IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 14, 2020, accepted July 24, 2020, date of publication August 11, 2020, date of current version August 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3015893

Revocable Certificateless Public Key Encryption With Outsourced Semi-Trusted Cloud Revocation Agent

MINGXIN MA[1], (Graduate Student Member, IEEE), GUOZHEN SHI[2], XINYI SHI[2], MANG SU[3], AND FENGHUA LI[4], (Member, IEEE)

[1]School of Cyber Engineering, Xidian University, Xi'an 710071, China
[2]School of Information Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China
[3]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[4]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Corresponding author: Fenghua Li (lfh@iie.ac.cn)

This work was supported in part by the National Key Research Program of China under Grant 2017YFB0802705, Grant 2017YFB0801803, and Grant 2016YFB0800304, and in part by the National Natural Science Foundation of China under Grant 61702266.

ABSTRACT Certificateless public key cryptography (CL-PKC) not only eliminates the need for certificates in traditional certificate-based PKC but also solves the inherent key escrow problem in identity-based PKC. However, an unsolved but critical issue in CL-PKC is how to revoke a misbehaving user. Some revocable certificateless public key encryption (RCL-PKE) schemes have been proposed, but these schemes have two main drawbacks: 1) public key uniqueness is not guaranteed, thus allowing the existence of multiple copies of each initial secret key. 2) The existing outsourced RCL-PKE schemes place excessive trust in the cloud server, which may continue to update decryption keys stealthily for misbehaving users. In this paper, we address these issues by proposing a novel RCL-PKE with semi-trusted cloud revocation agents (s-CRAs). We describe the framework and the security model for the RCL-PKE with s-CRA and prove that the proposed scheme is semantically secure against adaptive chosen-ciphertext attacks under the bilinear Diffie-Hellman assumption in the random oracle model. Furthermore, we compare the proposed scheme with previous RCL-PKE schemes in terms of performance and robustness. The evaluation results show that the proposed scheme achieves public key uniqueness and reliable revocation flexibility at low computational and communication costs.

INDEX TERMS Certificateless encryption, revocable, random oracle, cloud computing, outsourcing.

I. INTRODUCTION

In traditional public key infrastructure (PKI), certificates are used to bind the public keys to the identities of the holders of the corresponding private keys and provide an assurance of these relationships by signing the certificates by a Certification Authority (CA). However, senders must verify the validity of the certificates before encrypting messages by the receiver's public key, and the management of the certificates raises a number of issues. First, it is challenging to handle a large number of certificates in the case of large distributed systems, especially the verification of certificate chains signed by intermediate CAs. Second, certificate management operations, such as revoking misbehaving users, lead to too much overhead. Efficient revocation has been well studied in traditional PKI [7], [8], [19]–[24], and some mechanisms, such as the certificate revocation list (CRL) and online certificate status protocol (OCSP), have been introduced to PKI to revoke users. Therefore, the system robustness is enhanced by introducing these methods but lead to large computational and communication costs.

To eliminate the need for certificate, Shamir [25] proposed identity-based public key cryptography (ID-PKC) in which the user's public key is its identity information. Thereafter, Boneh and Franklin [1] proposed the first practical identity-based encryption (IBE) scheme by using Weil pairing on elliptic curves. Subsequently, many identity-based public key schemes have been proposed [2], [3], [5], [6],

The associate editor coordinating the review of this manuscript and approving it for publication was Ana Lucila Sandoval Orozco.

[8]–[12], [16], [17], [26], [27], [42], [43]. In ID-PKC, private keys are generated for entities by a trusted third party called private key generator (PKG). However, the dependence on PKG, which uses a system-wide master key to generate private keys, inevitably introduces key escrow problem to ID-PKC systems in the sense that the PKG can decrypt any ciphertext for any user or forge any entity's signatures [32]. To address the built-in key escrow feature of ID-PKC, certificateless public key cryptography (CL-PKC) was proposed by Al-Riyami and Paterson [33]. The core concept of the scheme to eliminate the need for a certificate is that the user's public key is implicitly bound to user identity (that is to say, the ciphertext corresponds to both the public key and the user's identity). Meanwhile, the user's private key consists of two components, a partial private key and a secret value, which are dominated by the PKG and the user, respectively.

Hence, CL-PKC avoids the key escrow problem in ID-PKC and does not suffer the overhead of certificate management operations, which makes it a promising public key paradigm. Since then, CL-PKC has attracted much attention, and many studies have been published [4], [13], [14], [18], [30]–[41].

A. RELATED WORK

Due to the security threats posed by expired or misbehaving users, the public key system (PKS) must provide a revocation mechanism to revoke them. However, CRL and OCSP are not suitable for CL-PKC, and there is little work in the literature about the revocation mechanism in CL-PKC. Since the revocation mechanism of CL-PKC is essentially identical to ID-PKC, we discuss a few recently proposed revocable schemes of both paradigms.

Boneh and Franklin [1] suggested a trivial revocation mechanism in which the public key is composed of user identity and the current time period, and users update their private keys periodically. Unfortunately, PKG must generate keys and send them securely for all non-revoked users. Boneh *et al.* [2] and Libert *et al.* [3] presented another way to revoke users where each ciphertext is decrypted with the help of a semi-trusted third party called a mediator who holds shares of all users' private keys. The revocation is achieved when the mediator simply stops issuing the secret shares of revoked users. This method was developed by Ju *et al.* [4] to construct a mediated certificateless public key encryption scheme (mCL-PKE). However, the mediator must remain online and remains the bottleneck of the network.

To mitigate the workload of the PKG for key updates and enable non-revoked users to decrypt ciphertext of their own, Boldyreva *et al.* [5] proposed the first revocable IBE (RIBE) scheme proved in the selective-revocable-ID (sRID) model. The idea of their RIBE is based on the Fuzzy IBE [6] and decreases the total number of participants of key updates from linear to logarithmic by introducing a binary complete subtree [7], [8]. Nevertheless, their scheme suffers from significant computational and communication overhead: 1) The scheme requires 12 exponentiations to encrypt a message and 4 pairing computations to decrypt a message. 2) Each user needs to store up to $3 \log n$ elliptic curve elements as secret keys, where n is the number of users. These drawbacks make the scheme unacceptable for resource-constrained environments. Thereafter, Libert and Vergnaud [9] enhanced the security notions and proposed an adaptive-ID secure RIBE scheme. In consideration of the delegation of both key generation and revocation functionalities, Seo and Emura [10] proposed the revocable hierarchical IBE (RHIBE). Then, the authors reviewed their implements and presented a new method of history-free update construction [11].

To remove the secure channel between each user and the PKG to securely transmit the user's periodic private keys, Tseng and Tsai [12] proposed an RIBE scheme with a public channel. Furthermore, they transplanted this concept to CL-PKC and proposed the first revocable certificateless public key encryption (RCL-PKE) [13]. The authors provided an efficient revocation method to divide the user private key into an identity-related initial key and a time-related time update key. The identity key is associated with user identity and is fixed, which is securely transmitted to the user. The time update key is associated with the current time period and each user's identity. The PKG periodically generates current time update keys for non-revoked users and publishes them through a public channel. Shen *et al.* [14] proposed an efficient CCA2-secure RCL-PKE scheme in the standard model based on the decisional truncated q-ABDHE assumption and decisional bilinear Diffie-Hellman (DBDH) assumption. However, Shen *et al.'s* scheme was later shown to be insecure [15].

With the rapid development of cloud computing, some schemes outsource complicated computing tasks (*i.e.* key updates) to the powerful cloud server. By introducing a key update cloud service provider (KU-CSP) to IBE, Li *et al.* [16] proposed an outsourced cloud-aided revocable IBE scheme. They used a similar technique adopted in [12] and migrated the load to the cloud server to mitigate the load of the PKG. Hereafter, Tseng *et al.* [17] took advantage of [16] to propose another cloud-aided RIBE with cloud revocation authority (CRA), which overcomes the shortcomings in [16] that KU-CSP must keep a time key for each user and consequently unscalability. Recently, Tseng and Tsai [18] further proposed a delegated RCL-PKE scheme in the standard model to reduce the workload of the PKG.

B. MOTIVATIONS

To the best of our knowledge, the revocation mechanism in CL-PKC has not received much attention. The proposed RCL-PKE schemes have two main drawbacks.

First, the public key uniqueness is not guaranteed since the public key is not bound to the initial secret key in [13], [14], [18] (*i.e.*, the initial secret key in [13], [18] and the initial partial private key in [14]); the public key generation algorithm in these works allows users to create more than one public key for the same initial secret key. Conversely, the PKG may replace the public key to impersonate legal users.

Second, the efficient outsourced schemes [12], [16], [18] placed excessive trust in the cloud server. In other words, the cloud server in these schemes is regarded as a secure, well-resourced and patulous PKG. In fact, we cannot expect the cloud to be exactly honest with the PKG. Although a curious cloud server cannot decrypt messages of certain users, it can still generate time update keys for revoked users secretively to obtain illegal income, especially when multiple cloud servers are deployed in the system and share the common master time key.

C. OUR CONTRIBUTIONS

In this paper, we propose a novel revocable certificateless public key encryption scheme with outsourced semi-trusted cloud revocation agent (s-CRA) to address the multiple public key and dishonest cloud server problems mentioned before. We investigate the security notions for RCL-PKE [13], [14], [18], [29], [30] and enhance our scheme to resist new possible threats, *i.e.*, the greedy cloud server. In our new RCL-PKE security model, we consider four types of adversaries: the Type-I adversary (the revoked user), Type-II adversary (outside attacker), Type-III adversary (the curious PKG) and Type-IV adversary (the greedy cloud revocation agent). We describe our proposed scheme in detail and analyze its security under the bilinear Diffie-Hellman (BDH) assumption. We prove that our scheme is semantically secure against adaptive CCA in the random oracle model. We also provide performance evaluation of our proposed scheme and comparison with other RCL-PKE schemes.

