An Efficient Revocable Identity-Based Encryption with Equality Test Scheme for the Wireless Body Area Network

Tung-Tso Tsai, Han-Yu Lin, and Hsiao-Chieh Chang

Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung 202, Taiwan

Correspondence should be addressed to Tung-Tso Tsai; tttsai@mail.ntou.edu.tw

Received 16 June 2022; Revised 18 July 2022; Accepted 1 August 2022; Published 13 August 2022

Copyright © 2022 Tung-Tso Tsai 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.

With the rapid development and popularization of cloud computing, people are willing to upload their own data to the cloud to enjoy the services. However, some personal and private data are not suitable for uploading directly to the cloud. Therefore, these data must be encrypted before uploading to the cloud to ensure the confidentiality. To achieve the confidentiality of data and enjoy cloud services, a notion of identity-based encryption with equality test (IBEET) was proposed. Using IBEET, two ciphertexts encrypted under different public keys can be tested to confirm whether they contain the same plaintext. The equality test can be applied to the wireless body area network system in which the cloud can utilize ciphertexts from patients and medical institutions to perform equality tests to determine whether the patient’s status is abnormal. Indeed, revoking illegal or expired users on any cryptosystem is an important issue. To the best of our knowledge, there is little research on the design mechanism of user revocation in the IBEET. In this paper, we propose a novel notion of revocable identity-based encryption with an equality test, called RIBEET. Based on the notion, we present the first RIBEET scheme. Meanwhile, the proposed scheme will be proven to be secure under the bilinear Diffie-Hellman (BDH) assumption.

1. Introduction

With the rapid development and popularization of cloud computing, people are willing to upload their own data to the cloud to enjoy the services. However, some personal and private data are not suitable for uploading directly to the cloud. To ensure the confidentiality of data, several encryption mechanisms [1–4] have been applied to cloud computing. Identity-based encryption (IBE) [5] is one of the encryption mechanisms of public key systems. The system of an IBE contains two roles: the private key generator (PKG) and users (including senders and receivers). Each user utilizes his own identity (e.g., e-mail address, name, or social security number) to register with the PKG to obtain a private key. Senders can regard the identity of the receiver as a public key to encrypt private data. After receiving the encrypted message (ciphertext), the receiver can decrypt it with her/his own private key.

To achieve the confidentiality of data and enjoy cloud services, the first identity-based encryption with equality test (IBEET) was proposed by Ma [6]. Using IBEET, two ciphertexts encrypted under different public keys can be tested to confirm whether they contain the same plaintext. Ma [6] also gave an application of IBEET used to classify encrypted e-mails. Each encrypted e-mail can be attached with a tag for classification, while the tag can be encrypted under different public keys in the IBEET system. An e-mail server in the cloud can test the equality of any two encrypted tags to classify encrypted e-mails. Subsequently, many studies on IBEET have been published in the literature [7–11].

The equality test can be applied to the wireless body area network (WBAN) system [12–17] in which the cloud can utilize ciphertexts from patients and medical institutions to perform equality tests to determine whether the patient’s status is abnormal. Figure 1 shows the architecture of WBANs. A patient is equipped with wearable sensors to collect her/his health record data from sensors of electroencephalogram (EEG), electrocardiogram (ECG), blood pressure, pulse oximeter, insulin pump, electromyogram (EMG), and motion. These health record data are encrypted through the mobile device and uploaded to the cloud server. On the other hand, the medical institution also uploads the
patient’s encrypted health data to the cloud server. The ciphertexts can be tested for equality without knowing the health data of the patient by the cloud server. If the patient’s health data are different from the medical institution’s health data, it means that the patient’s health data are abnormal.

Indeed, revoking illegal or expired users on any cryptosystem is an important issue. In the traditional public key cryptosystem (PKC), public key infrastructures (PKI) must be established to manage each user’s certificate which links the user’s identity and public key. In addition, the certificate revocation list [18] is also included in the PKI to revoke illegal or expired users. In identity-based public key cryptosystems (ID-PKC), the first IBE was presented by Boneh and Franklin [5] in which a user can be revoked by the PKG, who sends new private keys for all nonrevoked users at each period, if the user did not receive the new private key. So far, many literatures related to revocable IBE [19–26] have been published. To the best of our knowledge, there is little research on design mechanism of user revocation in the IBEET. In this paper, we propose a novel notion of revocable identity-based encryption with equality test, called RIBEET. Based on the notion, we present the first RIBEET scheme. Meanwhile, the scheme will be proven to be secure under the bilinear Diffie-Hellman (BDH) assumption.

1.1. Related Work. In the era of advanced network communication, cloud computing is an indispensable part. The terminal devices on the user side usually do not have high-performance computing power. However, users can entrust large computing tasks to the cloud. Then, the cloud will return the corresponding results to users after finishing the tasks. Indeed, the cloud can assist each user in performing tasks that require a lot of computation, but it also means that the cloud can know each user’s data if the data is not encrypted. Typically, users will encrypt data to the cloud if the data is sensitive or private. In addition, encrypted data also needs to be quickly retrieved from the cloud. To achieve this function, several schemes [3, 4, 27, 28] related to public key encryption with a keyword search were proposed. Although these schemes can retrieve encrypted data, only data encrypted under the same public key can be retrieved.

