Identity-Based Proxy Signcryption Protocol with Universal Composability

Huifang Yu 1, Zhicang Wang 2, Jianmin Li 3, and Xinzhe Gao 4

1 School of Communication and Information Engineering, Xi’an University of Posts & Telecommunications, Xi’an 710121, China
2 School of Automation, Xi’an University of Posts & Telecommunications, Xi’an 710121, China
3 Meteorological Observatory of Qinghai Province, Xining 810001, China
4 School of Computer, Qinghai Normal University, Xining 810008, China

Correspondence should be addressed to Huifang Yu, yuhuifang@qhnu.edu.cn

Received 25 August 2018; Accepted 14 November 2018; Published 2 December 2018

Copyright © 2018 Huifang Yu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Proxy signcryption means that the proxy signcrypter obtains the delegate authorization from the original signcrypter and then signcrypts the specified message on behalf of the original signcrypter. In this paper, we construct an identity-based proxy signcryption protocol (IBPSP) based on the universally composable (UC) framework. In the random oracle model, we prove that this protocol has the semantic security under the gap bilinear Diffie-Hellman and computational Diffie-Hellman assumptions. At the same time, an ideal functionality of the identity-based proxy signcryption protocol is defined in the UC security framework, and we also prove the equivalence between the universally composable identity-based proxy signcryption protocol and its IND-CCA2 and UF-CMA security. Analysis shows this IBPSP has both low computation complexity and semantic security together with UC security.

1. Introduction


The UC security framework [7] can meet the requirement of modular design of protocols. The salient property of UC definitions of security is that they guarantee security even when a secure protocol is composed of an arbitrary set of protocols, or when the protocol is used as a component of an arbitrary system. It is an essential property in a complex and unpredictable environment. Moreover, UC definitions can guarantee security even when an unbounded number of protocol instances are carried out concurrently in an adversarially controlled manner; furthermore, they can guarantee nonmalleability with respect to arbitrary protocols. In the UC framework, a protocol is abstracted as an ideal functionality which is considered as a trusted third party (TTP) for specific tasks. Every player can deliver a command to the ideal functionality in a secure and authentic manner, and the ideal functionality executes the command according to its specification. Informally, one protocol can securely carry out a given ideal functionality if no adversary can obtain more advantages from an attack on an instance of real execution of the protocol than from an attack on an ideal process where the parties only submit their inputs to
a TTP with appropriate functionality. In other words, it is required that an instance of real execution can be simulated in the ideal process. A protocol is said to securely realize ideal functionality if any effect caused by an adversary attacking the protocol can be achieved by an adversary attacking the ideal functionality. i.e., when devising the complex protocols, one can allow the involved parties to have secure access to ideal functionalities; when implementing the protocol, each ideal functionality is replaced by one protocol securely realizing functionalities; when devising the complex protocols, one protocol can be achieved by an adversary attacking the ideal functionality. A protocol is said to securely realize ideal functionalities; when devising the complex protocols, one protocol can be achieved by an adversary attacking the ideal functionality. Any ideal functionality can be UC realized as long as a majority of parties are supposed to be honest.


Identity-based proxy signcryption is appropriate for applications in electronic governments and electronic commerce [18], such as online proxy auctions or contract signing by an authorized proxy signer. Consider an application such as a bank account owned by a busy boss. So as to withdraw money from their saving account, the boss must sign a withdrawal slip which can only be verified by the bank teller. In this case, when the boss is unable to sign personally, they can delegate their signing power to a proxy signer who can legitimately conduct transactions on their behalf. It is an interesting research problem of how to construct a secure identity-based proxy signcryption protocol applying to a more complex network environment. In this paper, we construct a novel identity-based proxy signcryption protocol (IBPSP) with universal composability, whose semantic security is based on the intractability of the gap bilinear Diffie-Hellman (gap BDH) and computational Diffie-Hellman (CDH) problems. In the random oracle model, we prove its indistinguishability against adaptive chosen-ciphertext attacks (IND-CCA2) and existential unforgeability against adaptive chosen-message attacks (UF-CMA). At the same time, we define the ideal functionality of IBPSP and illustrate the equivalence between the universally composable IBPSP and its IND-CCA2 and UF-CMA security.

2. Preliminaries

In this section, we briefly introduce bilinear pairings and several computational problems together with the universally composable (UC) framework.

2.1. Bilinear Pairings. Let $G_1$ denote one cyclic addition group with prime order $q$ and let $G_2$ denote one cyclic multiplication group with the same prime order. In addition, $P$ is a generator of additive group $G_1$ and $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map with the properties as below:

- **Bilinearity**: $e(aU,bV) = e(U,V)^{ab}$ for any $U,V \in G_1$ and $a,b \in \mathbb{Z}_q^*$
- **Nondegeneracy**: there exists $U,V \in G_1$ such that $e(U,V) \neq 1_{G_2}$, where $1_{G_2}$ is the identity element of $G_2$
- **Computability**: there is an efficient algorithm to calculate $e(U,V)$ for any $U,V \in G_1$

2.1.1. BDH Problem. Given a $P_aP_bP_c >$ for any $a,b,c \in \mathbb{Z}_q^*$, the bilinear Diffie-Hellman (BDH) problem is that it is computationally infeasible to determine the value of $e(P,P)^{abc} \in G_2$.

2.1.2. CDH Problem. Given a $P_aP_bP_c >$ for any $a,b \in \mathbb{Z}_q^*$, the computational Diffie-Hellman (CDH) problem is that it is computationally infeasible to determine the value of $abP \in G_1$.

2.1.3. DBDH Problem. Given a $P_aP_bP_c >$ for any $a,b,c \in \mathbb{Z}_q^*$ and $Z \in G_2$, the decisional bilinear Diffie-Hellman (DBDH) problem is to decide whether $e(P,P)^{abc} \in Z$. If it holds, the DBDH oracle $\tilde{O}_{\text{DBDH}}$ returns 1; else, $\tilde{O}_{\text{DBDH}}$ returns 0.

2.1.4. Gap BDH Problem. Given a $P_aP_bP_c >$ for any $a,b,c \in \mathbb{Z}_q^*$, the gap bilinear Diffie-Hellman (gap BDH) problem is to determine the value of $e(P,P)^{abc} \in G_2$ with the help of $\tilde{O}_{\text{DBDH}}$.

2.2. UC Framework. The UC framework [8] consists of a real-life model, ideal model, and hybrid model. In the UC framework, $\mathcal{Z}$ represents the external environment that consists of arbitrary protocols running concurrently with the given protocol. $\mathcal{Z}$ generates the inputs to all parties, reads all outputs, and interacts with the adversary in an arbitrary way throughout the computation. A protocol $\pi$ is said to securely realize an ideal functionality $\mathcal{F}$ if, for any real-life adversary $\mathcal{A}$ interacting with the protocol, there exists a simulator $\hat{\mathcal{S}}$ such that no $\mathcal{Z}$ can distinguish with nonnegligible probability whether it interacts with $\mathcal{A}$ and parties running the protocol $\pi$ or with $\hat{\mathcal{S}}$ and parties interacting with $\mathcal{F}$ in the ideal process.

