belonging to different subgroups), then $e(g, h) = 1$.

6.2. Complexity Assumptions

We first restate the complexity assumptions we use, which were originally introduced by Lewko and Waters in [20] and used in [14].

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 ($N = p_1p_2p_3$). 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 secret key $sk_{P'}$ according to 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

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

$\mathsf{Encrypt}(pp,\ P,\ m)$: 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 $C = (C_1, C_2, C_3, C_4)$

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

$\mathsf{Decrypt}(sk_P,\ C,\ P')$: 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 Section 6.1.

6.4. Security Proof

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

Before presenting hybrid games, we introduce two additional structures used in the proofs of security similar to [14,20]:

Semi-Functional (SF) Ciphertext. Let g_2 denote a generator of $\mathbb{G}_{p_2}$. An SF ciphertext is constructed as follows: 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 generate 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.

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

Note that SF ciphertexts can be decrypted by normal secret keys and SF secret keys can be used to decrypt normal ciphertext. But an SF ciphertext, for pattern P', cannot be decrypted by an SF secret key. 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.

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 [14]) 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 [14]) 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 games defined 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 $\alpha, \delta \overset{\$}{\leftarrow} \mathbb{Z}_N$ and $\eta_i \overset{\$}{\leftarrow} \mathbb{Z}_{N i\in[L]}$, 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 $\alpha, \delta \overset{\$}{\leftarrow} \mathbb{Z}_N$, $(\eta_i \overset{\$}{\leftarrow} \mathbb{Z}_N)_{i\in[L]}$, 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 $0 \leq l \leq L$. $\mathcal{B}$ 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 $\alpha, \delta \overset{\$}{\leftarrow} \mathbb{Z}_N$, $(\eta_i \overset{\$}{\leftarrow} \mathbb{Z}_N)_{i\in[L]}$, 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 $zk_1, zk_2, zk_3 \overset{\$}{\leftarrow} \mathbb{Z}_N$, and $(zk_{b,i}, zk_{c,i} \overset{\$}{\leftarrow} \mathbb{Z}_N)_{i \in W(P)}$, and $(zk_{d,i} \overset{\$}{\leftarrow} \mathbb{Z}_N)_{i \in \overline{W}(P)}$. Then choose random $r, t \overset{\$}{\leftarrow} \mathbb{Z}_N$, $W_1, W_2, W_3 \overset{\$}{\leftarrow} \mathbb{G}_{p_3}$, $\{W_{a,i}, W_{d,i} \overset{\$}{\leftarrow} \mathbb{G}_{p_3}\}_{i \in \overline{W}(P)}$, and $\{W_{b,i}, W_{c,i} \overset{\$}{\leftarrow} \mathbb{G}_{p_3}\}_{i \in W(P)}$. 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 $r', t' \overset{\$}{\leftarrow} \mathbb{Z}_N$, $W_1, W_2, W_3 \overset{\$}{\leftarrow} \mathbb{G}_{p_3}$, $\{W_{a,i}, W_{d,i} \overset{\$}{\leftarrow} \mathbb{G}_{p_3}\}_{i \in \overline{W}(P)}$, and $\{W_{b,i}, W_{c,i} \overset{\$}{\leftarrow} \mathbb{G}_{p_3}\}_{i \in W(P)}$. 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:

$$\begin{aligned}
C_1 &= X_1 X_2, & C_2 &= \prod_{i \in \overline{W}(P)} (X_1 X_2)^{\eta_i \cdot P_i} \\
C_3 &= m \cdot e(\gamma, (X_1 X_2)^{\delta}), & C_4 &= \prod_{i \in W(P)} (X_1 X_2)^{\eta_i}
\end{aligned}$$

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 modulo p_2, we have that zk set and zc set are independent and randomly distributed. If $\mathcal{B}$ checks that the k-th key is SF by using the above procedure (to create an SF ciphertext for k-th secret key pattern), then $zk_2 \cdot zc_2 + zk_3 \cdot zc_4 = zk_1$, 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 show the experimental results by implementing SWIBE on an Intel Edison IoT machine with 32-bit 500 Mhz Atom processor and ublinux 3.10.17. We also implemented WIBE [11], WKD-IBE [13], WW-IBE [14], and constant CP-ABE (CCP-ABE) [3] on the same environment, and provide a comparison on encryption and decryption times.