To support searchable encrypted data under different public keys, Yang et al. [29] proposed a comparison mechanism of two ciphertexts encrypted under different public keys in the traditional public key cryptosystem, called public key encryption with equality test (PKEET). However, the traditional public key cryptosystem must rely on the public key infrastructure to manage each user’s certificate which links the user’s identity and her/his public key. To avoid the use of public key infrastructure and certificates, Shamir [30] introduced a new concept of ID-PKC in which a user’s public key is her/his identity such as name, e-mail, or telephone number. In this way, certificates will no longer be needed in the ID-PKC since the public key is meaningful and can represent the user’s identity. Combining the concepts of PKEET and ID-PKC, Ma [6] proposed the first identity-based encryption with equality test, called IBEET. To consider more types of authorizations, Li et al. [31] proposed the IBEET scheme with four types of authorizations. Unfortunately, the proposed scheme of Li et al. [31] is not suitable for the IoT environment because the performance of the scheme is not good. Immediately, Elhabob et al. [10] proposed another IBEET scheme with four types of authorizations which has higher performance.

For the issue of user revocation in the ID-PKC, Boneh and Franklin [5] suggested that the new private keys should be resent to users who have not been revoked at different periods. As a result, secure channels will be established to send these private keys, and the PKG’s workload will also increase. To reduce the PKG’s workload, Boldyreva et al. [19] hired a binary tree to propose an IBE scheme with user revocation, named revocable IBE (RIIBE). However, Boldyreva et al.’s scheme [19] only satisfied the selective-ID security. Later, Libert and Vergnaud [20] proposed another
Table 1: Comparisons between the existing schemes and our RIBEET scheme.

| Schemes   | Public key setting | Avoiding the use of certificates | Supporting equality test of ciphertexts | Providing user revocation |
|-----------|--------------------|----------------------------------|----------------------------------------|--------------------------|
| PKET [29] | PKI-based          | No                               | Yes                                    | Yes                      |
| IBE [5]   | ID-based           | Yes                              | No                                     | No                       |
| RIBE [21] | ID-based           | Yes                              | No                                     | Yes                      |
| IBEET [6] | ID-based           | Yes                              | Yes                                    | No                       |
| Our RIBEET| ID-based           | Yes                              | Yes                                    | Yes                      |

RIBE scheme which meets the adaptive-ID security. A mechanism for revoking users through public channels was proposed by Tseng and Tsai [21], in which each user’s full private key is divided into two parts: a fixed key and a time updated key. The fixed key is delivered to the user through secure channels only once, while the time updated key is delivered to the user through public channels at different periods. Users can be revoked if they do not receive the new time updated keys. For the security of decryption key exposure, Seo and Emura [22] proposed a new RIBE scheme to enhance the security. To reduce the length of public parameters and meet the security of decryption key exposure resistance, Watanabe et al. [23] presented another RIBE scheme. In addition, several lattice-based RIBE schemes [24–26] were proposed to resist quantum attacks.

1.2. Motivation. As mentioned earlier, revoking illegal or expired users on any cryptosystem is still an important issue. In the traditional PKC, the PKET [29] can hire the certificate revocation list [18] to revoke illegal or expired users. However, the IBE [5] cannot effectively revoke illegal or expired users in the ID-PKC, so the RIBE [21] was proposed. To the best of our knowledge, there is little research on the design mechanism of user revocation in the IBEET [6]. Table 1 shows the comparisons between the PKET [29], the IBE [5], the RIBE [21], the IBEET [6], and our RIBEET in terms of public key setting, avoiding the use of certificates, supporting the equality test of ciphertexts, and providing user revocation. Hence, we attempt to propose the first revocable identity-based encryption with equality test, called RIBEET. In the following, we list specific contributions.

(i) Based on the existing syntax and security notions of IBEET, we consider the property of user revocation to define a new syntax and security notions of RIBEET.

(ii) Following the syntax of RIBEET, a concrete RIBEET scheme is proposed.

(iii) In the security notions of RIBEET, the proposed scheme is proven to be secure under the bilinear Diffie-Hellman (BDH) assumption.

(iv) We compare the proposed scheme with the previous RIBE scheme and IBEET scheme. We demonstrate that the proposed scheme not only provides user revocation but also supports the equality test for ciphertexts.

The rest of the article includes six sections. Preliminaries are given in Section 2. Section 3 presents the syntax and security notions of RIBEET. A concrete RIBEET scheme is proposed in Section 4. The security analysis of the RIBEET scheme is shown in Section 5. We compare the RIBEET scheme with other existing schemes in Section 6. The last section gives the conclusion.

2. Preliminaries

In this section, we introduce two definitions related to a mathematical tool and security assumption. We hire the bilinear pairings [5] as a mathematical tool to construct our RIBEET scheme. To prove the security of the proposed scheme, we consider the bilinear Diffie-Hellman (BDH) problem and then give a BDH assumption [6]. The definition of the bilinear pairings is given as follows.

**Definition 1.** Let \( G_1, G_2, \) and \( G_T \) be three multiplicative cyclic groups of a prime order \( q \). Assume that a mapping \( \tilde{e} : G_1 \times G_2 \rightarrow G_T \) is an asymmetric bilinear map. Then, the map \( \tilde{e} \) satisfies the following properties.

1. **Bilinearity:** \( \tilde{e}(g_1^a, g_2^b) = \tilde{e}(g_1, g_2)^{ab} \) for \( g_1 \in G_1, g_2 \in G_2, \) and \( a, b \in \mathbb{Z}_q^* \).

2. **Nondegeneracy:** \( \tilde{e}(g_1, g_2) \neq 1 \) for some \( g_1 \in G_1 \) and \( g_2 \in G_2 \).