Let $\text{real}_{\pi,\mathcal{A},\mathcal{Z}}(k,z,r)$ denote the output of $\mathcal{Z}$ when interacting with the adversary $\mathcal{A}$ and parties $P_1, \ldots, P_n$ running protocol $\pi$ on security parameter $k$, input $z$, and random input $r = r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, \ldots, r_n$ ($z$ and $r_{\mathcal{Z}}$ for $\mathcal{Z}$, $r_{\mathcal{A}}$ for $\mathcal{A}$, and
$r_i$ for party $P_i$). Let $\text{real}_{\pi,d,\mathcal{X}}(k, z)$ be the random variable describing $\text{real}_{\pi,d,\mathcal{X}}(k, z, r)$ when $r$ is uniformly chosen. Here $\text{real}_{\pi,d,\mathcal{X}}$ is the ensemble

$$\{\text{real}_{\pi,d,\mathcal{X}}(k, z)\}_{k \in \mathbb{N}, z \in \{0,1\}}.$$  

Let $\text{ideal}_{\pi,d,\mathcal{X}}(k, z, r)$ denote the output of $\mathcal{X}$ after interacting with the adversary $\delta$ and ideal functionality $\mathcal{F}$ on security parameter $k$, input $z$, and random input $r = r_{\mathcal{F}}, r_{\delta,\mathcal{F}}(z)$ and $r_{\mathcal{X}}$ for $\mathcal{X}$, $r_{\delta}$ for $\delta$, and $r_{\mathcal{F}}$ for $\mathcal{F}$. Let $\text{ideal}_{\pi,d,\mathcal{X}}(k, z)$ be the random variable describing $\text{ideal}_{\pi,d,\mathcal{X}}(k, z, r)$ when $r$ is uniformly chosen. Here $\text{ideal}_{\pi,d,\mathcal{X}}$ is the ensemble

$$\{\text{ideal}_{\pi,d,\mathcal{X}}(k, z)\}_{k \in \mathbb{N}, z \in \{0,1\}}.$$  

Definition 1 (indistinguishability). Two binary distribution ensembles $X$ and $Y$ are indistinguishable (written $X \approx Y$) if, for any $c \in \mathbb{N}$, there exists $k_0 \in \mathbb{N}$ such that, for all $k > k_0$ and for all $a$, we have

$$|\Pr(X(k, a) = 1) - \Pr(Y(k, a) = 1)| < k^{-c}.$$ 

Definition 2 (UC simulation). Let $\mathcal{F}$ denote an ideal functionality and let $\pi$ denote a probabilistic polynomial time protocol. We say that $\pi$ securely realizes $\mathcal{F}$ if, for any adversary $\delta$, there exists an ideal-process adversary $\delta'$ such that, for any environment $\mathcal{I}$, we have

$$\text{ideal}_{\pi,d,\mathcal{X}} \approx \text{real}_{\pi,d,\mathcal{X}}.$$  

Theorem 3 (universal composition). Let $\mathcal{F}$ and $\mathcal{G}$ be ideal functionalities. Let $\pi$ be a protocol in the $\mathcal{F}$-hybrid model and let $\rho$ be a protocol which securely realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model. Then for any adversary $\delta$, there exists an adversary $\delta'$ in the $\mathcal{G}$-hybrid model such that, for any environment $\mathcal{I}$, we have

$$\text{ideal}_{\pi,d,\mathcal{X}} \approx \text{real}_{\pi',d,\mathcal{X}}.$$ 

3. Formal Definition of IBPSP

3.1. Algorithm Model. A universally composable IBPSP consists of five probabilistic polynomial time algorithms: Setup, Extract, PKeyGen, ProxyS, and Uns. Moreover, it involves four parties: a private key generator (PKG), an original signcyrptor with identity $I_a$, a proxy signcryption with identity $I_p$, and a receiver with identity $I_b$.

**Setup** is an initialization algorithm that takes a security parameter $k$ as input. The PKG runs this algorithm and outputs the master key $s$ along with a set of system parameters $\beta$.

**Extract** is an extraction algorithm that takes $\beta$ and an identity $I_i$ ($i$ is $a$, $p$, or $b$) of some user as input. The PKG runs this algorithm and outputs the public key $Y_i$ and private key $S_i$, of this user. Keep in mind that $I_a$ is the identity of the original signcyrptor, $I_p$ is the identity of the proxy signcryption, and $I_b$ is the identity of the receiver.

Apparently, the public and private key pair of the original signcyrptor, proxy signcyrptor, and receiver is $< Y_a, S_a >$, $< Y_p, S_p >$, and $< Y_b, S_b >$, respectively.

PKeyGen is a proxy key generation algorithm that takes $< \beta, I_a, I_p >$ as input. The original signcyrptor with identity $I_a$ generates an authorization certificate $m_w$ and then interacts with the proxy signcryption with identity $I_p$. Finally, this algorithm outputs a proxy signcryption key $S_{ap}$ for the proxy signcryption.

ProxyS is a proxy signcryption algorithm that takes $< \beta, \sigma, m_w, I_a, I_p, I_b, Y_p, Y_b, Y_a, S_{ap} >$ as input. The proxy signcryption with identity $I_a$ runs this algorithm and outputs a ciphertext $\sigma$ on message $m$ to the receiver with identity $I_b$.

Uns is an unsigncryption algorithm that takes $< \beta, \sigma, m_w, I_a, I_b, Y_p, Y_a, Y_b, S_{ap} >$ as input. The receiver with identity $I_b$ runs this unsigncryption algorithm and outputs a plaintext $m$ or a symbol $\perp$.

3.2. Security Model. In this section, we describe the formal security model of the universally composable IBPSP. A universally composable IBPSP must satisfy the confidentiality (IND-CCA2 security) and unforgeability (UF-CMA security). Keep in mind that we do not allow the queries when the identities of the original signcyrptor, proxy signcyrptor, or receiver are the same.

For the confidentiality of the universally composable IBPSP, we adopt the IND-CCA2 security model. Now, we describe the interactive game IND-IBPSP-CCA2 played between an adversary $\mathcal{A}$ and a challenger $C$.

**IND-IBPSP-CCA2:** First of all, $C$ runs the initialization algorithm to obtain a set of system parameters $\beta$ and the master key $s$. Then $C$ keeps the secrecy of $s$ but returns $\beta$ to $\mathcal{A}$.

**Phase 1.** $\mathcal{A}$ makes a sequence of a polynomial bounded number of queries in an adaptive fashion.

Extraction queries: $\mathcal{A}$ submits a query of identity $I$ to the extraction oracle. $C$ returns the private key $S_I$ and public key $Y_I$ by a call to the extraction algorithm.

Proxy key queries: $\mathcal{A}$ submits a query of the triplicate $< I_a, I_p, m_w >$ to the proxy key generation oracle. $C$ returns the proxy key $S_{ap}$ by a call to the proxy key generation algorithm.

Proxy signcryption queries: $\mathcal{A}$ submits a query of the quaternion $< I_a, I_b, I_p, m >$ to the proxy signcryption oracle. $C$ returns the ciphertext $\sigma$ on message $m$ by a call to the proxy signcryption algorithm.

Unsigncryption queries: $\mathcal{A}$ submits a query of the quaternion $< I_a, I_b, I_p, \sigma >$ to the unsigncryption oracle. $C$ returns a plaintext $m$ or a symbol $\perp$ by a call to the unsigncryption algorithm.