D. ORGANIZATION

The rest of the paper is organized as follows. Section II provides preliminary information, including the definition of bilinear pairing and computational assumptions. Section III describes the framework of an RCL-PKE scheme with outsourced semi-trusted cloud revocation agent and its security model. Section IV describes our proposed RCL-PKE scheme in detail. We analyze the security of our scheme in section V. Section VI presents the performance evaluation and comparisons of our scheme. Finally, the conclusions are offered in Section VII.

II. PRELIMINARIES

In this section, we introduce the concept of bilinear pairings and computational assumptions required in this paper.

A. BILINEAR PAIRINGS

Let $\mathbb{G}$ be an additive cyclic group, whose order is a large prime q. P is a generator of $\mathbb{G}$. $\mathbb{G}_T$ is a multiplicative cyclic group of the same order q. A bilinear pairing is defined to be a map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ if it satisfies:

1) Bilinearity: $\hat{e}(xP, yQ) = \hat{e}(P, Q)^{xy}$ for all $P, Q \in \mathbb{G}$, $x, y \in Z_q^*$.

2) Non-degeneracy: there exists $P, Q \in \mathbb{G}$ such that $\hat{e}(P, Q) \neq 1_{\mathbb{G}_T}$.

3) Computability: there exists polynomial time algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}$.

B. COMPUTATIONAL ASSUMPTIONS

Bilinear Diffie-Hellman (BDH) problem. Given P, aP, bP, $cP \in \mathbb{G}$ for unknown $a, b, c \in Z_q^*$, computing $\hat{e}(P, P)^{abc} \in \mathbb{G}_T$.

Definition 1 (BDH Assumption): Given P, aP, bP, $cP \in \mathbb{G}$ for some $a, b, c \in Z_q^*$, there is no probabilistic polynomial time (PPT) algorithm that can solve BDH problem. The advantage of adversary A is $Adv_A = \Pr\left[A(\langle P, aP, bP, cP\rangle) = \hat{e}(P, P)^{abc}\right]$.

III. SYSTEM FRAMEWORK AND SECURITY MODEL

In this section, we describe the system framework of RCL-PKE with outsourced s-CRA and its security model.

A. SYSTEM FRAMEWORK

The proposed scheme involves three parties: the PKG, the s-CRAs and the users. At the beginning of the system initialization stage, the PKG generates and publishes system parameters and sends a secret master time key share to each s-CRA. Then, the user selects a secret value and inputs the system parameters to output its public key. The PKG then issues the partial identity key for each user with its master secret key, the user's identity and the user's public key. Each s-CRA issues and updates the user's time update key share with its own master time key share according to the revocation list. If a user is revoked, the honest s-CRA refuses to generate the time update key share for it. If a user receives t time update key shares from n s-CRAs, where n is the number of s-CRAs and t is the threshold, the user can recover its own time update key and decrypt messages. The system framework is presented in Fig. 1.

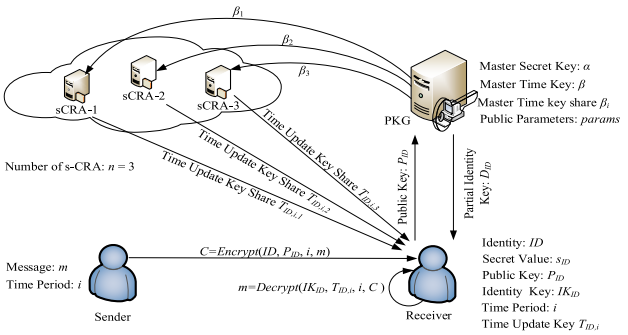

FIGURE 1. Revocable certificateless public key encryption with outsourced semi-trusted cloud revocation agent.

Definition 2: An RCL-PKE with outsourced s-CRA is a seven-tuple of polynomial time algorithms (System setup, Set public key, Partial identity key extract, Identity key extract, Time key update, Encryption and Decryption) as follows.

1) System setup. The PKG takes a security parameter λ, the number of s-CRAs n, the threshold t, and the total number of time periods I as inputs. The algorithm outputs a master secret

key α, a master time key β, the partitions of master time key $\beta_j (j = 1 \ldots n)$, and public parameters *params*. *params* are published to all users in the system.

2) Set public key. The user with identity *ID* runs this algorithm, takes *params* and the randomly selected secret value s_{ID} as inputs and outputs its public key P_{ID}.

3) Partial identity key extract. The PKG takes the master secret key α, user identity *ID* and user public key P_{ID} as inputs. The algorithm outputs the user's partial identity key D_{ID} and sends it via a secure channel.

4) Identity key extract. The user with identity *ID* runs this algorithm, takes partial identity key D_{ID} and its secret value s_{ID} as inputs and outputs the user's identity key IK_{ID}.

5) Time key update. For a time period i, each s-CRA takes its own time update key share β_j and a user's identity *ID* as inputs. The algorithm outputs the set of user's time update key shares $\langle T_{ID,i,j} \rangle$ corresponding to the time update key $T_{ID,i}$.

6) Encryption. For a time period i, the algorithm takes a user's identity *ID*, user's public key P_{ID}, and a message M as inputs. The algorithm outputs a ciphertext C.

7) Decryption. The algorithm takes a ciphertext C, the user's identity key IK_{ID}, and the user's time update key $T_{ID,i}$ as inputs. The algorithm outputs a plaintext M or outputs $\perp$ indicating a decryption failure.

B. SECURITY MODEL

We modify the security notions for RCL-PKE [13], [14], [18], [29] to enhance our scheme to resist new possible threats, *i.e.*, the greedy cloud server. The RCL-PKE is semantically secure against an adaptive RCL indistinguishability CCA (RCL-IND-CCA) adversary if no PPT adversary A has a nonnegligible advantage against challenger B in the RCL-IND-CCA game. We first consider the following four types of adversaries.

1) TYPE-I ADVERSARY A_I (A REVOKED USER)

This adversary is defined as a revoked user with identity ID^* who used to be a legal user and has been revoked by the PKG at some time period i^*. Such an adversary tries to obtain valuable information from ciphertext encrypted at or after period i^* and may collude with others. Therefore, such a Type-I adversary is allowed to obtain the identity key of every user and is able to obtain the time update key of all the users at arbitrary periods, except the target time update key of identity ID^* at period i^*.

2) TYPE-II ADVERSARY A_{II} (AN OUTSIDE ATTACKER)

This adversary is defined as an outside attacker who aims to obtain valuable information from the ciphertext of the target identity ID^*. Since the time update keys are published via a public channel, the attacker can obtain the time update key for the target identity. Therefore, such a Type-II adversary can obtain all of the time update keys and the identity key of any user, except the target identity ID^*.

3) TYPE-III ADVERSARY A_{III} (A CURIOUS PKG)

This adversary is defined as a curious PKG who does have access to the master secret key α and master time key β. Such a Type-III adversary can compute partial identity keys and generate a time update key for self-use. The adversary can also request public keys and make identity key extraction queries and decryption queries of its choice.

4) TYPE-IV ADVERSARY A_{IV} (A GREEDY s-CRA)

This adversary is defined as a greedy s-CRA who does have access to its own time update key share β_j and stealthily issues the time key update for illegal users. Although the greedy s-CRA cannot decrypt the ciphertext of the target user ID^*, the agent helps the target user obtain the plaintext. That is, the adversary is allowed to obtain the identity key of every user and is able to obtain the numbered time update key shares of the target user.

We define the security model of an RCL-PKE with outsourced s-CRA through the following game between a challenger and one of the above adversaries.

Definition 3 (RCL-IND-CCA): We say that an RCL-PKE is semantically secure against an adaptive RCL indistinguishability CCA (RCL-IND-CCA) adversary if no PPT adversary A has a nonnegligible advantage against the challenger B in following the RCL-IND-CCA game.

System setup. The challenger B runs the system setup algorithm to output $\langle \mathbb{G}, \mathbb{G}_T, \hat{e} \rangle$, and outputs a group generator $P \in \mathbb{G}$, the master secret key α, the master time key β, the partitions of master time key $\beta_j (j = 1 \ldots n)$, and the public parameters *params*. Then, it forwards *params* to the adversary A. Meanwhile, B gives master secret key α to A if A is the Type-III adversary, or B gives at most t master time key shares $\beta_j (j = 1 \ldots t, t < n)$ to A if A is the Type-IV adversary. Otherwise, B keeps α and β secret.

Phase 1.

The adversary A is allowed to issue the following queries in an adaptive manner.

Public key request query (*ID*). Upon receiving such a query for identity $ID \in \{0, 1\}^*$, B runs the public key extract algorithm to generate a proper public key P_{ID} and sends back to A.

Public key replace query (ID, P'_{ID}). The adversary A is allowed to replace the public key associates with identity *ID* to P'_{ID} of its choice. The challenger B uses the current value of an entity's public key in any computations (*e.g.* preparing a challenge ciphertext) or responses to A's requests (*e.g.* replying to a request for the public key). Note that A cannot both replace the public key for the challenge identity ID^* before the challenge phase and extract the partial identity key for ID^* in some phase.

Partial identity key extract query (ID, P_{ID}). When A issues such a query with (ID, P_{ID}), B runs the partial identity key extract algorithm to generate the partial identity key D_{ID} and sends back to A.