Figure 1 shows the encryption and decryption times of SWIBE, WIBE, and CCP-ABE when increasing the maximal pattern depth (L) from 5 to 20. Note that SWIBE has more advanced functionality since SWIBE can support wildcards in both encryption and key generation, while WIBE and CCP-ABE allow for wildcards only in encryption. In CCP-ABE, each ID bit is considered as an attribute, and we assumed each pattern as 32-bit. The decryption time in CCP-ABE is slower, since the decryption process of the CCP-ABE involves multiple pairing operations proportional to the number of attributes. In the WIBE, when the user receives the ciphertext, the user needs to delegate the ciphertext to match his own specific ID pattern, which costs point multiplications. However, in SWIBE, the user can simply adjust the secret key of his pattern to the ciphertext pattern, which costs only negligible point additions. Therefore, the decryption time in SWIBE remains as constant; SWIBE improves the decryption time by up to 3 times compared with the WIBE, and 650 times compared with the CCP-ABE.

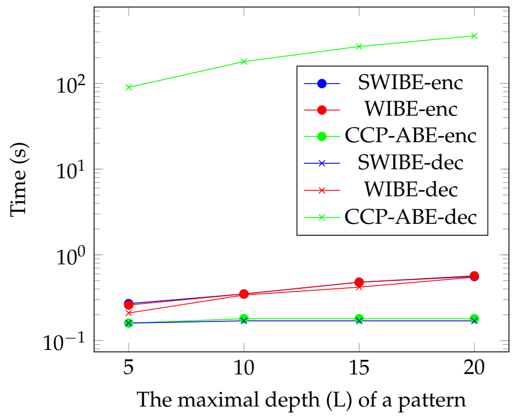

Figure 1. Encryption and decryption time in SWIBE, WIBE, and CCP-ABE.

Figure 2 shows the encryption times and decryption times of SWIBE and WKD-IBE, when varying the maximal pattern depth (L) from 5 to 20. Note that the WKD-IBE allows the wildcard only in the key derivation, while SWIBE can include wildcards in both encryption and key derivation. Despite the better functionality, the encryption time and decryption time of SWIBE is almost identical to the results in WKD-IBE.

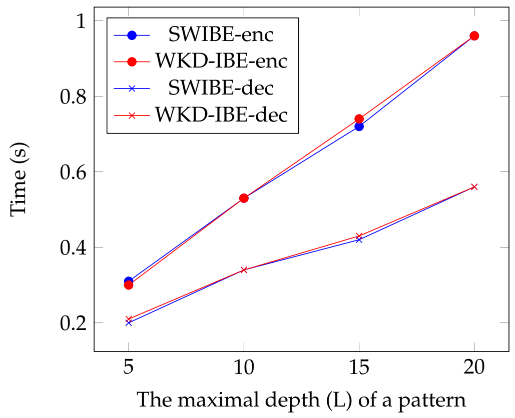

Figure 2. Encryption and decryption time in SWIBE and WKD-IBE.

Figure 3 shows the encryption times and decryption times of SWIBE and WW-IBE, when varying the maximal pattern depth (L) from 5 to 20. Both encryption time and decryption time in SWIBE is faster than that of the WW-IBE, since WW-IBE involves heavy composite order pairing groups for encryption and requires $2L$ pairing operations in decryption; SWIBE improves the decryption time by 10 times compared with the WW-IBE.

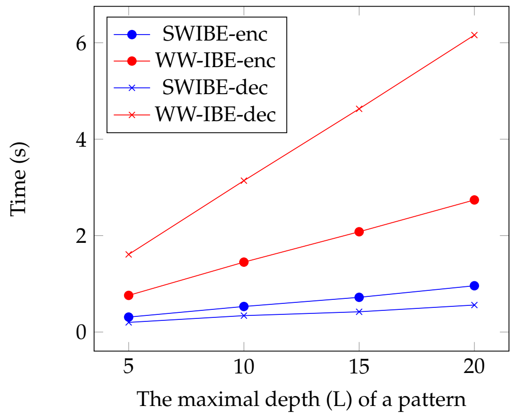

Figure 3. Encryption and decryption time in SWIBE and WW-IBE.