**Challenge.** At the end of **Phase 1**, the adversary outputs two messages $< m_{w1}, m_{w2} > \in \{0,1\}^\ell$ and quaternion $< I_a, I_b, I_p, m_w * >$ on which it wishes to be challenged. In **Phase 1**, the adversary cannot query the private key of identity $I_b^*$. $C$ selects a random $\delta \in \{0,1\}$ and obtains a challenge ciphertext $\sigma^*$ on message $m_w$ by a call to the proxy signcryption algorithm. Finally, $C$ returns $\sigma^*$ as the answer.

**Phase 2.** $\mathcal{A}$ makes another sequence of queries in an adaptive fashion as in the first phase. $C$ answers these queries as in the
first phase. However, the adversary cannot extract the private key of identity $I_o^*$ and cannot make the unsigncryption query on $\sigma^*$.

At the end of the game, the adversary outputs a guess $\delta' \in \{0, 1\}$. If $\delta = \delta'$, this shows that the adversary wins IND-IBPS-CCA2 above.

**Definition 4** (confidentiality). A universally composable IBPS is said to have the IND-CCA2 security if no probabilistic polynomial time adversary $\mathcal{A}$ wins IND-IBPS-CCA2 with nonnegligible advantage.

For the unforgeability of the universally composable IBPSP, we adopt the UF-CMA security model. Keep in mind that, in the security models, $\mathcal{A}_p$ is in possession of the private key of the proxy signcrypter and $\mathcal{A}_o$ owns the private key of the original signcrypter.

**UF-IBPS-CMA-I**: This is an interactive game performed between an adversary $\mathcal{A}_p$ and a challenger $C$.

First of all, $C$ generates a set of system parameters $\beta$ and the master key $s$ by a call to Setup($^1\beta^1$). It then keeps the secrecy of $s$ but delivers $\beta$ to the adversary.

**Queries.** $\mathcal{A}_p$ makes a series of a polynomial bounded number of queries in an adaptive manner as in the first phase in IND-IBPS-CCA2. Moreover, $C$ answers these queries as in the first phase in IND-IBPS-CCA2.

**Forgery.** At the end of queries, $\mathcal{A}_p$ outputs a forged ciphertext $<I_o^*, I_b^*, I_p^*, \sigma^*>$ to $C$. In queries, $\mathcal{A}_p$ cannot issue the private key query of identity $I_p^*$. If the unsigncryption verification equality holds, this indicates that $\mathcal{A}_p$ wins UF-IBPS-CMA-I.

**UF-IBPS-CMA-II**: This is an interactive game performed between an adversary $\mathcal{A}_o$ and a challenger $C$.

First of all, $C$ obtains the master key $s$ and a set of system parameters $\beta$ by a call to Setup($^1\beta^1$). It then returns $\beta$ to the adversary but keeps the secrecy of $s$.

**Queries.** $\mathcal{A}_o$ adaptively makes a series of a polynomial bounded number of queries as in Phase 1 in IND-IBPS-CCA2. $C$ answers these queries as in Phase 1 in IND-IBPS-CCA2.

**Forgery.** At the end of queries, $\mathcal{A}_o$ outputs a forged ciphertext $<I_o^*, I_b^*, I_p^*, \sigma^*>$ to $C$. In queries, $\mathcal{A}_o$ cannot extract the private key of identity $I_o^*$. If the verification equality holds, this shows that $\mathcal{A}_o$ wins UF-IBPS-CMA-II.

**Definition 5** (unforgeability). A universally composable IBPS is said to have UF-CMA-I security if no probabilistic polynomial time adversary $\mathcal{A}_p$ wins UF-IBPS-CMA-I with nonnegligible advantage. In a similar manner, a universally composable IBPSP is said to have the UF-CMA-II security if probabilistic polynomial time adversary $\mathcal{A}_o$ wins UF-IBPS-CMA-II with nonnegligible advantage. Hence, a universally composable IBPSP is UF-CMA secure if it is both UF-CMA-I and UF-CMA-II secure.

## 4. Concrete Protocol

In this section, we devise a concrete example of the universally composable IBPSP; its algorithm details are described as follows.

### 4.1. Setup

In this system initialization algorithm with the input $^1\beta$, the PKG obtains the master key $s$ and a set of system parameters $\beta$ by carrying out the steps as below.

Let $G_i$ denote a cyclic addition group with prime order $q$ and let $G_i$ denote a cyclic multiplication group with the same prime order. Moreover, $P$ is a generator of addition group $G_1$ and $e: G_1 \times G_1 \rightarrow G_2$ is a bilinear map. The PKG chooses four cryptography hash functions:

$$h_1 : \{0, 1\}_* \rightarrow G_1$$
$$h_2 : \{0, 1\}^t \times G_1 \rightarrow Z_q^*$$
$$h_3 : \{0, 1\}^t \times G_1 \times G_2 \rightarrow \{0, 1\}^t$$
$$h_4 : \{0, 1\}^t \times \{0, 1\}^t \times G_1 \times G_2 \rightarrow Z_q^*.$$ (6)

Then the PKG selects a random $s \in Z_q^*$ as the system master key and calculates the system public key $y = sP$. Finally, the PKG keeps the secrecy of master key $s$ but publishes a set of the system parameters

$$\beta = \langle G_1, G_2, q, e, t, \Pi, y, h_1 \sim h_4 \rangle.$$ (7)

### 4.2. Extract

In this extraction algorithm with the input $^1\beta$ and an identity $I_i$ ($i$ is $a, b, p$) of some user, the PKG carries out the following calculations to achieve this user’s public key and private key. Concretely speaking, the PKG calculates the public and private key for the original signcrypter with identity $I_a$ as follows:

$$Y_a = h_1(I_a),$$
$$S_a = sY_a.$$ (8)

In the same manner, the PKG calculates the public and private key for the proxy signcrypter with identity $I_p$ as follows:

$$Y_p = h_1(I_p),$$
$$S_p = sY_p.$$ (9)

We also readily obtain the public and private key for the receiver with identity $I_b$ as follows:

$$Y_b = h_1(I_b),$$
$$S_b = sY_b.$$ (10)

### 4.3. PKeyGen

In this proxy key generation algorithm with the input $<\beta, I_a, I_p>$, the original signcrypter with identity $I_a$ generates an authorization certificate $m_w$ which includes the identity information of the original signcrypter and proxy
signcrypter together with other restriction contents. The original signcrypter selects a random $k_a \in \mathbb{Z}_q^*$ and calculates

$$
U_a = k_a Y_a \\
V = h_2(m_w, U_a) S_a.
$$

After that, the original signcrypter delivers the trituple $< U_a, V, m_w >$ to the proxy signcrypter with identity $I_p$. Upon receiving $< U_a, V, m_w >$, the proxy signcrypter verifies whether the equality holds as below:

$$
e(P, V) = e(y, Y_a)^{h(m_w, U_a)}. \tag{11}
$$

If the verification is true, the proxy signcrypter calculates the proxy signcryption key

$$S_{ap} = S_p + V. \tag{13}
$$

4.4. ProxyS. In this proxy signcryption algorithm with the input $< \beta, m, m_w, I_1, I_p, I_b, Y_p, Y_b, S_{ap}>$, the proxy signcrypter with identity $I_p$ obtains a ciphertext $\sigma$ on message $m$ and delivers it to the receiver with identity $I_b$.