Secret value extract query (*ID*). Upon receiving such a query for identity $ID \in \{0,1\}^*$, B returns the associated secret value s_{ID} to A. The restriction is that A cannot extract the secret value s_{ID^*} of ID^* which has been queried the public key replace query.

Time key update query (*ID*, *i*). When A issues such a query for identity $ID \in \{0,1\}^*$ and a period i^*, B runs the time key update algorithm to generate the proper time update key $T_{ID,i}$ and sends back to A.

Decryption query (*C*, *ID*, *i*). While receiving the query along with the ciphertext C, the identity $ID \in \{0,1\}^*$ and the period i, B runs the key extract algorithm and the time key update algorithm to obtain the private key pair $(IK_{ID}, T_{ID,i})$. Then, the challenger B runs the decryption algorithm to decrypt the ciphertext C and returns the plaintext M to A.

Challenge. Adversary A generates two different plaintexts (M_0, M_1) of the same bitlength, then A sends a target identity ID^*, a target period i^* and (M_0, M_1) to the challenger B. The challenger B flips a random coin $\gamma \in (0, 1)$ and computes the ciphertext $C^* = E\left(ID^*, P_{ID^*}, i^*, M_\gamma\right)$ and returns C^* to A. The restrictions for different types of adversaries are as follows.

1) If A is the Type-I adversary, we require that A cannot issue the time key update query with (ID^*, i^*) in Phase 1.

2) If A is the Type-II adversary, we require that A cannot issue the partial identity key extract query with ID^* in Phase 1.

3) If A is the Type-III adversary, we require that A cannot simultaneously issue the secret value extract query and public key replace query with ID^* in Phase 1.

4) If A is a Type-IV adversary, it is partially identical to the Type-I adversary.

Phase 2. The adversary adaptively issues more queries as in phase 1 with the restriction that A cannot issue the decryption query with (ID^*, i^*, C^*). Other restrictions are the same with those in Phase 1 and the Challenge phase.

Guess. A outputs a guess $\gamma' \in (0, 1)$. If $\gamma' = \gamma$, the adversary A wins the game.

The advantage of adversary A in attacking the RCL-IND-CCA scheme is defined as $Adv_A(\lambda) = \Pr\left[\gamma' = \gamma\right] - 1/2$.

IV. PROPOSED RCL-PKE SCHEME

In this section, we describe our proposed RCL-PKE with outsourced s-CRA scheme. As defined in section III, our scheme is specified by seven algorithms: System setup, Set public key, Partial identity key extract, Identity key extract, Time key update, Encryption and Decryption algorithms.

System setup:

1. The PKG takes a security parameter λ to generate $\langle \mathbb{G}, \mathbb{G}_T, \hat{e} \rangle$ where $\mathbb{G}$ is an additive cyclic group and $\mathbb{G}_T$ is a multiplicative cyclic group of prime order $q > 2^\lambda$, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map.

2. The PKG chooses an arbitrary generator $P \in \mathbb{G}$.

3. The PKG takes a maximal period number I as input and randomly chooses two secret values $\alpha, \beta \in \mathbb{Z}_q^*$ and sets $P_0 = \alpha \cdot P$, the cloud public key $C_{pub} = \beta \cdot P$. α and β are the master secret key and the master time key, respectively.

4. the PKG selects four cryptographic hash functions $H_1, H_2 : \{0,1\}^* \to \mathbb{G}$, $H_3 : \mathbb{G}_T \to \{0,1\}^l$, $H_4 : \{0,1\}^* \to \{0,1\}^l$. Here, l will be the bit-length of plaintexts.

5. The PKG takes the threshold t and randomly generates a $t-1$ degree polynomial $f(x) = \beta + a_1 x + \cdots + a_{t-1} x^{t-1}$. Then, the PKG evaluates the master time key share $\beta_j = f(j)$ $(j = 1, \ldots n)$, where n is the number of s-CRAs.

Thereafter, the PKG securely transmits the master time key share β_j to s-CRA_j. The system parameters are $params = \langle \mathbb{G}, \mathbb{G}_T, \hat{e}, P, P_o, C_{pub}, n, t, H_1, H_2, H_3, H_4 \rangle$. The message space is $\mathcal{M} = \{0,1\}^l$ and the ciphertext space is $\mathcal{C} = G \times \{0,1\}^{2l}$.

Set public key:

This algorithm takes *params* and the user's randomly selected secret value $s_{ID} \in Z_q^*$ as inputs and outputs the public key as $P_{ID} = \langle X_{ID} Y_{ID} \rangle$, where $X_{ID} = s_{ID} P$ and $Y_{ID} = s_{ID} P_0 = s_{ID} \alpha P$.

Partial identity key extract:

Upon receiving the public key P_{ID} of an authorized user with identity $ID \in \{0,1\}^*$, the PKG uses the master secret key α to compute the corresponding partial identity key D_{ID} by following steps:

1. Computes $Q_{ID} = H_1(ID, P_{ID}) \in \mathbb{G}$.

2. Outputs the partial identity key $D_{ID} = \alpha \cdot Q_{ID}$ and sends it to the user in a secure channel.

Notice that user can verify the correctness of the partial identity key D_{ID} by checking $\hat{e}(D_{ID}, P) = \hat{e}(Q_{ID}, P_0)$.

Identity key extract:

The algorithm takes *params*, the user's partial identity key D_{ID} and the user's secret value $s_{ID} \in Z_q^*$ as inputs. Then, computes the identity key $IK_{ID} = s_{ID} \cdot D_{ID} = s_{ID} \alpha Q_{ID} \in \mathbb{G}$.

Time key update:

To generate the time update key periodically, each s$-$CRA$_j$ uses its own master time key share β_j to compute the time update key share $T_{ID,i,j}$ at period i for the user with identity $ID \in \{0,1\}^*$ by following steps:

1. Computes $R_{ID,i} = H_2(ID, i) \in \mathbb{G}$.

2. Outputs the time update key share $T_{ID,i,j} = \beta_j \cdot R_{ID,i}$.

Finally, each s-CRA$_j$ sends the time update key share $T_{ID,i,j}$ to the user via a public channel. By interpolating t time update key shares, the valid user can recover the time update key $T_{ID,i}$.

$$\begin{aligned} T_{ID,i} &= \beta \cdot R_{ID,i} \\ &= \sum_{j=1}^{t} \prod_{k=1,k\neq j}^{t} \frac{-k}{j-k} \beta_j R_{ID,i} \\ &= \sum_{j=1}^{t} \beta_j R_{ID,i} \prod_{k=1,k\neq j}^{t} \frac{-k}{j-k} \\ &= \sum_{j=1}^{t} T_{ID,i,j} \prod_{k=1,k\neq j}^{t} \frac{-k}{j-k} \end{aligned}$$

Encrypt:

To encrypt a message $M \in \mathcal{M}$ for a receiver with identity ID and the public key $P_{ID} = \langle X_{ID} Y_{ID} \rangle$ at period i, the sender performs the following steps:

1. Checks $X_{ID}, Y_{ID} \in \mathbb{G}$ and the equality $\hat{e}(X_{ID}, P_0) = \hat{e}(Y_{ID}, P)$, *i.e.* P_{ID} is a correct public key. Otherwise, outputs $\perp$ and aborts.

2. Computes $Q_{ID} = H_1(ID, P_{ID}) \in \mathbb{G}$ and $R_{ID,i} = H_2(ID, i) \in \mathbb{G}$.

3. Chooses a random value $r \in Z_q^*$ and computes $U = r \cdot P$.

4. Computes and outputs the ciphertext:

$$\begin{aligned} g_1 &= \hat{e}(Q_{ID}, Y_{ID}) \\ g_2 &= \hat{e}\left(R_{ID,i}, C_{pub}\right) \\ V &= M \oplus H_3\left((g_1 \cdot g_2)^r\right) \\ W &= H_4(U, V, M, ID, P_{ID}, i) \\ C &= (U, V, W) \end{aligned}$$

Decrypt:

To decrypt a ciphertext $C = (U, V, W)$ for identity ID and the public key $P_{ID} = \langle X_{ID} Y_{ID} \rangle$ at period i, the receiver uses its identity key IK_{ID} and time update key $T_{ID,i}$ as follows:

1. computes $M' = V \oplus H_3\left(\hat{e}\left(IK_{ID} + T_{ID,i}, U\right)\right)$.

2. computes $W' = H_4\left(U, V, M', ID, P_{ID}, i\right)$.

3. if $W' = W$, outputs M' as the decryption of C. Otherwise, outputs $\perp$ and rejects the ciphertext.

V. CORRECTNESS AND SECURITY ANALYSIS

In this section, we present the correctness and security analysis of our proposed scheme.

1) The correctness of the decrypt algorithm due to the fact that:

$$\begin{aligned} & H_3\left(\hat{e}\left(IK_{ID} + T_{ID,i}, U\right)\right) \\ &= H_3\left(\hat{e}\left(IK_{ID}, U\right) \cdot \hat{e}\left(T_{ID,i}, U\right)\right) \\ &= H_3\left(\hat{e}\left(s_{ID}\alpha Q_{ID}, rP\right) \cdot \hat{e}\left(\beta R_{ID,i}, rP\right)\right) \\ &= H_3\left(\hat{e}\left(Q_{ID}, s_{ID}\alpha P\right)^r \cdot \hat{e}\left(R_{ID,i}, \beta P\right)^r\right) \\ &= H_3\left(\hat{e}\left(Q_{ID}, Y_{ID}\right)^r \cdot \hat{e}\left(R_{ID,i}, \beta P\right)^r\right) \\ &= H_3\left((g_1 \cdot g_2)^r\right) \end{aligned}$$

2) The security analysis consists of four lemmas against Type I, II, III and IV adversaries. Then, we conclude that the proposed revocable certificateless public key encryption scheme with outsourced semi-trusted cloud revocation agent scheme is secure in the sense of RCL-IND-CCA adversary. For simplicity but without loss of generality, we replace the master time key shares β_j to master time key β in the proof of the first three lemmas and prove its correctness in lemma 4.