3. **Computability:** \( \tilde{e}(g_1, g_2) \) can be efficiently computed for \( g_1 \in G_1, g_2 \in G_2 \).

For the asymmetric bilinear map, the BDH problem is to compute \( \tilde{e}(g_1, g_2)^{abc} \) by given a tuple \( \langle q, G_1, G_2, G_T, \tilde{e}, g_1, g_1^a, g_1^b, g_2^a, g_2^b \rangle \). We define the BDH assumption as follows.
Figure 2: The syntax of RIBEET.

Figure 3: The procedure for user revocation.
Table 2: Notations.

| Notations | Meaning |
|-----------|---------|
| PSP       | The public system parameters |
| SLT       | The system lifetime |
| mpk       | The master private key |
| ID        | The user’s identity |
| IKID      | The user initial key |
| TKID      | The user time key |
| TDID      | The ciphertext trapdoor |
| CT        | The ciphertext-trapdoor pair of the user $U_i$ |
| TD        | The ciphertext-trapdoor pair of the user $U_j$ |

Definition 2. On inputting a tuple $(q, G_1, G_2, G_1, \tilde{e}, g_1, g_1^t, g_2, g_2^t, g_2^h)$, we say that the BDH problem holds if no algorithm $\mathcal{A}$ has nonnegligible advantage in computing $\tilde{e}(g_1, g_2)^{abc}$. The advantage can be denoted as $\text{Pr}[\mathcal{A}(g_1, g_1^t, g_1^t, g_2, g_2^t, g_2^h) = \tilde{e}(g_1, g_2)^{abc}] < \epsilon$.

3. Syntax and Security Notions

3.1. Syntax of RIBEET. Based on the syntax of IBEET schemes [6], we employ the revocation technique [21] to present a new syntax of RIBEET depicted in Figure 2 which consists of three roles and seven algorithms, namely Setup, InitialKey, TimeKey, Encryption, Decryption, Trapdoor, and Test. The first role is the private key generator (PKG) who is responsible for executing the first three algorithms, and the second role is the users who can, respectively, utilize Encryption, Decryption, and Trapdoor algorithms for encryption, decryption, and authorization. The last role is the cloud server (CS) who runs the Test algorithm to compare the two ciphertexts. For the user revocation, we use Figure 3 to illustrate how users are revoked by the PKG. If the PKG stops sending the time key to a user, it means that the user has been revoked since both initial key and time key are required to execute Decryption and Trapdoor algorithms. Here, we arrange some notations used in these algorithms in Table 2. The algorithms of RIBEET are described in detail as follows.

(i) Setup: this algorithm is performed by the PKG who takes a security parameter $k$ and a time period $t$ as input to produce the public system parameters PSP, the system lifetime SLT, and the master private key mpk

(ii) InitialKey: this algorithm is performed by the PKG who takes the public system parameter PSP, the master private key mpk, and a user’s identity ID $\in \{0, 1\}^*$ as input to produce user initial key IKID

(iii) TimeKey: this algorithm is performed by the PKG who takes the public system parameter PSP, the master private key mpk, a user’s identity ID $\in \{0, 1\}^*$, and a period $T \in \text{SLT}$ as input to produce user time key TKID

(iv) Encryption: this algorithm is performed by a user (sender) who takes the public system parameter PSP, a user’s identity ID $\in \{0, 1\}^*$, a period $T \in \text{SLT}$, and a message $M \in \{0, 1\}^*$ as input to produce a ciphertext CT

(v) Decryption: this algorithm is performed by a user (receiver) who takes the public system parameter PSP, the receiver’s initial key IKID, the receiver’s time key TKID, and the ciphertext CT as input to produce the message $M$

(vi) Trapdoor: this algorithm is performed by a user who takes her/his initial key IKID and time key TKID as input to produce the trapdoor TDID

(vii) Test: this algorithm is performed by the CS who takes the public system parameters PSP and two ciphertext-trapdoor pairs $(CT_1, TD_1)$ and $(CT_2, TD_2)$ from any two users $U_i$ and $U_j$ as input to produce 1 or 0

3.2. Security Notions of RIBEET. In this section, we define the security notions of RIBEET which includes four types of adversaries. Two of these types are the same as the security notions of IBEET [6]. Considering the revoked users from RIBEET, we need to add two types of adversaries in the security notions. These four types of adversaries are presented as follows.

1. Type I adversary: such an adversary can obtain all information (including time key TKID) transmitted through public channels. The adversary can be regarded as an outside attacker

2. Type II adversary: such an adversary owns her/his initial key IKID, but he does not have the current time key TKID. The adversary can be regarded as a revoked user

3. Type III adversary: this adversary is identical to the type I adversary, except that she/he possesses the trapdoor TD

4. Type IV adversary: this adversary is identical to the type II adversary, except that she/he possesses the trapdoor TD

Following the security notions of IBEET [6], we consider revoked users to define the new security notions of RIBEET. Definitions 3 and 4, respectively, are given to state IND-ID-CCA and OW-ID-CCA security of an RIBEET scheme.
Definition 3 (IND-ID-CCA). Let $\mathcal{A}$ be a type I or type II adversary for an RIBEET scheme and $\mathcal{B}$ be a challenger in the following game. The scheme is IND-ID-CCA secure if the advantage that $\mathcal{A}$ wins the game is negligible.