Concretely speaking, the proxy signcrypter selects a random $k_p \in \mathbb{Z}_q^*$ to set $U_p = k_p P$ and then continues to calculate

$$
R = e(y, Y_b)^{k_p} \\
c = m \oplus h_3(m_w, U_p, R) \\
\mu = h_4(m, I_a, I_p, I_b, U_p, R) \\
S = \mu S_{ap} + k_p Y_p.
$$

Finally, the ciphertext $\sigma = < c, m_w, S, U_a, U_p >$ is sent to the receiver.

4.5. Uns. In this unsigncryption algorithm with the input $< \beta, \sigma, m_w, I_a, I_b, I_p, Y_a, Y_p, Y_b, S_b>$, the receiver with identity $I_a$ recovers $m$ using two equalities as follows:

$$
R = e(U_p, S_b) \\
m = c \oplus h_3(m_w, U_p, R). \tag{15}
$$

After that, the receiver continues to calculate

$$\mu = h_4(m, I_a, I_p, I_b, U_p, R). \tag{16}
$$

Finally, this receiver checks whether the verification equality holds as follows:

$$
e(P, S) = e\left(y, h_2(m_w, U_a) Y_a + Y_p\right)^\mu e(U_p, Y_p). \tag{17}
$$

If the above verification is true, this illustrates that $\sigma$ is valid; otherwise, it is invalid.

It is easy for us to verify the correctness of our concrete protocol instance using three equalities as follows:

$$
e(P, V) = e(P, h_2(m_w, U_a) S_a) = e(P, sY_a)^{h(m_w, U_a)} = e(y, Y_a)^{h(m_w, U_a)} \\
R = e(y, Y_b)^{k_p} = e(U_p, sY_b) = e(U_p, S_b) \\
e(P, S) = e(P, \mu S_{ap} + k_p Y_p) = e(P, sY_p + h_2(m_w, U_a) S_a)^\mu e(U_p, Y_p) = e(y, h_2(m_w, U_a) Y_a + Y_p)^\mu e(U_p, Y_p). \tag{18}
$$

5. Analysis of Semantic Security

In this section, we describe the proof process of the confidentiality and unforgeability of our protocol instance in the random oracle model.

5.1. Confidentiality

**Theorem 6.** If no probabilistic polynomial time adversary $A$ wins IND-IBPS-CCA2 in Definition 4 with nonnegligible advantage $\varepsilon$ by asking $l_1$ queries to the $h_i$ ($i=1,2,3,4$) oracle, $l_e$ queries to the extraction oracle, and $l_{ap}$ queries to the proxy key generation oracle, then there exists an algorithm $C$ that can solve the gap BDH problem with the advantage $\varepsilon'$ ($\varepsilon$ is the base of natural logarithm), where

$$
\varepsilon' \geq \frac{\varepsilon}{e l_3(l_e + l_{ap})}. \tag{19}
$$

**Proof.** Assume the algorithm $C$ obtains a random instance $< P, aP, bP, cP >$ of the gap BDH problem, and the purpose of $C$ is to determine $R = (P, Y)^{abc} \in G_2$. For this purpose, $C$ runs the adversary $A$ as a subroutine and acts as the role of its challenger in the following interactive game.

At the start of the game, $C$ runs the system initialization algorithm with the security parameter $k$ and obtains a set of system parameters $\beta$ with $y = aP$. Then $C$ delivers $\beta$ to the adversary. In the whole game, $C$ maintains five lists $L_1 \sim L_4$ and $L_{ap}$, which are empty in the beginning; $L_1 \sim L_4$ are used to trace the $h_i$ ($i=1,2,3,4$) oracle and $L_{ap}$ is used to trace the extraction oracle and proxy key generation oracle.

**Phase 1.** $A$ submits a series of a polynomial bounded number of queries in an adaptive fashion.

$h_1$ queries: $C$ first selects an identity $I_1$ ($1 \leq \lambda \leq l_1$) as the target identity in the challenge phase. Bear in mind that $l_1$ is the query time to the $h_1$ oracle. Let $y$ be the probability of $I_1 = I_1$, and the value of $y$ will be determined later. $A$ submits a query of identity $I_1$ to the $h_1$ oracle. If there is a matching tuple in the list $L_1$, $C$ returns the public key $Y_1$ as the answer; otherwise, $C$ considers two cases as the response to this query.
Case 1. If it is the $\lambda$th query, $C$ sets $Y_i = h_1(I_i) = bP$. It then delivers public key $Y_i$ to the adversary and records $<I_i, Y_i, - , - >$ to the list $L_1$.

Case 2. If it is not the $\lambda$th query, $C$ uses a random $x_i \in Z^*_q$ of its choice to calculate $Y_i = h_1(I_i) = x_iP$. It then returns $Y_i$ as the answer and records $<I_i, Y_i, x_i, - >$ to the list $L_1$.

$h_1$ queries: $\mathcal{A}$ submits a query to the $h_1$ oracle. If there is a matching tuple in the list $L_2$, $C$ returns $f$ as the answer; otherwise, $C$ returns a random $f \in Z^*_q$ of its choice and records $<m_{wp}, U_{ap}, f, >$ to the list $L_2$.

$h_2$ queries: $\mathcal{A}$ submits a query to the $h_2$ oracle. If there exists a matching tuple in the list $L_3$, $C$ returns $\phi$ as the answer; otherwise, $C$ returns a random $\phi \in \{0, 1\}^l$ of its choice and records $<m_{wp}, R, U_p, \phi, >$ to the list $L_3$.

$h_4$ queries: $\mathcal{A}$ submits a query to the $h_4$ oracle. If there is a matching tuple in the list $L_4$, $C$ delivers $\mu$ to the adversary; otherwise, $C$ returns a random $\mu \in Z^*_q$ of its choice and stores $<m, I_a, I_b, I_p, U_p, R, \mu, >$ to the list $L_4$.

Extraction queries: $\mathcal{A}$ submits a query of identity $I_1$ to the extraction oracle. $C$ terminates this simulation if $I_1 = I_i$; otherwise, $C$ calculates $S_1 = x_i P = x_i y$ and then returns $<Y_1, S_1, >$ and stores $<I_1, Y_1, x_1, S_1, >$ into the list $L_1$.

Proxy key queries: $\mathcal{A}$ submits a proxy key query of the trituple $<I_a, I_p, m_{wp}, >$. Let us assume the adversary has made the $h_1$ and $h_2$ oracle queries along with key extraction queries before the proxy key query. If $I_a = I_{1a}$, $C$ terminates this simulation; otherwise, $C$ makes use of a random $k_a \in Z^*_q$ of its choice to calculate

$$U_a = k_a Y_a, \quad V = f x_a y. \quad (20)$$

If the verification equality $e(P, V) = e(y, Y_a)^f$ holds, $C$ calculates the proxy key $S_{ap}$ using the equality as below

$$S_{ap} = x_p y + V \quad (21)$$

and delivers $S_{ap}$ to the adversary. Eventually, $C$ stores $<m_{wp}, S_{ap}, U_{ap}, V, >$ into the list $L_{ap}$.