8. Conclusions

In this paper, we propose a scalable wildcarded identity-based encryption (SWIBE), which is a new wildcarded identity-based encryption (WIBE) which first achieves the constant-size ciphertext. SWIBE supports the wildcard to be included both in encryption and key derivation, where one can encrypt messages to a group of users by using wildcards in the pattern, and one can delegate the secret key from the wildcard pattern to another identity string. We propose a standard IND-sID-CPA-secure SWIBE, and extend it to the IND-sID-CCA-secure SWIBE, both with formal security proofs. Then we also propose the fully-secure version of SWIBE, which overcomes the selective security. Our experimental results show that SWIBE improves the decryption time by 3, 10, and 650 times compared with the WIBE, WW-IBE, and CCP-ABE.

Author Contributions: Conceptualization, J.K. and H.O.; methodology, J.K. and H.O.; software, J.L. and S.L.; validation, J.L. and S.L.; writing—original draft preparation, J.L. and S.L.; writing—review and editing, J.K., J.L., S.L., and H.O. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT Korea (2017-0-00661, 2016-6-00599).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the IEEE Computer Society, Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
2. Emura, K.; Miyaji, A.; Nomura, A.; Omote, K.; Soshi, M. A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length. In Proceedings of the Information Security Practice and Experience, 5th International Conference (ISPEC 2009), Xi'an, China, 13–15 April 2009; pp. 13–23.
3. 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 2010), Chicago, IL, USA, 4–8 October 2010; pp. 753–755.
4. Shamir, A. Identity-based cryptosystems and signature schemes. In Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Paris, France, 9–11 April 1984; pp. 47–53.
5. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; pp. 213–229.
6. Waters, B. Efficient identity-based encryption without random oracles. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; pp. 114–127.
7. Boneh, D.; Boyen, X.; Goh, E.J. Hierarchical identity based encryption with constant size ciphertext. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; pp. 440–456.

8. Boneh, D.; Gentry, C.; Waters, B. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2005; pp. 258–275.
9. Liu, W.; Liu, J.; Wu, Q.; Qin, B. Hierarchical Identity-Based Broadcast Encryption. In Proceedings of the Australasian Conference on Information Security and Privacy, Wollongong, Australia, 7–9 July 2014; Volume 14, pp. 242–257.
10. Boneh, D.; Hamburg, M. Generalized identity based and broadcast encryption schemes. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, 7–11 December 2008; pp. 455–470.
11. Abdalla, M.; Catalano, D.; Dent, A.W.; Malone-Lee, J.; Neven, G.; Smart, N.P. Identity-based encryption gone wild. In Proceedings of the International Colloquium on Automata, Languages, and Programming, Venice, Italy, 10–14 July 2006; pp. 300–311.
12. Birkett, J.; Dent, A.W.; Neven, G.; Schuldt, J.C.N. Efficient chosen-ciphertext secure identity-based encryption with wildcards. In Proceedings of the Australasian Conference on Information Security and Privacy, Townsville, Australia, 2–4 July 2007; pp. 274–292.
13. Abdalla, M.; Kiltz, E.; Neven, G. Generalized key delegation for hierarchical identity-based encryption. In Proceedings of the European Symposium on Research in Computer Security, Dresden, Germany, 24–26 September 2007; pp. 139–154.
14. Abdalla, M.; Caro, A.D.; Phan, D.H. Generalized Key Delegation for Wildcarded Identity-Based and Inner-Product Encryption. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 1695–1706. [CrossRef]
15. Kim, J.; Lee, S.; Lee, J.; Oh, H. Scalable wildcarded identity-based encryption. In Proceedings of the European Symposium on Research in Computer Security, Barcelona, Spain, 3–7 September 2018; pp. 269–487.
16. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. *SIAM J. Comput.* **2003**, *32*, 586–615. [CrossRef]
17. Joux, A. Multicollisions in iterated hash functions. Application to cascaded constructions. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2004; pp. 306–316.
18. Boneh, D.; Boyen, X. Efficient Selective Identity-Based Encryption without Random Oracles. *J. Cryptol.* **2011**, *24*, 659–693. [CrossRef]
19. Canetti, R.; Halevi, S.; Katz, J. Chosen-ciphertext security from identity-based encryption. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; pp. 207–222.
20. Lewko, A.; Waters, B. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Proceedings of the Theory of Cryptography Conference, Zurich, Switzerland, 9–11 February 2010; pp. 455–479.