1. Setup. The challenger $\mathcal{B}$ takes a security parameter $k$ and a time period $t$ as input to produce the public system parameters $\text{PSP}$, the system life time $\text{SLT}$, and the master private key $\text{mpk}$. The public system parameters $\text{PSP}$ and the system life time $\text{SLT}$ are sent to the adversary $\mathcal{A}$

2. Phase 1. Several queries below can be issued by the adversary $\mathcal{A}$

(a) InitialKey query($\text{ID}$): given an identity $\text{ID}$, the challenger $\mathcal{B}$ generates an initial key $\text{IK}_{\text{ID}}$ as the response by running the InitialKey algorithm of the RIBEET scheme

(b) TimeKey query($\text{ID}, T$): given an identity $\text{ID}$ and a period $T$, the challenger $\mathcal{B}$ generates a time key $T \text{K}_{\text{ID}}$ as the response by running the TimeKey algorithm of the RIBEET scheme

(c) Decryption query($\text{ID}, T, \text{CT}$): given an identity $\text{ID}$, a period $T$, and a ciphertext $\text{CT}$, the challenger $\mathcal{B}$ generates the resulting message $M$ as the response by running the Decryption of the RIBEET scheme

(d) Trapdoor query($\text{ID}, T$): given an identity $\text{ID}$ and a period $T$, the challenger $\mathcal{B}$ generates a trapdoor $T \text{D}_{\text{ID}}$ as the response by running the Trapdoor of the RIBEET scheme

3. Challenge. Two messages $M_0^*, M_1^*$, an identity $\text{ID}^*$, and a period $T^*$ are submitted by the adversary $\mathcal{A}$. The challenger $\mathcal{B}$ chooses $M_0^*$ from these two messages, where $b \in \{0, 1\}$ is a random coin. The challenger $\mathcal{B}$ then generates a ciphertext $\text{CT}^*$ as the challenge one by running the Encryption of the RIBEET scheme with $(\text{ID}^*, T^*, M_0^*)$. Here, the following restrictions must be satisfied

(i) The adversary $\mathcal{A}$ cannot issue the Trapdoor query with $\text{ID}^*$

(ii) The adversary $\mathcal{A}$ cannot issue the InitialKey query with $\text{ID}^*$ if it is the type I adversary

(iii) The adversary $\mathcal{A}$ cannot issue the TimeKey query with $(\text{ID}^*, T^*)$ if it is the type II adversary

4. Phase 2. Under the above restrictions, $\mathcal{A}$ can execute the same tasks as in phase 1

5. Guess. The adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins the game if $b' = b$. The advantage that $\mathcal{A}$ wins the game can be denoted as $\text{Adv}_{\mathcal{A}}(k) = |\text{Pr}[b' = b] - 1/2|

Definition 4 (OW-ID-CCA). Let $\mathcal{A}$ be a type III or type IV adversary for an RIBEET scheme and $\mathcal{B}$ be a challenger in the following game. The scheme is OW-ID-CCA secure if the advantage that $\mathcal{A}$ wins the game is negligible.

1. Setup. The challenger $\mathcal{B}$ takes a security parameter $k$ and a time period $t$ as input to produce the public system parameters $\text{PSP}$, the system life time $\text{SLT}$, and the master private key $\text{mpk}$. The public system parameters $\text{PSP}$ and the system life time $\text{SLT}$ are sent to the adversary $\mathcal{A}$

2. Phase 1. Several queries below can be issued by the adversary $\mathcal{A}$

(a) InitialKey query($\text{ID}$): given an identity $\text{ID}$, the challenger $\mathcal{B}$ generates an initial key $\text{IK}_{\text{ID}}$ as the response by running the InitialKey algorithm of the RIBEET scheme

(b) TimeKey query($\text{ID}, T$): given an identity $\text{ID}$ and a period $T$, the challenger $\mathcal{B}$ generates a time key $T \text{K}_{\text{ID}}$ as the response by running the TimeKey algorithm of the RIBEET scheme

(c) Decryption query($\text{ID}, T, \text{CT}$): given an identity $\text{ID}$, a period $T$, and a ciphertext $\text{CT}$, the challenger $\mathcal{B}$ generates the resulting message $M$ as the response by running the Decryption of the RIBEET scheme

(d) Trapdoor query($\text{ID}, T$): given an identity $\text{ID}$ and a period $T$, the challenger $\mathcal{B}$ generates a trapdoor $T \text{D}_{\text{ID}}$ as the response by running the Trapdoor of the RIBEET scheme

3. Challenge. An identity $\text{ID}^*$ and a period $T^*$ are submitted by the adversary $\mathcal{A}$. The challenger $\mathcal{B}$ randomly chooses $M^*$ and then generates a ciphertext $\text{CT}^*$ as the challenge one by running the Encryption
of the RIBEET scheme with \((\text{ID}^*, T^*, M^*)\). Here, the following restrictions must be satisfied:

(a) The adversary \(\mathcal{A}\) cannot issue the InitialKey query with \(\text{ID}^*\) if it is the type III adversary

(b) The adversary \(\mathcal{A}\) cannot issue the TimeKey query with \((\text{ID}^*, T^*)\) if it is the type IV adversary

(4) Phase 2. Under the above restrictions, \(\mathcal{A}\) can execute the same tasks as in phase 1

(5) Guess. The adversary \(\mathcal{A}\) outputs a guess \(M'\) and wins the game if \(M^* = M'\). The advantage that \(\mathcal{A}\) wins the game can be denoted as \(\text{Adv}_{\mathcal{A}}(k) = \Pr[M^* = M']\).