Proxy signcryption queries: Assume the adversary has made various hash oracle queries together with the extraction and proxy key generation query before a proxy signcryption query. $\mathcal{A}$ requests a query of the quaternion $<I_a, I_p, m_{wp}, \sigma, >$. $C$ considers two cases as the response to this proxy signcryption query.

Case 1. $I_a \neq I_{1a}$, $C$ returns a result by a call to the actual signcryption algorithm.

Case 2. $I_a = I_{1a}$, $C$ goes over the list $L_3$ to seek the tuple $<m_{wp}, U_{ap}, R, \phi, >$ for different $R$ such that $\theta_{DBDH}$ returns 1 when the adversary made a query on $<y, Y_p, U_p, R, >$. If there exists this case, $C$ calculates

$$m = c \oplus \phi \quad (25)$$

Finally, $C$ checks whether the verification equality holds as below:

$$e(P, S) = e(y, h_2 (m_{wp}, U_{ap}) Y_a + Y_p)^\mu (U_p, Y_p) . \quad (26)$$

If the verification is true, $C$ returns $m$; otherwise, $C$ returns $\perp$.

Challenge. As the adversary decides to end the first phase, it outputs $m_0$ and $m_1$ with the same length along with the quaternion $<I_a^*, I_b^*, I_p^*, m_{wp}^*, >$ on which it wishes to be challenged. Assume the adversary has queried various hash oracles together with the extraction and proxy key generation oracle before a challenge query. In the first phase, the private key of identity $I_b^*$ cannot be extracted. Upon receiving a challenge query, $C$ deals with this query by considering two cases as follows.

Case 1. If $I_p = I_{1a}$, $C$ terminates this simulation.
Case 2. If \( I_p^* \neq I_\lambda \), C selects a random \( \delta \in \{0,1\} \) and \( R^* \in G_2 \). Then C sets \( U_p^* = cP \) and calculates

\[
c^* = m_\delta \oplus \phi^* \\
\mu^* = h_4 \left( m_\delta, l_a^*, I_p^*, l_b^*, U_p^*, R^* \right).
\]  

(27)

C stores \( < m_w^*, U_p^*, R^*, \phi^* > \) into the list \( L_3 \) and records \( < m_\delta, l_a^*, I_p^*, l_b^*, U_p^*, R^*, \mu^* > \) to the list \( L_4 \). Finally, C continues to calculate

\[
S^* = \mu^* S_{ap}^* + x_p^* U_p^* 
\]

(28)

and returns \( \sigma^* = < c^*, m_w^*, S^*, U_a^*, U_p^* > \) as challenge ciphertext.

Phase 2. \( \mathcal{A} \) issues another sequence of oracle queries in an adaptive fashion as the first phase. In addition, \( C \) answers these oracle queries as in the first phase. However, \( \mathcal{A} \) cannot extract the private key of identity \( I_p^* \) and cannot make use of \( \sigma^* \) to query the unsigncryption oracle.

At the end of the game, the adversary outputs a guess \( \delta' \in \{0,1\} \) of \( \delta \). If \( \delta = \delta' \), C outputs the gap BDH problem instance:

\[
R^* = e \left( U_p^*, S_b^* \right) = e \left( cP, aY_b^* \right) = e \left( P, P \right)^{abc}
\]

(29)

Analysis of probability. In the following, we analyze the success probability of \( C \) in solving the gap BDH problem.

According to the detailed description in the interactive game, we have that the probability of \( C \) not terminating the simulation in the first phase or second phase is \( \gamma^{l_{ap}} \). And the probability of \( C \) not terminating the simulation in the challenge phase is \( 1-\gamma \). Thus, the probability of \( C \) not terminating the simulation is \( \gamma^{l_{ap}}(1-\gamma) \) whose value is maximized at

\[
\gamma = 1 - \frac{1}{1 + l_c + l_{ap}}.
\]

(30)

Referring to the method of the probability analysis in [20], we obtain that the probability of \( C \) not terminating the simulation is at least

\[
\frac{1}{e\left(l_c + l_{ap}\right)}
\]

(31)

At the same time, the probability of the adversary querying the \( h_1 \) oracle is \( 1/l_1 \). Hence, the probability of \( C \) in solving the gap BDH problem is at least

\[
\frac{\varepsilon}{e\left(l_c + l_{ap}\right)}
\]

(32)

If the adversary succeeds with probability \( \varepsilon \), \( C \) can solve the gap BDH problem with probability \( \varepsilon' \). It contracts with the initial gap BDH assumption. Therefore, \( \varepsilon \) must be negligible if the gap BDH problem is hard.

5.2. Unforgeability

Theorem 7. If no probabilistic polynomial time adversary \( \mathcal{A}_p \) (or \( \mathcal{A}_a \)) wins UF-IBPS-CMA-I (or UF-IBPS-CMA-II) with nonnegligible advantage \( \varepsilon \) by making \( l_i \) queries to the \( h_i \) (\( i=1,2,3,4 \)) oracle, \( l_x \) queries to the extraction oracle, \( l_w \) queries to the proxy key generation oracle, and \( l_{usc} \) queries to the unsigncryption oracle, then there exist two cases. (1) In UF-IBPS-CMA-I, there exists an algorithm \( C \) which can solve the CDH problem with the advantage \( \varepsilon_1 \), where

\[
\varepsilon_1 \geq \varepsilon \frac{e}{e_{usc}\left(l_c + l_{ap}\right)}
\]

(33)

(2) In UF-IBPS-CMA-II, there is an algorithm \( C \) which can solve the CDH problem with the advantage \( \varepsilon_2 \), where

\[
\varepsilon_2 \geq \varepsilon \frac{e}{e_{usc}}
\]

(34)

Proof. Give a random instance \( < P, aP, bP > \) of the CDH problem, and the aim of \( C \) is to determine the value of \( abP \in G_1 \). In order to achieve this aim, the algorithm \( C \) runs the adversary \( \mathcal{A}_p \) or \( \mathcal{A}_a \) as a subroutine and acts as the role of the challenger of \( \mathcal{A}_p \) or \( \mathcal{A}_a \) in the interactive game.

Initial. \( C \) obtains a set of system parameters \( \beta \) with \( y = aP \) by a call to the initialization algorithm and delivers \( \beta \) to \( \mathcal{A}_p \) or \( \mathcal{A}_a \). In the interactive game, \( C \) needs to maintain five lists \( L_1 \sim L_4 \) and \( l_{ap} \), which are empty in the beginning, and these lists make use of tracing the relevant random oracles.

Queries. These are the same as those in Phase 1 in Theorem 6.

Forgery. At the end of these queries, the response to the forgery is described as follows.

(1) As the queries are over, \( \mathcal{A}_p \) outputs a forged ciphertext \( \sigma^* = < c^*, m_w^*, S^*, U_a^*, U_p^* > \) to \( C \). In queries, \( \mathcal{A}_p \) cannot extract the private key of identity \( I_p^* \) and \( \sigma^* \) should not be the response of any proxy signcryption oracle query made by \( \mathcal{A}_p \). \( C \) considers two cases as the response to this query

Case 1. If \( I_p^* = I_\lambda \) and \( I_a^* = I_\lambda \), \( C \) terminates this simulation.