Lemma 1: In the random oracle model, assume that a Type-I adversary A_I in attacking the proposed revocable certificateless encryption in the sense of RCL-IND-CCA security. We will build a simulator B_I to solve the BDH problem with a non-negligible probability.

Proof: Suppose that there exists a Type-I adversary A_I with advantage ϵ_I who can break the proposed Revocable Certificateless Encryption. We will build a simulator B_I to solve the BDH problem with advantage ϵ_I'. The simulator B_I inputs BDH parameters $\langle \mathbb{G}, \mathbb{G}_T, \hat{e} \rangle$ and $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$ where P is a generator of group $\mathbb{G}$. We say the simulator B_I can solve the BDH problem if B_I has a non-negligible advantage to compute $\hat{e}(P, P)^{abc}$.

System setup. The challenger B_I first chooses the random master secret key $\alpha \in \mathbb{Z}_q^*$ and set $C_{pub} = aP$. Then B_I provides A_I with $params = \langle \mathbb{G}, \mathbb{G}_T, \hat{e}, P, P_o, C_{pub}, H_1, H_2, H_3, H_4 \rangle$. A_I is allowed to issue queries in the following types controlled by B_I.

H_1-queries: B_I maintains a list $\mathcal{L}_1$ of $\langle ID, P_{ID}, Q_{ID}, u \rangle$ to store the answers to the hash oracle H_1. Upon receiving the H_1-query along with (ID, P_{ID}), B_I performs a check on $\mathcal{L}_1$. If (ID, P_{ID}) appears in $\mathcal{L}_1$, then B_I responds with $H_1(ID, P_{ID}) = Q_{ID}$. Otherwise, B_I randomly selects $u \in \mathbb{Z}_q^*$ and computes $Q_{ID} = u \cdot P$. After storing $\langle ID, P_{ID}, Q_{ID}, u \rangle$ in $\mathcal{L}_1$, B_I returns $H_1(ID, P_{ID}) = Q_{ID}$ to A_I.

H_2-queries: B_I maintains a list $\mathcal{L}_2$ of $\langle ID, i, R_{ID,i}, v, coin \rangle$ to store the answers to the hash oracle H_2. Upon receiving the H_2-query along with $\langle ID, i \rangle$, B_I performs a check on $\mathcal{L}_2$. If $\langle ID, i \rangle$ appears in $\mathcal{L}_2$, then B_I responds with $H_2(ID, i) = R_{ID,i}$. Otherwise, B_I randomly selects $v \in \mathbb{Z}_q^*$, then B_I flips a random $coin \in \{0, 1\}$ and sets $R_{ID} = v \cdot P$ if $coin = 0$ and $R_{ID,i} = v \cdot bP$ if $coin = 1$. After storing $\langle ID, i, R_{ID,i}, v \rangle$ in $\mathcal{L}_2$, B_I returns $H_2(ID, i) = R_{ID,i}$ to A_I.

H_3-queries: B_I maintains a list $\mathcal{L}_3$ of $\langle X, Y \rangle$ to store the answers to the hash oracle H_3. Upon receiving the H_3-query along with X, B_I performs a check on $\mathcal{L}_3$. If X appears in $\mathcal{L}_3$, then B_I responds with $H_3(X) = Y$. Otherwise, B_I randomly chooses a string $Y \in \{0, 1\}^l$. After storing $\langle X, Y \rangle$ in $\mathcal{L}_3$, B_I returns $H_3(X) = Y$ to A_I.

H_4-queries: B_I maintains a list $\mathcal{L}_4$ of $\langle U, V, M, ID, i, w \rangle$ to store the answers to the hash oracle H_4. Upon receiving the H_4-query along with $\langle U, V, M, ID, i \rangle$, B_I performs a check on $\mathcal{L}_4$. If $\langle U, V, M, ID, i \rangle$ appears in $\mathcal{L}_4$, then B_I responds with $H_4(U, V, M, ID, i) = w$. Otherwise, B_I randomly chooses a string $w \in \{0, 1\}^l$. After storing $\langle U, V, M, ID, i, w \rangle$ in $\mathcal{L}_4$, B_I returns $H_4(U, V, M, ID, i) = w$ to A_I.

Phase 1. B_I replies to the four queries as follows.

Public key request query (ID). To respond to such a query, the challenger B_I maintains a list $\mathcal{L}_{PK}$ of $\langle ID, P_{ID}, s_{ID} \rangle$. B_I first accesses the list $\mathcal{L}_{PK}$, if ID already appears in the $\mathcal{L}_{PK}$, then B_I responds with P_{ID}. Otherwise, B_I randomly selects $s_{ID} \in \mathbb{Z}_q^*$ and computes $P_{ID} = s_{ID} \cdot P$. After storing $\langle ID, P_{ID}, s_{ID} \rangle$ in $\mathcal{L}_{PK}$, B_I returns P_{ID} to A_I.

Public key replace query (ID, P'_{ID}). Upon receiving such a query with (ID, P'_{ID}), B_I replaces the tuple $\langle ID, P_{ID}, s_{ID} \rangle$ in $\mathcal{L}_{PK}$ list to $\langle ID, P_{ID}, \perp \rangle$.

Partial identity key extract query (ID, P_{ID}). To respond to such a query, the challenger B_I first accesses the list $\mathcal{L}_1$ to obtain u. Then, B_I sets the partial identity key as $D_{ID} = \alpha \cdot Q_{ID} = \alpha \cdot u \cdot P$ which is a valid partial identity key. B_I returns the partial identity key D_{ID} to A_I.

Secret value extract query (ID). To respond to such a query, the challenger B_I first accesses the list $\mathcal{L}_{PK}$ to obtain s_{ID} associates with identity ID. If ID does not appear in $\mathcal{L}_{PK}$,

B_I issues public key request query with ID first. B_I returns the secret value s_{ID} to A_I.

Time key update query (ID, i). To respond to such a query, the challenger B_I first accesses the list $\mathcal{L}_2$ to obtain v and $coin$. If $coin = 1$, the simulation failures and aborts. Otherwise, B_I sets the time key as $T_{ID,i} = v \cdot C_{pub} = v \cdot aP = a \cdot vP = a \cdot R_{ID}$ which is a valid Time update key. B_I returns the Time update key $T_{ID,i}$ to A_I.

Decryption query ($C = \langle U, V, W\rangle, ID, i$). To respond to such a query, the challenger B_I first accesses the list $\mathcal{L}_4$ to obtain M corresponding to $\langle U, V, -, ID, i, W\rangle$. If M was not found, the simulation failures and aborts. Otherwise, B_I returns M to A_I.

Challenge. At some point, A_I decides to end Phase 1 and picks a target identity ID^* and a target period i^*, then it issues two messages M_0, M_1 to be challenged. We assume that A_I did not issue H_2 query to obtain the target time update key. B_I uses (ID^*, i^*) to scan the list $\mathcal{L}_2 = \langle ID, i, R_{ID}, v, coin\rangle$. If $coin = 0$, then the simulation failures and aborts. If $coin = 1$, B_I flips a random coin $\gamma \in (0, 1)$ and computes $V = M_\gamma \oplus Y^*$, where $Y^* = H_3\left(\hat{e}\left(Q_{ID}, s_{ID} \cdot \alpha \cdot cP\right) \cdot \hat{e}\left(v \cdot bP, aP\right)\right) = H_3(D)$. Then, B_I randomly selects a string $w \in \{0, 1\}^l$ and adds $\langle U = cP, V = M_\gamma \oplus Y^*, M_\gamma, ID^*, i^*, w\rangle$ in $\mathcal{L}_4$. Finally, B_I returns the target ciphertext $C^* = (U, V, W = w)$ to A_I.

Phase 2. B_I continues to respond to requests in the same way as it did in Phase 1. We restrict A_I cannot issue the time key update query with (ID^*, i^*) and the decryption query with (ID^*, i^*, C^*).

Guess. A_I will make a guess γ' for γ. The advantage ϵ_I of an IND-ID-CCA adversary A_I to attack the proposed revocable certificateless encryption scheme is evaluated by $Adv_{A_I} = \left|\Pr\left[\gamma' = \gamma\right] - \frac{1}{2}\right|$. If the adversary A_I who breaks the proposed scheme with a non-negligible advantage ϵ_I, then the challenger B_I can solve the BDH problem with a non-negligible advantage ϵ_I'.