4. Concrete RIBEET Scheme

A revocable identity-based encryption with equality test scheme, which we denote by RIBEET, consists of algorithms Setup, InitialKey, TimeKey, Encryption, Decryption, Trapdoor, and Test. Each of the algorithms is described as follows.

(1) Setup: this algorithm is performed by the PKG who takes a security parameter \(k\) and a time period \(t\) as input to produce an asymmetric bilinear map \(\hat{e} : G_1 \times G_2 \rightarrow G_T\) and a system life time \(\text{SLT} = \{T_0, T_1, \cdots, T_{l}\}\), where \(G_1, G_2,\) and \(G_T\) are multiplicative cyclic groups of prime order \(q\). The PKG first chooses two arbitrary generators \(g_1 \in G_1\) and \(g_2 \in G_2\) and picks eight cryptographic one-way hash functions \(H_1 : \{0,1\}^* \rightarrow G_2\), \(H_2 : \{0,1\}^* \rightarrow G_2\), \(H_3 : \{0,1\}^* \rightarrow G_2\), \(H_4 : \{0,1\}^* \rightarrow G_2\), \(H_5 : G_T \times G_1 \times G_1 \rightarrow \{0,1\}^{|G_1|}\), \(H_6 : \{0,1\}^L \rightarrow G_2\), \(H_7 : \{0,1\}^L \rightarrow Z_q^*\), and \(H_8 : G_T \rightarrow G_2\), where \(L\) and \(l\) are fixed lengths. Then, a random value \(s \in Z_q^*\) is chosen, and \(P_{\text{pub}} = g_1^s\) is computed. The public system parameters are \(\text{PSP} = \{q, G_1, G_2, G_T, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8\}\).

### Table 3: Comparisons of the proposed RIBEET with the existing RIBE and several IBEET.

| Schemes     | The cost of performing encryption | The cost of performing decryption | The cost of performing equality test | Supporting equality test of ciphertexts | Providing user revocation |
|-------------|-----------------------------------|-----------------------------------|-------------------------------------|----------------------------------------|--------------------------|
| RIBE [21]   | 1 · Pair + 2 · Exp               | 1 · Pair                          | —                                   | No                                     | Yes                      |
|             | (8.7843 ms)                      | (7.8351 ms)                       |                                     |                                        |                          |
| IBEET [6]   | 2 · Pair + 6 · Exp               | 2 · Pair + 2 · Exp                | 4 · Pair                            | Yes                                    | No                       |
|             | (18.5178 ms)                     | (16.6194 ms)                      | (31.3404 ms)                        |                                        |                          |
| IBEET [10]  | 2 · Pair + 4 · Exp               | 2 · Pair + 1 · Exp                | 2 · Pair + 2 · Exp                  | Yes                                    | No                       |
|             | (17.5686 ms)                     | (16.1448 ms)                      | (16.6194 ms)                        |                                        |                          |
| IBEET [11]  | 2 · Pair + 9 · Exp               | 2 · Pair + 2 · Exp                | 2 · Pair + 2 · Exp                  | Yes                                    | No                       |
|             | (19.9416 ms)                     | (16.6194 ms)                      | (16.6194 ms)                        |                                        |                          |
| Our RIBEET  | 2 · Pair + 5 · Exp               | 2 · Pair + 2 · Exp                | 4 · Pair                            | Yes                                    | Yes                      |
|             | (18.0432 ms)                     | (16.6194 ms)                      | (31.3404 ms)                        |                                        |                          |

### Table 4: Comparison of communication cost.

| Schemes     | | | |
|-------------|---------|---------|---------|
| RIBE [21]   | 2\(|G_1|\) | \(|G_1| + |Z_q|\) | —       |
| IBEET [6]   | \(|G_1|\) | 4\(|G_1| + |Z_q|\) | \(|G_1|\) |
| IBEET [10]  | \(|G_1|\) | 2\(|G_1| + 2|Z_q|\) | \(|G_1|\) |
| IBEET [11]  | \(|G_1| + |Z_q|\) | 3\(|G_1| + |Z_q|\) | 4\(|G_1|\) |
| Our RIBEET  | 2\(|G_2|\) | 3\(|G_1| + |Z_q|\) | \(|G_2|\) |
, G2, GT, ̃ε, g1, g2, P_pub, H1, H2, H3, H4, H5, H6, H7, H8 \}, the system life time is SLT = \{T_0, T_1, \cdots, T_r\}, and the master private key is mpk = s

(2) InitialKey: this algorithm is performed by the PKG who takes the public system parameter PSP, the master private key mpk, and a user’s identity ID ∈ \{0, 1\}^* as input to produce user initial key

\[ \text{IK}_{ID} = (\text{IK}_{ID1}, \text{IK}_{ID2}) = \left( H_1(ID)^{mpk}, H_2(ID)^{mpk} \right) \]  

(1)

\[ = (H_1(ID)^s, H_2(ID)^s). \]  

Here, the procedure of this algorithm is depicted in Figure 4.

(3) TimeKey: this algorithm is performed by the PKG who takes the public system parameter PSP, the master private key mpk, a user’s identity ID ∈ \{0, 1\}^*, and a period T ∈ SLT as input to produce user time key

\[ \text{TK}_{ID} = (\text{TK}_{ID1}, \text{TK}_{ID2}) = \left( H_3(ID, T)^{mpk}, H_4(ID, T)^{mpk} \right) \]  

(2)

\[ = (H_3(ID, T)^s, H_4(ID, T)^s). \]  

Here, the procedure of this algorithm is depicted in Figure 5.