Case 2. If \( I_p^* \neq I_\lambda \) and \( I_a^* = I_\lambda \), \( C \) queries various hash oracles to obtain \( < x_p^*, f^*, \mu^* > \) and \( Y_a^* = bP \) from the list \( L_1 \). Then \( C \) outputs the solution of the CDH problem instance

\[
abP = \frac{S^* - \mu^* S_{ap}^* - x_p^* U_p^*}{\mu^* f^*}
\]

(35)

If \( C \) wins in the interactive game above, this shows that the verification equality must hold:

\[
\begin{align*}
e(P, S^*) & = e \left( y, h_5 \left( m_w^*, U_a^* \right) Y_a^* + Y_p^* \right)^{\mu} e \left( U_p^*, Y_p^* \right) \\
& = e \left( y, \mu^* f^* Y_a^* \right) e \left( y, \mu^* Y_p^* \right) e \left( P, x_p^* U_p^* \right) \\
& = e \left( P, \mu^* f^* abP \right) e \left( P, \mu^* S_{ap}^* \right) e \left( P, x_p^* U_p^* \right).
\end{align*}
\]

(36)
Analysis of probability. Now, we analyze the success probability that $C$ obtains the solution of the CDH problem instance.

From the interactive game, we know that the probability of $C$ not aborting the game in the first phase or second phase is $y^{k+1}$ and the probability of $C$ not aborting the execution of game in the challenge phase is $1-\gamma$. Thus, the probability of $C$ not aborting the execution of game is $y^{k+1}(1-\gamma)$. Referring to the probability analysis in Theorem 6, we can obtain that the probability of $C$ not aborting the execution of game is at least

$$\frac{1}{e(l_c + l_{ab})}. \tag{37}$$

Furthermore, the probability that $\sigma^*$ passes the unsigncryption verification is $l/l_{usc}$. Therefore, the probability of $C$ in solving the CDH problem is at least

$$\frac{e \cdot l_{usc}}{l(l_c + l_{ab})}. \tag{38}$$

(2) At the end of the queries, $\mathcal{A}_a$ outputs a forged ciphertext $\sigma^* = <e^*, m_w^*, S^*, U_a^*, U_p^*, l >$ to $C$, where $U_p^* = y$. In queries, $\mathcal{A}_a$ cannot issue the private key query of identity $I_a^*$ and $\sigma^*$ should not be the response of any proxy signcryption oracle query made by $\mathcal{A}_a$. It is needful for $C$ to deal with this query by considering two cases as below

Case 1. If $l_a^* = I_\lambda$ and $l_p^* = I_\lambda$, $C$ terminates this simulation.

Case 2. If $l_a^* \neq I_\lambda$ and $l_p^* = I_\lambda$, $C$ calls various hash oracles to achieve $< x_p^*, f^*, \mu^* >$ and $Y_p^* = bP$ from the list $L_1$. Then $C$ outputs the solution of the CDH problem instance:

$$abP = \frac{S^* - \mu^* f^* x_p^* y}{\mu^* + 1}. \tag{39}$$

If $C$ wins in the interactive game above, this indicates that the verification equality must hold:

$$e(P, S^*)
\begin{align*}
&= e\left(\left(y, h_2(m_w^*, U_a^*)Y_a^* + Y_p^*\right), e(U_p^*, Y_p^*)\right) \\
&= e\left(\left(y, \mu^* f^* Y_a^*\right), e\left(y, bP\right)\right) \\
&= e(P, \mu^* f^* x_a^* y) e(P, \mu^* abP) e(P, abP). \tag{40}
\end{align*}$$

Analysis of probability. Now, let us analyze the success probability of $C$ in solving the CDH problem.

From the interactive game above, we obtain that the probability of $C$ not aborting the game in the first phase or second phase is $y^{k+1}$ and the probability of $C$ not aborting the game in the challenge phase is $1-\gamma$. Thus, the probability of $C$ not aborting the game is $y^{k+1}(1-\gamma)$. Referring to the probability analysis in Theorem 6, we have that the probability of $C$ not aborting the game is at least $1/e(l_c)$.

In addition, the probability that $\sigma^*$ passes the unsigncryption verification is $l/l_{usc}$. Therefore, the probability of $C$ in solving the CDH problem is at least

$$\frac{e}{e(l_{usc})}. \tag{41}$$

If $\mathcal{A}_r$ or $\mathcal{A}_o$ succeeds with probability $e$, $C$ can solve the CDH problem with probability $e_1$ or $e_2$. It contracts with the initial CDH assumption. Hence, $e$ must be negligible if the CDH problem is hard.

6. Analysis of UC Security

6.1 Protocol $\pi_{IBPSP}$. A concrete $\pi_{IBPSP}$ usually makes use of the PKG to control two algorithms: Setup and Extract. $\pi_{IBPSP}$ in the UC framework proceeds as below, with parties $P_1, ..., P_n$ and an adversary $\mathcal{A}$.

(1) Upon input $<\text{PKG}, \text{Setup}, \text{sid}>$ from $\mathcal{A}$, the PKG verifies whether $\text{sid} = <\text{PKG}, \text{sid}^\prime>$. If not, the PKG ignores this request; else, the PKG obtains $<s, \beta>$ by a call to $\text{Setup}(t)$ and returns $\beta$.

(2) Upon input $<\text{Extract}, \text{sid} >$ from $\mathcal{A}$, $P_i$ obtains $<Y_i, S_i>$ by a call to $\text{Extract}(\beta, s, I_i)$ and returns $<Y_i, S_i>$.

(3) Upon input $<\text{PKeyGen}, \text{sid} >$ from $\mathcal{A}$, $P_i$ obtains $S_{ap}$ by a call to $\text{PKeyGen}(\beta, I_i, P_i)$ and returns $S_{ap}$.

(4) Upon input $<\text{ProxyS}, \text{sid}, m>$ from $\mathcal{A}$, $P_i$ obtains a ciphertext $\sigma$ by a call to the algorithm $\text{ProxyS}(\beta, m, m_w, I_a, I_b, I_p, Y_p, Y_p, S_{ap})$ and returns $\sigma$.

(5) Upon input $<\text{UnsC}, \text{sid}, \sigma >$ from $\mathcal{A}$, $P_i$ obtains $m \leftarrow \text{Uns}(\beta, \sigma, m_w, I_a, I_b, I_p, Y_p, Y_p, S_{ap})$ and then returns $v(\beta, \sigma, m, \text{sid})$.

6.2 Ideal Functionality $\mathcal{F}_{IBPSP}$. In this section, we introduce the definition of an ideal functionality $\mathcal{F}_{IBPSP}$ for the IBPSP, based on the security definitions in Section 2.2, $\mathcal{F}_{IBPSP}$ and $\pi_{IBPSP}$ have the exact same interface; that is to say, $\mathcal{F}_{IBPSP}$ can be realized by $\pi_{IBPSP}$ under certain condition. $\mathcal{F}_{IBPSP}$ proceeds as follows, with parties $P_1, ..., P_n$ and an ideal adversary $\mathcal{I}$. $\mathcal{I}$ is a deterministic verification algorithm.