The probability that B_I does not abort during the simulation is analyzed as follows. In Phase 1 and 2, if $coin = 1$, the simulation failures and aborts since challenger B_I cannot answer the correct time update query. Otherwise, the simulation continues. Let δ denotes the probability that $coin = 0$. Since the adversary A_I makes at most q_u and q_d queries to time update queries and decryption queries in Phase 1 and 2, respectively, the probability that the simulation does not abort is δ^{q_u}. In the challenge phase, if $coin = 1$, the simulation continues, which means the probability that the simulation does not abort is $1 - \delta$. Thus, the total probability that the simulation does not abort is $\delta^{q_u} \cdot (1 - \delta)$ in Game 1. By using the similar technique to Coron's analysis of the full domain hash signature scheme [44], the value is maximized at $\delta = 1 - 1/(q_u + 1)$ and the probability that B_I does not abort is at least $1/e(1 + q_u)$. Furthermore, the probability to guess the correct answer D in the real attack is at least $2\epsilon_I/q_3$ [1]. To respond to the decryption query, B_I scans the list $\mathcal{L}_4$ to obtain M. Since the simulation would success if $\langle U, V, -, ID, i, W\rangle$ appears in list $\mathcal{L}_4$ and there are at most q_d decryption queries, the probability the simulation aborts is q_d/q. In summary, B_I can solve the BDH problem with a non-negligible advantage $\epsilon_I' = 2\epsilon_I/e(1 + q_u)\, q_3 - q_d/q$.

Lemma 2: In the random oracle model, assume that an Type-II adversary A_{II} in attacking the proposed revocable certificateless encryption in the sense of RCL-IND-CCA security. We will build a simulator B_{II} to solve the BDH problem with a non-negligible probability.

Proof: Suppose that there exists a Type-II adversary A_{II} with advantage ϵ_{II} who can break the proposed Revocable Certificateless Encryption. We will build a simulator B_{II} to solve the BDHP problem with advantage ϵ_{II}'. The simulator B_{II} inputs BDHP parameters $\langle \mathbb{G}, \mathbb{G}_T, \hat{e}\rangle$ and $\langle P, aP, bP, cP\rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$ where P is a generator of group $\mathbb{G}$. We say the simulator B_{II} can solve the BDHP if B_{II} has a non-negligible advantage to compute $\hat{e}(P, P)^{abc}$.

System setup. The challenger B_{II} first chooses the random master time key $\beta \in Z_q^*$ and set $C_{pub} = \beta P$. Then B_{II} provides A_{II} with $params = \langle \mathbb{G}, \mathbb{G}_T, \hat{e}, P, P_o, C_{pub}, H_1, H_2, H_3, H_4\rangle$. A_{II} is allowed to issue queries in the following types controlled by B_{II}.

H_1-queries: B_{II} maintains a list $\mathcal{L}_1$ of $\langle ID, P_{ID}, Q_{ID}, u, coin\rangle$ to store the answers to the hash oracle H_1. Upon receiving the H_1-query along with ID, B_{II} performs a check on $\mathcal{L}_1$. If (ID, P_{ID}) appears in $\mathcal{L}_1$, then B_{II} responds with $H_1(ID, P_{ID}) = Q_{ID}$. Otherwise, B_{II} randomly selects $u \in Z_q^*$, then B_{II} flips a random $coin \in \{0, 1\}$ and sets $Q_{ID} = u \cdot P$ if $coin = 0$ and $Q_{ID} = u \cdot bP$ if $coin = 1$. After storing $\langle ID, P_{ID}, Q_{ID}, u, coin\rangle$ in $\mathcal{L}_1$, B_{II} returns $H_1(ID, P_{ID}) = Q_{ID}$ to A_{II}.

H_2-queries: B_{II} maintains a list $\mathcal{L}_2$ of $\langle ID, i, R_{ID,i}, v\rangle$ to store the answers to the hash oracle H_2. Upon receiving the H_2-query along with $\langle ID, i\rangle$, B_{II} performs a check on $\mathcal{L}_2$. If $\langle ID, i\rangle$ appears in $\mathcal{L}_2$, then B_{II} responds with $H_2(ID, i) = R_{ID,i}$. Otherwise, B_{II} randomly selects $v \in \mathbb{Z}_q^*$ and computes $R_{ID,i} = v \cdot P$. After storing $\langle ID, i, R_{ID,i}, v\rangle$ in $\mathcal{L}_2$, B_{II} returns $H_2(ID, i) = R_{ID,i}$ to A_{II}.

H_3-queries: B_{II} maintains a list $\mathcal{L}_3$ of $\langle X, Y\rangle$ to store the answers to the hash oracle H_3. Upon receiving the H_3-query along with X, B_{II} performs a check on $\mathcal{L}_3$. If X appears in $\mathcal{L}_3$, then B_{II} responds with $H_3(X) = Y$. Otherwise, B_{II} randomly chooses a string $Y \in \{0, 1\}^l$. After storing $\langle X, Y\rangle$ in $\mathcal{L}_3$, B_{II} returns $H_3(X) = Y$ to A_{II}.

H_4-queries: B_{II} maintains a list $\mathcal{L}_4$ of $\langle U, V, M, ID, i, w\rangle$ to store the answers to the hash oracle H_4. Upon receiving the H_4-query along with $\langle U, V, M, ID, i\rangle$, B_{II} performs a check on $\mathcal{L}_4$. If $\langle U, V, M, ID, i\rangle$ appears in $\mathcal{L}_4$, then B_{II} responds with $H_4(U, V, M, ID, i) = w$. Otherwise, B_{II} randomly chooses a string $w \in \{0, 1\}^l$. After storing $\langle U, V, M, ID, i, w\rangle$ in $\mathcal{L}_4$, B_{II} returns $H_4(U, V, M, ID, i) = w$ to A_{II}.

Phase 1. B_{II} replies to the four queries as follows.

Public key request query (ID). To respond to such a query, the challenger B_{II} maintains a list $\mathcal{L}_{PK}$ of $\langle ID, P_{ID}, s_{ID}\rangle$. B_{II} first accesses the list $\mathcal{L}_{PK}$, if ID already

appears in the $\mathcal{L}_{PK}$, then B_{II} responds with P_{ID}. Otherwise, B_{II} randomly selects $s_{ID} \in \mathbb{Z}_q^*$ and computes $P_{ID} = s_{ID} \cdot P$. After storing $\langle ID, P_{ID}, s_{ID}\rangle$ in $\mathcal{L}_{PK}$, B_{II} returns P_{ID} to A_{II}.

Public key replace query (ID, P'_{ID}). Upon receiving such a query with (ID, P'_{ID}), B_{II} replaces the tuple $\langle ID, P_{ID}, s_{ID}\rangle$ in $\mathcal{L}_{PK}$ list to $\langle ID, P_{ID}, \perp\rangle$.

Partial identity key extract query (ID, P_{ID}). To respond to such a query, the challenger B_{II} first accesses the list $\mathcal{L}_1$ to obtain u and $coin$. If $coin = 1$, the simulation failures and aborts. Otherwise, B_{II} sets the partial identity key as $D_{ID} = u{\cdot}aP = a{\cdot}Q_{ID}$ which is a valid partial identity key. B_{II} returns the partial identity key D_{ID} to A_{II}.

Secret value extract query (ID). To respond to such a query, the challenger B_{II} first accesses the list $\mathcal{L}_{PK}$ to obtain s_{ID} associates with identity ID. If ID does not appear in $\mathcal{L}_{PK}$, B_{II} issues public key request query with ID first. B_{II} returns the secret value s_{ID} to A_{II}.

Time key update query (ID, i). To respond to such a query, the challenger B_{II} first accesses the list $\mathcal{L}_2$ to obtain v. Then, B_{II} sets the time key as $T_{ID,i} = v \cdot C_{pub} = v \cdot \beta P = \beta \cdot vP = \beta \cdot R_{ID}$ which is a valid Time update key. B_{II} returns the Time update key $T_{ID,i}$ to A_{II}.

Decryption query ($C = \langle U, V, W\rangle, ID, i$). To respond to such a query, the challenger B_{II} first accesses the list $\mathcal{L}_4$ to obtain M corresponding to $\langle U, V, -, ID, P_{ID}, i, W\rangle$. If M was not found, the simulation failures and aborts. Otherwise, B_{II} returns M to A_{II}.

Challenge. At some point, A_{II} decides to end Phase 1 and picks a target identity ID^* and a target period i^*, then it issues two messages M_0, M_1 to be challenged. We assume that A_{II} did not issue H_1 query to obtain the target partial identity key. B_{II} uses ID^* to scan the list $\mathcal{L}_1 = \langle ID, Q_{ID}, u, coin\rangle$. If $coin = 0$, then the simulation failures and aborts. If $coin = 1$, B_{II} flips a random coin $\gamma \in (0, 1)$ and computes $V = M_\gamma \oplus Y^*$, where $Y^* = H_3\left(\hat{e}\left(s_{ID} \cdot u \cdot aP, bP\right) \cdot \hat{e}\left(R_{ID,i}, \beta \cdot cP\right)\right) = H_3(D)$. Then, B_{II} randomly selects a string $w \in \{0, 1\}^l$ and adds $\langle U = cP, V = M_\gamma \oplus Y^*, M_\gamma, ID^*, i^*, w\rangle$ in $\mathcal{L}_4$. Finally, B_{II} returns the target ciphertext $C^* = (U, V, W = w)$ to A_{II}.

Phase 2. B_{II} continues to respond to requests in the same way as it did in Phase 1. We restrict A_{II} cannot issue the partial identity key extract with ID^* and the decryption query with (ID^*, i^*, C^*).

Guess. A_{II} will make a guess γ' for γ. The advantage ϵ_{II} of an IND-ID-CCA adversary A_{II} to attack the proposed revocable certificateless encryption scheme is evaluated by $Adv_{A_{II}} = \left|\Pr\left[\gamma' = \gamma\right] - \frac{1}{2}\right|$. If the adversary A_{II} who breaks the proposed scheme with a non-negligible advantage ϵ_{II}, then the challenger B_{II} can solve the BDH problem with a non-negligible advantage ϵ'_{II}.