(4) Encryption: this algorithm is performed by a sender who takes the public system parameter PSP, a user’s identity ID ∈ \{0, 1\}^*, a period T ∈ SLT, and a message M ∈ \{0, 1\}^k as input to produce ciphertexts CT = (CT_1, CT_2, CT_3, CT_4) which are shown as follows

(a) CT_1 = g_1^t
(b) CT_2 = g_1^u
(c) CT_3 = H_5(\(\tilde{e}(P_{pub}, H_1(ID) \cdot H_3(ID, T))^u\), CT_1, CT_2) \oplus (M\|V)
(d) CT_4 = H_6(M') \cdot H_6(\(\tilde{e}(P_{pub}, H_2(ID) \cdot H_4(ID, T))^u\))

Here, r = H_7(M, V) and the two values V ∈ \{0, 1\}^l and u ∈ \mathbb{Z}^*_q are chosen in random.

(5) Decryption: this algorithm is performed by a receiver who takes the public system parameter PSP, the receiver’s initial key IK_{ID} = (IK_{ID1}, IK_{ID2}), the receiver’s time key TK_{ID} = (TK_{ID1}, TK_{ID2}), and the ciphertext CT = (CT_1, CT_2, CT_3, CT_4) as input to produce the message M. The detailed process is shown as follows:

(a) Compute CT_3 \oplus H_5(\(\tilde{e}(CT_2, IK_{ID1} \cdot TK_{ID1}), CT_1, CT_2 \)) to obtain M'\|V'
(b) Compute r' = H_7(M', V')
(c) Produce the message M' as M if CT_1 = g_1^t and CT_4 = H_6(M') \cdot H_6(\(\tilde{e}(CT_2, IK_{ID2} \cdot TK_{ID2})\)) both hold

The correctness of obtaining M’\|V’ can be demonstrated as follows.

\[ CT_3 \oplus H_5(\(\tilde{e}(CT_2, IK_{ID2} \cdot TK_{ID2}), CT_1, CT_2 \)) = H_5(\(\tilde{e}(P_{pub}, H_1(ID) \cdot H_3(ID, T))^u\), CT_1, CT_2) \oplus (M'\|V') \oplus H_5(\(\tilde{e}(CT_2, IK_{ID1} \cdot TK_{ID1}), CT_1, CT_2 \)) \]

\[ = H_5(\(\tilde{e}(g_1^t, H_1(ID) \cdot H_3(ID, T))^u\), CT_1, CT_2) \oplus (M'\|V') \oplus H_5(\(\tilde{e}(g_1^u, H_1(ID) \cdot H_4(ID, T))^u\), CT_1, CT_2) \]

\[ = H_5(\(\tilde{e}(g_1^t, H_1(ID) \cdot H_3(ID, T))^u\), CT_1, CT_2) \oplus (M'\|V') \oplus H_5(\(\tilde{e}(g_1^u, H_1(ID) \cdot H_4(ID, T))^u\), CT_1, CT_2) = M'\|V'. \]

(3)

(6) Trapdoor: this algorithm is performed by a user who takes her/his initial key IK_{ID} = (IK_{ID1}, IK_{ID2}) and time key TK_{ID} = (TK_{ID1}, TK_{ID2}) as input to produce the trapdoor

\[ TD_{ID} = IK_{ID2} \cdot TK_{ID2} = H_2(ID)^s \cdot H_4(ID, T)^s. \]

(7) Test: this algorithm is performed by the CS who takes the public system parameters PSP and two ciphertext-trapdoor pairs (CT_1', TD_1') and (CT_2', TD_2'), where C T_1 = (CT_1, CT_2, CT_3, CT_4) and CT_2 = (CT_3, C
Table 5: Comparison of energy cost.

| Energy cost | RIBE [21] |IBE-ET [6] | IBET [10] | IBEET [11] | Our RIBEET |
|-------------|-----------|-----------|-----------|-----------|-----------|
| Performing encryption | 358.256 $\mu J$ | 755.224 $\mu J$ | 716.508 $\mu J$ | 813.292 $\mu J$ | 735.868 $\mu J$ |

(b) Compute $\tilde{e}(CT_{11}, R_y)$ and $\tilde{e}(CT_{11}, R_c)

(i) $\tilde{e}(CT_{11}, R_y) = \tilde{e}(g_1^{H_s(M_y V_y)}, H_b(M_y H_r(M_y V_y))) = \tilde{e}$

(ii) $\tilde{e}(CT_{11}, R_c) = \tilde{e}(g_1^{H_s(M_c V_c)}, H_b(M_c H_r(M_c V_c))) = \tilde{e}$

(c) Return 1 if $\tilde{e}(CT_{11}, R_y) = \tilde{e}(CT_{11}, R_c)$. Otherwise, return 0.

In the following, we present the details of [leftmargin = 0em]

(i) $R_c = CT_{14}/H_s(\tilde{e}(CT_{14}, TD_{14})) = H_b(M_x) \cdot H_s(\tilde{e})$

(ii) $R_q = CT_{14}/H_s(\tilde{e}(CT_{14}, TD_{14})) = H_b(M_q) \cdot H_s(\tilde{e})$

Proof. An algorithm $B$ is constructed to solve the BDH problem. The algorithm $B$ is given a BDH tuple $\langle g_1, g_2, g_T, \tilde{e}, g_1, g_2, g_T, g_2, g_T \rangle$ which is defined in Section 2. The algorithm $B$ can be seen as a challenger to find the answer of the BDH problem. The answer $H = \tilde{e}(g_1, g_2)$ can be found by interacting with the PPT type I adversary $A$ in the following security game.