(1) Given $<\text{Setup}, \text{sid} >$ from the PKG, verify whether $\text{sid} = <\text{PKG}, \text{sid}^\prime >$ for some $\text{sid}^\prime$. If not, then ignore this request; else, forward $<\text{Setup}, \text{sid} >$ to $\mathcal{I}$. Then, on receiving $<\text{Setup}, \text{sid}, v >$ from $\mathcal{I}$, record $<\text{sid}, v >$ and output $<\text{sid}, v >$ to $\mathcal{D}$.

(2) On receiving $<\text{Extract}, \text{sid} >$ from some party $P_i$, verify whether $\text{sid} = <P_i, \text{sid}^\prime >$. If not, then ignore this request; else, forward $<\text{Extract}, \text{sid} >$ to $\mathcal{D}$. Then, on receiving $<Y_i, S_i, v >$ from $\mathcal{D}$, record $<Y_i, S_i, v >$ and output $<Y_i, S_i >$ to $P_i$.

(3) On receiving $<\text{PKeyGen}, \text{sid} >$ from $P_i$, verify whether $\text{sid} = <P_i, \text{sid}^\prime >$. If not, then ignore this request; else, forward $<\text{PKeyGen}, \text{sid} >$ to $\mathcal{D}$. Then, on receiving $<S_{ap}, v >$ from $\mathcal{D}$, record $<S_{ap}, v >$ and output $S_{ap}$ to $P_i$.

(4) On receiving $<\text{ProxyS}, \text{sid}, \text{sid}^\prime, m >$ for some $m$ from $P_i$, verify whether $\text{sid} = <P_i, \text{sid}^\prime >$. If not, then ignore this request. Else, do the following:

(i) If $P_i$ is not corrupt and $v = \text{sid}^\prime$, deliver $<\text{ProxyS}, \text{sid}, |m| >$ to $\mathcal{D}$, where $|m|$ is the length of message
m; else, deliver < ProxyS, sid, v' , m > to δ ⊗ On receiving the replay σ from δ, deliver < ProxyS, sid, v', m, σ > to the party P_b and record < m, σ >

(5) On input < Uns, sid, σ > from P_b, verify whether sid =< P_b, sid' >. If not, then ignore this. Else, do the following:

○ If < m, σ > has been recorded, then achieve f ← υ(β, σ, m, sid) and send < m, f = 1 > to P_b. Else, return < Uns, sid, σ > to δ and obtain m from δ, and then send < m, f = 0 > to P_b.

6.3. Proof of UC Security

Theorem 8. If π_IBPSP satisfies the IND-CCA2 security, then π_IBPSP can securely realize F_IBPSP in the UC framework.

Proof. Assume the protocol π_IBPSP satisfies the IND-CCA2 security; then π_IBPSP can securely realize F_IBPSP in the UC framework. i.e., if no probabilistic polynomial time θ wins IND-IBPSP-CCA2 with nonnegligible advantage, then any environment E cannot distinguish whether it is interacting with π_IBPSP and θ in the real-life model or with F_IBPSP and θ in the ideal model.

Environment E proceeds as follows.
(1) E delivers a request on Setup. On receiving < Setup, sid >, E inputs β and activates θ.
(2) On receiving a request on < Extract, sid > from θ, E activates some party P_i with input < Extract, sid > and outputs < Y_i, S_i > to θ.
(3) On receiving < PKeyGen, sid > from θ, E activates P_a and P_b with < PKeyGen, sid > and outputs S_ap to θ.
(4) On receiving < ProxyS, sid, m > from θ, E activates P_p with input < ProxyS, sid, m > and outputs σ to θ.
(5) On receiving a request on < Uns, sid, σ > from θ, E activates P_p with input < Uns, sid, σ > and outputs the value f of υ(σ, m, sid).

If θ wins IND-IBPSP-CCA2, E outputs f = 1. In addition, the probability that E outputs 1 is 0 in the ideal model. Obviously, if π_IBPSP satisfies the IND-CCA2 security, π_IBPSP can securely realize F_IBPSP in the UC framework.

Theorem 9. If π_IBPSP can securely realize F_IBPSP in the UC framework, then π_IBPSP satisfies the IND-CCA2 security.

Proof. Here we construct an ideal adversary δ such that no environment E can tell with nonnegligible advantage whether it interacts with F_IBPSP and δ in the ideal model or with π_IBPSP and δ in the real-life model.

δ runs a copy of θ and sends the input from E to θ, and simultaneously any output from θ is considered as the output of δ.

(1) On receiving < Setup, sid > from F_IBPSP, it runs the Setup algorithm and obtains < β, s >, and then returns < Setup, sid, β >.
(2) On receiving < Extract, sid > from F_IBPSP, it invokes the extraction algorithm to achieve < Y_i, S_i > and then returns < Y_i, S_i >.

(3) On receiving < PKeyGen, sid > from F_IBPSP, it invokes the proxy key generation algorithm to obtain S_ap and then returns S_ap.
(4) On receiving < ProxyS, sid, m > from F_IBPSP, it runs the proxy signcryption algorithm to obtain σ and then returns σ.
(5) On receiving < Uns, sid, σ > from F_IBPSP, it runs the unsigncryption algorithm to obtain m and then returns m. Then it returns f ← υ(σ, m, sid). If P_b is corrupt, then deliver < m, f = 0 > to all parties; else, deliver < m, f = 1 > to P_b.

In the following, we prove that the ideal adversary δ simulates the real-life adversary θ.

(1) On receiving β from the challenger C, Z is invoked. On receiving < Setup, sid > from E, it issues a request to C that returns η = (sid, β) as the answer.
(2) On receiving < Extract, sid > from E, it issues a request to C that returns < Y_i, S_i > as the answer.
(3) On receiving < PKeyGen, sid > from E, it issues a request to C that returns S_ap as the answer.
(4) On receiving < ProxyS, sid, m > from E, it issues a request to C that responds as follows:
○ If P_p is not corrupt, send < ProxyS, sid, |m| > to C that returns σ as the answer.
○ If P_p is corrupt, deliver < ProxyS, sid, m > to C that returns σ as the answer.
(5) On input < Uns, sid, σ > from E, it issues a request to C that responds as follows:
○ If both P_p and P_b are not corrupt, then send < Uns, sid, σ > to C that returns m' as the answer.
○ If P_p is corrupt and P_b is not corrupt, then send < Uns, sid, σ > to C that returns m' as the answer.
○ If both P_p and P_b are corrupt, then deliver < Uns, sid, σ > to C that returns m as the answer.
Then, E obtains f ← υ(β, σ, m, sid) and returns f. If f = 1, this shows that the ciphertext is valid; else, it is invalid.

According to the running process, we define four events: for E, the occurrence of each event means that it cannot tell whether it interacts with π_IBPSP and θ in the real-life model or with F_IBPSP and θ in the ideal model.

Event 1: P_p and P_b are both corrupt. In the real-life process, P_p obtains σ of m by the proxy signcryption algorithm and sends σ to P_b. And P_b can unsigncrypted σ to obtain m and verifies the validity of σ. In the ideal process, δ simulates P_b to obtain σ' of m' of its choice and sends σ' to P_b. δ answers with m' when P_b unsigncrypts σ'; the unsigncryption succeeds if f = 1.