The probability that B_{II} does not abort during the simulation is analyzed as follows. In Phase 1 and 2, if $coin = 1$, the simulation failures and aborts since challenger B_{II} cannot answer the correct partial identity key extract query. Otherwise, the simulation continues. Let δ denotes the probability that $coin = 0$. Suppose the adversary A_{II} makes at most q_p and q_d queries to partial identity key extract queries and decryption queries in Phase 1 and 2, respectively, the probability that the simulation does not abort is δ^{q_p}. In the challenge phase, if $coin = 1$, the simulation continues, which means the probability that the simulation does not abort is $1 - \delta$. Thus, the total probability that the simulation does not abort is $p(\delta) = \delta^{q_p} \cdot (1 - \delta)$ in Game 2. By using the similar technique to Coron's analysis of the full domain hash signature scheme [44], the value is maximized at $\delta = 1 - 1/(q_p + 1)$ and the probability that B_{II} does not abort is at least $1/e(1 + q_p)$. Furthermore, the probability to guess the correct answer D in the real attack is at least $2\epsilon_{II}/q_3$ [1]. To respond to the decryption query, B_{II} scans the list $\mathcal{L}_4$ to obtain M. Since the simulation would success if $\langle U, V, -, ID, i, W\rangle$ appears in list $\mathcal{L}_4$ and there are at most q_d decryption queries, the probability the simulation aborts is q_d/q. In summary, B_{II} can solve the BDH problem with a non-negligible advantage $\epsilon'_{II} = 2\epsilon_{II}/e(1 + q_p)\, q_3 - q_d/q$.

Lemma 3: In the random oracle model, assume that an Type-III adversary A_{III} in attacking the proposed revocable certificateless encryption in the sense of RCL-IND-CCA security. We will build a simulator B_{III} to solve the BDH problem with a non-negligible probability.

Proof: Suppose that there exists a Type-III adversary A_{III} with advantage ϵ_{III} who can break the proposed Revocable Certificateless Encryption. We will build a simulator B_{III} to solve the BDHP problem with advantage ϵ'_{III}. The simulator B_{III} inputs BDHP parameters $\langle \mathbb{G}, \mathbb{G}_T, \hat{e}\rangle$ and $\langle P, aP, bP, cP\rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$ where P is a generator of group $\mathbb{G}$. We say the simulator B_{III} can solve the BDHP if B_{III} has a non-negligible advantage to compute $\hat{e}(P, P)^{abc}$.

System setup. The challenger B_{III} first chooses the random master secret key and a random master time key $\alpha, \beta \in Z_q^*$ respectively and set $C_{pub} = \beta P$. Then B_{III} provides A_{III} with $params = \langle \mathbb{G}, \mathbb{G}_T, \hat{e}, P, P_o, C_{pub}, H_1, H_2, H_3, H_4\rangle$. A_{III} is allowed to issue queries in the following types controlled by B_{III}.

H_1-queries: B_{III} maintains a list $\mathcal{L}_1$ of $\langle ID, P_{ID}, Q_{ID}, u\rangle$ to store the answers to the hash oracle H_1. Upon receiving the H_1-query along with ID, B_{III} performs a check on $\mathcal{L}_1$. If ID appears in $\mathcal{L}_1$, then B_{III} responds with $H_1(ID, P_{ID}) = Q_{ID}$. Otherwise, B_{III} randomly selects $u \in Z_q^*$ and computes $Q_{ID} = u{\cdot}P$. After storing $\langle ID, P_{ID}, Q_{ID}, u\rangle$ in $\mathcal{L}_1$, B_{III} returns $H_1(ID, P_{ID}) = Q_{ID}$ to A_{III}.

H_2-queries: B_{III} maintains a list $\mathcal{L}_2$ of $\langle ID, i, R_{ID,i}, v\rangle$ to store the answers to the hash oracle H_2. Upon receiving the H_2-query along with $\langle ID, i\rangle$, B_{III} performs a check on $\mathcal{L}_2$. If $\langle ID, i\rangle$ appears in $\mathcal{L}_2$, then B_{III} responds with $H_2(ID, i) = R_{ID,i}$. Otherwise, B_{III} randomly selects $v \in Z_q^*$ and computes $R_{ID,i} = v \cdot P$. After storing $\langle ID, i, R_{ID,i}, v\rangle$ in $\mathcal{L}_2$, B_{III} returns $H_2(ID, i) = R_{ID,i}$ to A_{III}.

H_3-queries: B_{III} maintains a list $\mathcal{L}_3$ of $\langle X, Y\rangle$ to store the answers to the hash oracle H_3. Upon receiving the H_3-query along with X, B_{III} performs a check on $\mathcal{L}_3$. If X appears in $\mathcal{L}_3$, then B_{III} responds with $H_3(X) = Y$. Otherwise, B_{III}

randomly chooses a string $Y \in \{0,1\}^l$. After storing $\langle X, Y\rangle$ in $\mathcal{L}_3$, B_{III} returns $H_3(X) = Y$ to A_{III}.

H_4-queries: B_{III} maintains a list $\mathcal{L}_4$ of $\langle U, V, M, ID, i, w\rangle$ to store the answers to the hash oracle H_4. Upon receiving the H_4-query along with $\langle U, V, M, ID, i\rangle$, B_{III} performs a check on $\mathcal{L}_4$. If $\langle U, V, M, ID, i\rangle$ appears in $\mathcal{L}_4$, then B_{III} responds with $H_4(U, V, M, ID, i) = w$. Otherwise, B_{III} randomly chooses a string $w \in \{0,1\}^l$. After storing $\langle U, V, M, ID, i, w\rangle$ in $\mathcal{L}_4$, B_{III} returns $H_4(U, V, M, ID, i) = w$ to A_{III}.

Phase 1. B_{III} replies to the four queries as follows.

Public key request query (*ID*). To respond to such a query, the challenger B_{III} maintains a list $\mathcal{L}_{PK}$ of $\langle ID, P_{ID}, s_{ID}, coin\rangle$. B_{III} first accesses the list $\mathcal{L}_{PK}$, if ID already appears in the $\mathcal{L}_{PK}$, then B_{III} responds with P_{ID}. Otherwise, B_{III} randomly selects $s_{ID} \in \mathbb{Z}_q^*$, then B_{III} flips a random $coin \in \{0,1\}$ and set $P_{ID} = s_{ID} \cdot P$ if $coin = 0$ and $P_{ID} = s_{ID} \cdot aP$ if $coin = 1$. After storing $\langle ID, P_{ID}, s_{ID}, coin\rangle$ in $\mathcal{L}_{PK}$, B_{III} returns P_{ID} to A_{III}.

Public key replace query (ID, P'_{ID}). Upon receiving such a query with (ID, P'_{ID}), B_{III} replaces the tuple $\langle ID, P_{ID}, s_{ID}\rangle$ in $\mathcal{L}_{PK}$ list to $\langle ID, P_{ID}, \perp\rangle$.

Partial identity key extract query (ID, P_{ID}). To respond to such a query, the challenger B_{III} first accesses the list $\mathcal{L}_1$ to obtain u. Then, B_{III} sets the partial identity key as $D_{ID} = \alpha \cdot Q_{ID} = \alpha \cdot u \cdot P$ which is a valid partial identity key. B_{III} returns the partial identity key D_{ID} to A_{III}.

Secret value extract query (*ID*). To respond to such a query, the challenger B_{III} first accesses the list $\mathcal{L}_{PK}$ to obtain s_{ID} and $coin$. If ID does not appear in $\mathcal{L}_{PK}$, then B_{III} issues public key request query with ID first. If $coin = 1$, the simulation failures and aborts. Otherwise, B_{III} returns the secret value s_{ID} to A_{III}.

Time key update query (*ID*, *i*). To respond to such a query, the challenger B_{III} first accesses the list $\mathcal{L}_2$ to obtain v. Then, B_{III} set the time key as $T_{ID,i} = v \cdot C_{pub} = v \cdot \beta P = \beta \cdot vP = \beta \cdot R_{ID}$ which is a valid Time update key. B_{III} returns the Time update key $T_{ID,i}$ to A_{III}.

Decryption query ($C = \langle U, V, W\rangle$, *ID*, *i*). To respond to such a query, the challenger B_{III} first accesses the list $\mathcal{L}_4$ to obtain M corresponding to $\langle U, V, -, ID, i, W\rangle$. If M was not found, the simulation failures and aborts. Otherwise, B_{III} returns M to A_{III}.

Challenge. At some point, A_{III} decides to end Phase 1 and picks a target identity ID^* and a target period i^*, then it issues two messages M_0, M_1 to be challenged. We assume that A_{III} did not issue secret value extract query to obtain the target secret value of ID^*. B_{III} uses ID^* to scan the list $\mathcal{L}_{PK} = \langle ID, P_{ID}, s_{ID}, coin\rangle$. If $coin = 0$, then the simulation failures and aborts. If $coin = 1$, B_{III} flips a random coin $\gamma \in (0,1)$ and computes $V = M_\gamma \oplus Y^*$, where $Y^* = H_3\left(\hat{e}(\alpha \cdot u \cdot s_{ID} \cdot aP, bP) \cdot \hat{e}\left(R_{ID,i}, \beta \cdot cP\right)\right)$. Then, B_{III} randomly selects a string $w \in \{0,1\}^l$ and adds $\langle U = cP, V = M_\gamma \oplus Y^*, M_\gamma, ID^*, , P_{ID}, i^*, w\rangle$ in $\mathcal{L}_4$. Finally, B_{III} returns the target ciphertext $C^* = (U, V, W = w)$ to A_{III}.