(1) Setup: the challenger $B$ utilizes the BDH tuple to set $P_{pub} = g_1^x$ and then generates the public system parameters $PS = \{q, g_1, g_2, g_T, \tilde{e}, g_1, g_2, P_{pub}, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8\}$, where $H_i$ is a random oracle for $i = 1, 2, \ldots, 8$. In addition, the system life time $SLT = \{T_0, T_1, \ldots, T_3\}$ can be generated by the challenger $B$. Then, $B$ gives $A$ the public system parameters $PS$ and system life time $SLT$. Here, the adversary $A$ can issue queries to each random oracle as follows.

(a) $H_i$ query (ID): $A$ can utilize ID to obtain a response to the random oracle $H_i$ from the challenger $B$. To obtain the response, $B$ maintains a list, called ListH, which is composed of tuples, and the format of the tuple is $\langle 1, U_{ID}, u, rb \rangle$. The response is acquired from the ListH, which is initially empty and can be updated by the following steps.

(i) $B$ returns $U_{ID}$ as the response if ID exists in a tuple $(1, U_{ID}, u, rb)$ from the list.

(ii) Otherwise, $B$ picks a random value $u \in Z_q^*$ and a random bit $rb \in \{0, 1\}$ to compute

$$U_{ID} = \begin{cases} g_1^u, & \text{if } rb = 0, \\ g_2^u, & \text{if } rb = 1, \end{cases}$$

where $Pr[rb = 0] = \delta$ and $Pr[rb = 1] = 1 - \delta$ (which will be

5. Security Analysis

In this section, we give four theorems to show that the proposed scheme has the IND-ID-CCA security for type I and II adversaries and the OW-ID-CCA security for type III and IV adversaries.

Theorem 5. If the BDH assumption holds, the proposed RIBEET scheme satisfies the IND-ID-CCA security in the security game. More precisely, suppose that $A$ is a PPT type I adversary who has at least $\varepsilon$ advantage to break the RIBEET scheme. Then, there exists an algorithm $B$ to solve the BDH problem with the advantage

$$\varepsilon' \geq \frac{1}{q_{ID}} \left( \frac{e'(q_{IK} + q_T + 1)}{q} - q_{HR} \right),$$

where $q_{HR}$, $q_{IK}$, $q_T$, $q_B$, and $e$, respectively, are the number of queries to random oracle $H_5$, random oracle $H_6$, initialization key query, Trapdoor query, Decryption query, and Euler's number.

5. Security Analysis

In this section, we give four theorems to show that the proposed scheme has the IND-ID-CCA security for type I and II adversaries and the OW-ID-CCA security for type III and IV adversaries.

Theorem 5. If the BDH assumption holds, the proposed RIBEET scheme satisfies the IND-ID-CCA security in the security game. More precisely, suppose that $A$ is a PPT type I adversary who has at least $\varepsilon$ advantage to break the RIBEET scheme. Then, there exists an algorithm $B$ to solve the BDH problem with the advantage

$$\varepsilon' \geq \frac{1}{q_{ID}} \left( \frac{e'(q_{IK} + q_T + 1)}{q} - q_{HR} \right),$$

where $q_{HR}$, $q_{IK}$, $q_T$, $q_B$, and $e$, respectively, are the number of queries to random oracle $H_5$, random oracle $H_6$, initialization key query, Trapdoor query, Decryption query, and Euler's number.

5. Security Analysis

In this section, we give four theorems to show that the proposed scheme has the IND-ID-CCA security for type I and II adversaries and the OW-ID-CCA security for type III and IV adversaries.

Theorem 5. If the BDH assumption holds, the proposed RIBEET scheme satisfies the IND-ID-CCA security in the security game. More precisely, suppose that $A$ is a PPT type I adversary who has at least $\varepsilon$ advantage to break the RIBEET scheme. Then, there exists an algorithm $B$ to solve the BDH problem with the advantage

$$\varepsilon' \geq \frac{1}{q_{ID}} \left( \frac{e'(q_{IK} + q_T + 1)}{q} - q_{HR} \right),$$

where $q_{HR}$, $q_{IK}$, $q_T$, $q_B$, and $e$, respectively, are the number of queries to random oracle $H_5$, random oracle $H_6$, initialization key query, Trapdoor query, Decryption query, and Euler's number.

Proof. An algorithm $B$ is constructed to solve the BDH problem. The algorithm $B$ is given a BDH tuple $\langle g_1, g_2, g_T, \tilde{e}, g_1, g_2, g_T, g_2, g_T \rangle$ which is defined in Section 2. The algorithm $B$ can be seen as a challenger to find the answer of the BDH problem. The answer $H = \tilde{e}(g_1, g_2)$ can be found by interacting with the PPT type I adversary $A$ in the following security game.

(1) Setup: the challenger $B$ utilizes the BDH tuple to set $P_{pub} = g_1^x$ and then generates the public system parameters $PS = \{q, g_1, g_2, g_T, \tilde{e}, g_1, g_2, P_{pub}, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8\}$, where $H_i$ is a random oracle for $i = 1, 2, \ldots, 8$. In addition, the system life time $SLT = \{T_0, T_1, \ldots, T_3\}$ can be generated by the challenger $B$. Then, $B$ gives $A$ the public system parameters $PS$ and system life time $SLT$. Here, the adversary $A$ can issue queries to each random oracle as follows.