Event 2: P_p is not corrupt and P_b is corrupt. In the real-life process, E cannot determine which message is signcrypted by P_p. In the ideal process, δ simulates P_b with m' of its choice to obtain σ' and forwards σ' to θ. It follows that E cannot distinguish < m, σ > and < m', σ' >.

Event 3: P_p is corrupt and P_b is not corrupt. If P_p is corrupt, this shows θ obtains the process of proxy signcryption. On the other hand, δ can invoke a copy of θ; therefore, δ
can obtain the same information with \( \mathcal{A} \). That is to say, when \( \mathcal{A} \) makes the proxy signcryption, \( \mathcal{A} \) can perfectly simulate \( P_p \) that is corrupt.

Event 4: \( P_p \) and \( P_b \) are both corrupt. In this case, \( \mathcal{A} \) can obtain all their input information; i.e., \( \mathcal{A} \) can generate real data to simulate the execution of protocol.

In the four events defined above, if \( \mathcal{Z} \) can tell whether it interacts with \( \pi_{\text{IBPSP}} \) or with \( \mathcal{F}_{\text{IBPSP}} \), \( \mathcal{A} \) with probability \( \Pr(\mathcal{Z}^{(\pi_{\text{IBPSP}})}) \rightarrow 1 - \Pr(\mathcal{Z}^{(\mathcal{F}_{\text{IBPSP}})}) \rightarrow 1 \), then \( \mathcal{A} \) wins IND-IBPSP-CCA2 with probability \( \Pr(\mathcal{Z}^{(\pi_{\text{IBPSP}})}) \rightarrow 1 - \Pr(\mathcal{Z}^{(\mathcal{F}_{\text{IBPSP}})}) \rightarrow 1 \). In the real world, \( \Pr(\mathcal{Z}^{(\mathcal{F}_{\text{IBPSP}})}) \rightarrow 1 \) is a negligible probability; in addition, \( \Pr(\mathcal{Z}^{(\pi_{\text{IBPSP}})}) \rightarrow 1 \) is always equal to zero in the ideal world. Therefore, we readily obtain that \( \Pr(\mathcal{Z}^{(\pi_{\text{IBPSP}})}) \rightarrow 1 - \Pr(\mathcal{Z}^{(\mathcal{F}_{\text{IBPSP}})}) \rightarrow 1 \) is negligible.

**Theorem 10.** In the UC framework, \( \pi_{\text{IBPSP}} \) satisfies the existential unforgeability against adaptive chosen-message attacks.

**Proof.** Assume there exists a forger. Here we construct an environment \( \mathcal{Z} \) and an adversary \( \mathcal{A} \) such that, for any adversary \( \mathcal{A} \), \( \mathcal{Z} \) cannot tell with nonnegligible probability whether it interacts with \( \pi_{\text{IBPSP}} \) and \( \mathcal{A} \) in the real-life model or with \( \mathcal{F}_{\text{IBPSP}} \) and \( \mathcal{A} \) in the ideal model. \( \square \)

\( \mathcal{Z} \) proceeds as below. On receiving a request on proxy signcryption of \( \sigma \), \( \mathcal{Z} \) activates \( P_p \) and outputs the ciphertext \( \sigma \) to \( \mathcal{A} \). On receiving a request on unsigncryption of \( \sigma \), \( \mathcal{Z} \) activates \( P_b \) and outputs \( m, f \) to \( \mathcal{A} \).

\( \mathcal{A} \) proceeds as follows. \( \mathcal{A} \) first invokes the forger. As the forger requests to signcryption a message \( m \), \( \mathcal{A} \) requests \( \mathcal{Z} \) to signcryption a message \( m \) and outputs the ciphertext \( \sigma' \) to the forger. On receiving a request on unsigncryption of the forger, \( \mathcal{A} \) requests \( \mathcal{Z} \) to unsigncryption the ciphertext \( \sigma' \) and outputs \( <m', f> \) to \( \mathcal{A} \) and simultaneously delivers \( \sigma' \) to the forger. Once the forger receives \( m' \) and \( f = 1 \), then the forgery \( \sigma' \) is valid and at this time \( \mathcal{Z} \) outputs \( f = 1 \). Clearly, if the forger wins UF-IBPSP-CMA-I or UF-IBPSP-CMA-I with probability \( \varepsilon \), then the forger can succeed in forging valid proxy signcryption. Assume that such a forger exists with nonnegligible probability \( \varepsilon \); then the probability that \( \mathcal{Z} \) outputs \( f = 1 \) is nonnegligible. However, the probability that \( \mathcal{Z} \) outputs \( f = 1 \) is always equal to zero in the ideal model. In other words, if such a forger exists, \( \mathcal{Z} \) can always tell with nonnegligible probability whether it interacts with \( \pi_{\text{IBPSP}} \) and \( \mathcal{A} \) or with \( \mathcal{F}_{\text{IBPSP}} \) and \( \mathcal{A} \). It contracts with the initial assumption in **Theorem 10**. As mentioned above, such a forger cannot exist; that is to say, \( \pi_{\text{IBPSP}} \) satisfies the existential unforgeability against adaptive chosen-message attacks in the UC framework.

### 7. Efficiency Analysis

In this section, we compare our protocol with similar protocols in terms of computational complexity together with UC security and semantic security (see Table 1). In Table 1, \( E \) is one exponent operation, \( P \) is one pairing operation in \( G_2 \), \( M \) is one scalar multiplication operation in \( G_1 \), and \( H \) is one hash operation. Moreover, “\( \checkmark \)” means that one cryptography protocol satisfies the relevant security and “\( \times \)” means that one cryptography protocol does not satisfy the relevant security.

Let \( t_H \) denote the time cost of one hash operation. According to the literature [21], we can summarize and deduce the time cost of all operations as follows: \( P \approx 1440t_H \), \( E \approx 21t_H \), \( M \approx 29t_H \), and \( H \approx t_H \). We can readily obtain that the time cost of our protocol is \( 10323t_H \). Following the comparison analysis, there are 13253 \( t_H \) and 14640 \( t_H \) in protocols [4, 19], respectively.

From the comparison in Table 1, we obtain that the computation cost of our IBPSP is lower than those of other protocols. It is known from Table 1 that our IBPSP has both semantic security and UC security, but other protocols cannot guarantee the security in the UC framework. All the protocols in Table 1 satisfy the semantic security in the random oracle model.

### 8. Summary

The UC security framework provides the theory basis for the design of protocols in a complex and unpredictable environment. In this paper, we construct an IBPSP with universal composability and prove its semantic security in the random oracle model. We also define the ideal functionality of the IBPSP and prove the equivalence between IBPSP with universal composability and its IND-CCA2 and UF-CMA security. Our IBPSP can guarantee security even when it is composed of an arbitrary set of protocols, or when it is used as a component of an arbitrary system. What is more, our IBPSP can guarantee security even when it concurrently runs with an unbounded number of protocol instances.

### Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

### Conflicts of Interest

The authors declare that they have no conflicts of interest.
Acknowledgments
This work was supported by the Project of Basic Research of Qinghai Province under Grant 2016–ZJ–776 and the Chunhui Project of the Ministry of Education under Grant Z2017052.

References