Phase 2: B_{III} continues to respond to requests in the same way as it did in Phase 1. We restrict A_{III} cannot issue the secret value extract query and public key replace query with ID^*, and the decryption query with (ID^*, i^*, C^*).

Guess. A_{III} will make a guess γ' for γ. The advantage ϵ_{III} of an IND-ID-CCA adversary A_{III} to attack the proposed revocable certificateless encryption scheme is evaluated by $Adv_{A_{III}} = \left|\Pr\left[\gamma' = \gamma\right] - \frac{1}{2}\right|$. If the adversary A_{III} who breaks the proposed scheme with a non-negligible advantage ϵ_{III}, then the challenger B_{III} can solve the BDHP problem with a non-negligible advantage ϵ'_{III}.

The probability that B_{III} does not abort during the simulation is analyzed as follows. In Phase 1 and 2, if $coin = 1$, the simulation failures and aborts since challenger B_{III} cannot answer the correct secret value extract query. Otherwise, the simulation continues. Let δ denotes the probability that $coin = 0$. Since the adversary A_I makes at most q_s and q_d queries to secret value extract query and decryption queries in Phase 1 and 2, respectively, the probability that the simulation does not abort is δ^{q_s}. In the challenge phase, if $coin = 1$, the simulation continues, which means the probability that the simulation does not abort is $1 - \delta$. Thus, the total probability that the simulation does not abort is $\delta^{q_u} \cdot (1 - \delta)$ in Game 3. By using the similar technique to Coron's analysis of the full domain hash signature scheme [44], the value is maximized at $\delta = 1 - 1/(q_s + 1)$ and the probability that B_{III} does not abort is at least $1/e(1 + q_s)$. Furthermore, the probability to guess the correct answer D in the real attack is at least $2\epsilon_{III}/q_3$ [1]. To respond to the decryption query, B_{III} scans the list $\mathcal{L}_4$ to obtain M. Since the simulation would success if $\langle U, V, -, ID, i, W\rangle$ appears in list $\mathcal{L}_4$ and there are at most q_d decryption queries, the probability the simulation aborts is q_d/q. In summary, B_I can solve the BDH problem with a non-negligible advantage $\epsilon'_{III} = 2\epsilon_{III}/e(1 + q_s)\, q_3 - q_d/q$.

Lemma 4: In the random oracle model, assume that an adversary A_{IV} who obtains at most $t - 1$ time update key shares from $t - 1$ greedy s-CRAs among a total of n s-CRAs in attacking the proposed revocable certificateless encryption in the sense of RCL-IND-CCA security for Type-IV adversary. Then, there exists a PPT adversary A_I can attack the proposed scheme and hence solves the BDH problem with a non-negligible probability.

Proof: First, we assume that there are no more than $t - 1$ greedy s-CRAs in the proposed (t, n) system. According to the construction of Shamir's [45] (t, n) threshold scheme, the invalid user who obtains at most $t - 1$ time update key shares $T_{ID,i,j}$ cannot interpolate the coefficients of the secret $t - 1$ degree polynomial $f(x)$ generated by the PKG and thus cannot obtain the right time update key $T_{ID,i}$. Therefore, in this situation, the adversary A_{IV} is identical to adversary A_I in the proof of lemma 1.

Theorem 1: In the random oracle model, the proposed RCL-PKE with outsourced s-CRA is semantically secure against adaptive chosen-ciphertext attack (RCL-IND-CCA) under the BDH assumption.

Proof: By Lemmas 1, 2, 3 and 4, we can conclude the theorem.

TABLE 1. Notations of computational and communication costs.

Notation	Description
$TG_{\hat{e}}$	Time cost of a bilinear pairing map
T_m	Time cost of a scalar multiplication in $\mathbb{G}$
T_{exp}	Time cost of a modular exponentiation in $\mathbb{G}_T$
T_h	Time cost of a map-to-point hash function
$\|C\|$	The bit length of a parameter C

TABLE 2. Computational time for related operations.

Operation	Time (ms)
$TG_{\hat{e}}$	3.27
T_m	1.64
T_{exp}	0.25
T_h	2.66

VI. PERFORMANCE EVALUATION AND COMPARISONS

In this section, we present the performance evaluation of the proposed scheme. Previous implementations [12], [13], [18], [27], [30] have shown that the non-negligible related computation costs include bilinear pairing map, scalar multiplication, map-to-point hash and modular exponentiation operations. Table 1 lists the notations used to describe the computational costs of the related operations.

We evaluate the costs of the above operations using Pairing-Based Cryptography (PBC) library [46] on an Inter Core-i7 computer. We choose the PBC built-in type- A pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ on the elliptic curve over a finite field $E(F_p)$, where $\mathbb{G}, \mathbb{G}_T$ are groups of prime order q. For the considerations of security and efficiency, p and q are large prime numbers of 512 bits and 160 bits. The operating system of the computer is CentOS 7.0 for 64-bit with Intel(R) Core (TM) i7-4710MQ CPU @ 2.50GHz and 8GB RAM. The average computational times for related operations are measured one thousand times and listed in Table 2.

In Table 3, we demonstrate the comparisons between the proposed RCL-PKE with the outsourced semi-trusted CRA scheme and the previously proposed RCL-PKE schemes [13], [14], [18] in terms of the size of each private key, computational and communication costs, revocation flexibility and other criteria, where $l = 256$ bits is the bit-length of plaintexts, n is the number of s-CRAs, and t is the trust threshold.

For the size of each private key, $7|\mathbb{z}_q| + 4|\mathbb{G}|$, $4|\mathbb{G}|$, and $2|\mathbb{G}| + |\mathbb{z}_q|$ are required for Shen *et al.'s* scheme, Tsai *et al.'s* scheme, and Tsai *et al.'s* RCL-PKE with the DRA scheme, respectively, while $2|\mathbb{G}|$ is required for our scheme. Since Shen *et al.'s* scheme and Tsai *et al.'s* RCL-PKE with DRA scheme are based on the standard model, they require $7TG_{\hat{e}}+10T_{exp}$, $5T_{exp}$ to encrypt a message, and $3TG_{\hat{e}}+5T_{exp}$, $5TG_{\hat{e}}$ to decrypt it. Tsai *et al.'s* scheme and our scheme are based on random oracle model, $TG_{\hat{e}} + T_m + T_a + 2T_h + T_{exp}$ and $2TG_{\hat{e}}+T_m+2T_h+T_{exp}$ are required to encrypt a message, and $TG_{\hat{e}} + T_m + T_a$ and $TG_{\hat{e}} + T_a$ are required to decrypt it.

For the computational cost for key updates, both Shen *et al.'s* scheme and Tsai *et al.'s* RCL-PKE with DRA scheme require $6T_{exp}$. Indeed, Tsai *et al.'s* scheme requires the

TABLE 3. Scheme comparisons.

($|\mathbb{G}|$ =1024 bits, $|\mathbb{G}_T| = 1024$ bits, $|l|$=256 bits, n= 3, $t = 2$)

	Shen *et al.*'s Scheme [14]	Tsai *et al.*'s Scheme [13]	Tsai *et al.*'s RCL-PKE with DRA Scheme [18]	Our proposed RCL-PKE with s-CRA Scheme	
Size of each user's private key	$7\|\mathbb{z}_q\| + 4\|\mathbb{G}\|$	$2\|\mathbb{G}\| + \|\mathbb{z}_q\|$	$4\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	
Computational cost for encryption	$7TG_{\hat{e}} + 10T_{exp}$	$TG_{\hat{e}} + 2T_m + 2T_h + T_{exp}$	$5T_{exp}$	$2TG_{\hat{e}} + T_m + 2T_h + T_{exp}$	
	25.39 ms	12.12 ms	1.25 ms	13.75 ms (Theoretical) 15.66 ms (Actual)	
Computational cost for decryption	$3TG_{\hat{e}} + 5T_{exp}$	$TG_{\hat{e}} + 2T_m$	$5TG_{\hat{e}}$	$TG_{\hat{e}}$	
	11.06 ms	6.55 ms	16.35 ms	3.27 ms (Theoretical) 3.67 ms (Actual)	
Computational and communication costs for key update	$6T_{exp}$	$T_h + T_m$	$6T_{exp}$	$n(T_h + T_m)$ (n s-CRAs)	tT_m (Each User)
	1.50 ms	4.30 ms	1.50 ms	12.90 ms	3.28 ms
Bit length of ciphertext	$5\|\mathbb{G}\|$	$\|\mathbb{G}\| + 2\|l\|$	$4\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$\|\mathbb{G}\| + 2\|l\|$	
	5120 bits	1536 bits	5120 bits	1536 bits	
Revocation flexibility	NO	NO	YES	YES	
Public key uniqueness	NO	NO	NO	YES	
Security model	Standard model	Random oracle model	Standard model	Random oracle model	

minimum $T_h + T_m$ time consumption to update a user, while in our proposed scheme, n s-CRAs require $n(T_h + T_m)$ in total and a user needs tT_m to compose its time update key. Note that the total computational and communication costs of n s-CRAs can be reduced by optimizing their deployment pattern.

Furthermore, we evaluated our encryption and decryption operations on the test platform to determine the actual performance of the proposed scheme. The average computational times for encryption and decryption are measured one thousand times and are close to the theoretical value.

Note that our scheme provides enhanced revocation flexibility compared to other schemes. The DRA in Tsai *et al*. [18]'s scheme is fully honest and trusted to execute instructions from the PKG. In our proposed scheme, s-CRA cannot provide time update key alone and stealthily, which ensures that our scheme is effective against the greedy s-CRA attacker. Meanwhile, rather than taking user identity ID as the input of the initial secret key extract in [13], [14], [18], in the proposed scheme, we take user identity ID along with public key P_{ID} as the input of the partial identity key extract. This modification guarantees the PKG only generates partial identity key with the specified public key. Thus, by binding the user's initial secret key to its public key, our scheme guarantees the public key uniqueness, while the other schemes do not.

VII. CONCLUSION

In this paper, we propose a secure RCL-PKE scheme with outsourced semi-trusted cloud revocation agent based on bilinear pairings. We present the framework and formalize the security model. Under the BDH assumption, we have demonstrated that the proposed scheme is CCA secure against the four kinds of adversaries in the random oracle model. In our proposed scheme, s-CRA cannot provide time update key alone and stealthily, which ensures that our scheme is effective against the greedy s-CRA attacker. By composing enough pieces of time update key share from semi-trusted s-CRAs, legal users can decrypt ciphertext with small computational cost. Meanwhile, our scheme guarantees the public key uniqueness to prohibit the multiple copying of each initial secret key.

REFERENCES

[1] D. Boneh and M. K. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2001, pp. 213–229.

[2] D. Boneh, X. Ding, G. Tsudik, and C. M. Wong, "A method for fast revocation of public key certificates and security capabilities," in *Proc. USENIX Secur. Symp.*, 2001, p. 22.

[3] B. Libert and J. J. Quisquater, "Efficient revocation and threshold pairing based cryptosystems," in *Proc. 22nd Annu. Symp. Principles Distrib. Comput.*, 2003, pp. 163–171.

[4] H. S. Ju, D. Y. Kim, D. H. Lee, J. Lim, and K. Chun, "Efficient revocation of security capability in certificateless public key cryptography," in *Proc. Int. Conf. Knowl.-Based Intell. Inf. Eng. Syst.* Berlin, Germany: Springer, 2005, pp. 453–459.

[5] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. 15th ACM Conf. Comput. Commun. Secur. (CCS)*, 2008, pp. 417–426.

[6] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*. Berlin, Germany: Springer, 2005, pp. 457–473.

[7] W. Aiello, S. Lodha, and R. Ostrovsky, "Fast digital identity revocation," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 1998, pp. 137–152.

[8] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 2139. Berlin, Germany: Springer, 2001, pp. 41–62.

[9] B. Libert and D. Vergnaud, "Adaptive-ID secure revocable identity-based encryption," in *Proc. Cryptographers' Track RSA Conf.* Berlin, Germany: Springer, 2009, pp. 1–15.

[10] J. H. Seo and K. Emura, "Efficient delegation of key generation and revocation functionalities in identity-based encryption," in *Proc. Cryptographers' Track RSA Conf.* Berlin, Germany: Springer, 2013, pp. 343–358.

[11] J. H. Seo and K. Emura, "Revocable hierarchical identity-based encryption: History-free update, security against insiders, and short ciphertexts," in *Proc. Cryptographers' Track RSA Conf.* Cham, Switzerland: Springer, 2015, pp. 106–123.

[12] Y.-M. Tseng and T.-T. Tsai, "Efficient revocable ID-based encryption with a public channel," *Comput. J.*, vol. 55, no. 4, pp. 475–486, Apr. 2012.

[13] T.-T. Tsai and Y.-M. Tseng, "Revocable certificateless public key encryption," *IEEE Syst. J.*, vol. 9, no. 3, pp. 824–833, Sep. 2015.

[14] L. Shen, F. Zhang, and Y. Sun, "Efficient revocable certificateless encryption secure in the standard model," *Comput. J.*, vol. 57, no. 4, pp. 592–601, Apr. 2014.

[15] Y.-K. Tang, S. S. M. Chow, and J. K. Liu, "Comments on 'efficient revocable certificateless encryption secure in the standard model,'" *Comput. J.*, vol. 58, no. 4, pp. 779–781, Apr. 2015.

[16] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 425–437, Feb. 2015.

[17] Y.-M. Tseng, T.-T. Tsai, S.-S. Huang, and C.-P. Huang, "Identity-based encryption with cloud revocation authority and its applications," *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 1041–1053, Oct. 2018.

[18] T.-T. Tsai, Y.-M. Tseng, and S.-S. Huang, "Efficient revocable certificateless public key encryption with a delegated revocation authority," *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 3713–3725, Dec. 2015.

[19] S. Micali, "Enhanced certificate revocation," Lab. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Memo MIT/LCS/TM-542 b, Mar. 1996, pp. 1–10.

[20] S. M. Novomodo, "Scalable certificate validation and simplified PKI management," in *Proc. PKI Res. Workshop*, vol. 15, 2002, pp. 1–23.

[21] M. Naor and K. Nissim, "Certificate revocation and certificate update," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 4, pp. 561–570, Apr. 2000.

[22] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2003, pp. 272–293.

[23] V. Goyal, "Certificate revocation using fine grained certificate space partitioning," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2007, pp. 247–259.

[24] F. F. Elwailly, C. Gentry, and Z. Ramzan, "QuasiModo: Efficient certificate validation and revocation," in *Proc. Int. Workshop Publcic Key Cryptogr.* Berlin, Germany: Springer, 2004, pp. 375–388.

[25] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1984, pp. 47–53.

[26] O. Blazy, P. Germouty, and D. H. Phan, "Downgradable identity-based encryption and applications," in *Proc. Cryptographers' Track RSA Conf.* Cham, Switzerland: Springer, 2019, pp. 44–61.

[27] X. Jia, D. He, S. Zeadally, and L. Li, "Efficient revocable ID-based signature with cloud revocation server," *IEEE Access*, vol. 5, pp. 2945–2954, 2017.

[28] M. Girault, "Self-certified public keys," in *Advances in Cryptology—EUROCRYPT'91* (Lecture Notes in Computer Science), vol. 547. Berlin, Germany: Springer, 1992, pp. 490–497.

[29] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 2894. Berlin, Germany: Springer, 2003, pp. 452–473.

[30] M. H. Au, Y. Mu, J. Chen, D. S. Wong, J. K. Liu, and G. Yang, "Malicious KGC attacks in certificateless cryptography," in *Proc. 2nd ACM Symp. Inf., Comput. Commun. Secur.*, 2007, pp. 302–311.

[31] A. W. Dent, B. Libert, and K. G. Paterson, "Certificateless encryption schemes strongly secure in the standard model," in *Proc. Int. Workshop Public Key Cryptogr.*, Berlin, Germany: Springer, 2008, pp. 344–359.
[32] B. Libert and J. J. Quisquater, "On constructing certificateless cryptosystems from identity based encryption," in *Proc. Int. Workshop Public Key Cryptogr.* Berlin, Germany: Springer, 2006, pp. 474–490.
[33] D. H. Yum and P. J. Lee, "Generic construction of certificateless signature," in *Proc. Australas. Conf. Inf. Secur. Privacy*, Berlin, Germany: Springer, 2004, pp. 200–211.
[34] G. Zhang and X. Wang, "Certificateless encryption scheme secure in standard model," *Tsinghua Sci. Technol.*, vol. 14, no. 4, pp. 452–459, Aug. 2009.
[35] X. Huang, Y. Mu, W. Susilo, D. S. Wong, and W. Wu, "Certificateless signatures: New schemes and security models," *Comput. J.*, vol. 55, no. 4, pp. 457–474, Apr. 2012.
[36] R. Gao, J. Zeng, and L. Deng, "An efficient certificateless multi-receiver threshold decryption scheme," *RAIRO-Theor. Informat. Appl.*, vol. 53, nos. 1–2, pp. 67–84, 2019.
[37] J.-D. Wu, Y.-M. Tseng, S.-S. Huang, and W.-C. Chou, "Leakage-resilient certificateless key encapsulation scheme," *Informatica*, vol. 29, no. 1, pp. 125–155, Jan. 2018.
[38] Y. Sun, F. Zhang, and A. Fu, "Revocable Certificateless Encryption with Ciphertext Evolution," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Cham, Switzerland: Springer, 2018, pp. 741–749.
[39] Y.-M. Tseng, S.-S. Huang, and J.-D. Wu, "Secure certificateless signature resisting to continual leakage attacks," in *Proc. Int. Conf. Appl. Syst. Innov. (ICASI)*, May 2017, pp. 1263–1266.
[40] Y. Lu and J. Li, "Provably secure certificateless proxy signature scheme in the standard model," *Theor. Comput. Sci.*, vol. 639, pp. 42–59, Aug. 2016.
[41] Y. Lu, J. Li, and Y. Zhang, "Privacy-preserving and pairing-free multirecipient certificateless encryption with keyword search for cloud-assisted IIoT," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2553–2562, Apr. 2020, doi: 10.1109/JIOT.2019.2943379.
[42] Y. Sun, Y. Mu, W. Susilo, F. Zhang, and A. Fu, "Revocable identity-based encryption with server-aided ciphertext evolution," *Theor. Comput. Sci.*, vol. 815, pp. 11–24, May 2020.
[43] B. Qin, X. Liu, Z. Wei, and D. Zheng, "Space efficient revocable IBE for mobile devices in cloud computing," *Sci. China Inf. Sci.*, vol. 63, no. 3, Mar. 2020, Art. no. 139110.
[44] J. S. Coron, "On the exact security of full domain hash," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 1880. Berlin, Germany: Springer, 2000, pp. 229–235.
[45] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[46] B. Lynn. (2019). *Pairing-Based Cryptography (PBC) Library*. [Online]. Available: https://crypto.stanford.edu/pbc