(a) $H_i$ query (ID): $A$ can utilize ID to obtain a response to the random oracle $H_i$ from the challenger $B$. To obtain the response, $B$ maintains a list, called ListH, which is composed of tuples, and the format of the tuple is $\langle 1, U_{ID}, u, rb \rangle$. The response is acquired from the ListH, which is initially empty and can be updated by the following steps.

(i) $B$ returns $U_{ID}$ as the response if ID exists in a tuple $(1, U_{ID}, u, rb)$ from the list.

(ii) Otherwise, $B$ picks a random value $u \in Z_q^*$ and a random bit $rb \in \{0, 1\}$ to compute

$$U_{ID} = \begin{cases} g_1^u, & \text{if } rb = 0, \\ g_2^u, & \text{if } rb = 1, \end{cases}$$

where $Pr[rb = 0] = \delta$ and $Pr[rb = 1] = 1 - \delta$ (which will be
discussed later). Then, $B$ adds the tuple $(ID, U_{ID}, u, rb)$ to the List$H_1$ and returns $U_{ID}$ to $\mathcal{A}_1$.

(b) $H_2$\text{query}(ID): $\mathcal{A}_1$ can utilize ID to obtain a response to the random oracle $H_2$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_2$ which is composed of tuples, and the format of the tuple is $(ID, V_{ID}, v, rb)$. The response is acquired from the List$H_2$ which is initially empty and can be updated by the following steps:

(a) $B$ returns $V_{ID}$ as the response if ID exists in a tuple $(ID, V_{ID}, v, rb)$ from the List$H_2$

(b) Otherwise, $B$ picks a random value $v \in \mathbb{Z}_q^*$ and utilizes ID to find $rb$ in the List$H_2$. Then, $B$ computes

$$V_{ID} = \begin{cases} g_2^v, & \text{if } rb = 0, \\ g_2^{rb}, & \text{if } rb = 1, \end{cases} \quad (6)$$

and adds the tuple $(ID, V_{ID}, v, rb)$ to List$H_2$. $B$ returns $V_{ID}$ to $\mathcal{A}_1$.

(c) $H_3$\text{query}(ID, T): $\mathcal{A}_1$ can utilize $(ID, T)$ to obtain a response to the random oracle $H_3$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_3$ which is composed of tuples, and the format of the tuple is $(ID, T, U_{IDT}, \eta)$. The response is acquired from the List$H_3$ which is initially empty and can be updated by the following steps:

(i) $B$ returns $U_{IDT}$ as the response if $(ID, T)$ exists in a tuple $(ID, T, U_{IDT}, \eta)$ from the List$H_3$

(ii) Otherwise, $B$ picks a random value $\eta \in \mathbb{Z}_q^*$ to compute $U_{IDT} = g_2^\eta$. Then, $B$ adds the tuple $(ID, T, U_{IDT}, \eta)$ to the List$H_3$ and returns $U_{IDT}$ to $\mathcal{A}_1$.

(d) $H_4$\text{query}(ID, T): $\mathcal{A}_1$ can utilize $(ID, T)$ to obtain a response to the random oracle $H_4$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_4$ which is composed of tuples, and the format of the tuple is $(ID, T, V_{IDT}, \zeta)$. The response is acquired from the List$H_4$ which is initially empty and can be updated by the following steps:

(i) $B$ returns $V_{IDT}$ as the response if $(ID, T)$ exists in a tuple $(ID, T, V_{IDT}, \zeta)$ from the List$H_4$

(ii) Otherwise, $B$ picks a random value $\zeta \in \mathbb{Z}_q^*$ to compute $V_{IDT} = g_2^\zeta$. Then, $B$ adds the tuple $(ID, T, V_{IDT}, \zeta)$ to the List$H_4$ and returns $V_{IDT}$ to $\mathcal{A}_1$.

(e) $H_5$\text{query}(W, CT_1, CT_2): $\mathcal{A}_1$ can utilize $(W, CT_1, CT_2)$ to obtain a response to the random oracle $H_5$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_5$ which is composed of tuples, and the format of the tuple is $(W, CT_1, CT_2, \omega)$. The response is acquired from the List$H_5$ which is initially empty and can be updated by the following steps:

(i) $B$ returns $\omega$ as the response if $(W, CT_1, CT_2)$ exists in a tuple $(W, CT_1, CT_2, \omega)$ from the List$H_5$

(ii) Otherwise, $B$ picks a random value $\omega \in \{0, 1\}^{k+1}$ and adds the tuple $(W, CT_1, CT_2, \omega)$ to the List$H_5$. Then, $B$ returns $\omega$ to $\mathcal{A}_1$.

(f) $H_6$\text{query}(M): $\mathcal{A}_1$ can utilize $M$ to obtain a response to the random oracle $H_6$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_6$ which is composed of tuples, and the format of the tuple is $(M, Q)$. The response is acquired from the List$H_6$ which is initially empty and can be updated by the following steps:

(i) $B$ returns $Q$ as the response if $M$ exists in a tuple $(M, Q)$ from the List$H_6$

(ii) Otherwise, $B$ picks a random point $Q \in G$, and adds the tuple $(M, Q)$ to the List$H_6$. Then, $B$ returns $Q$ to $\mathcal{A}_1$.

(g) $H_7$\text{query}(M, V): $\mathcal{A}_1$ can utilize $(M, V)$ to obtain a response to the random oracle $H_7$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_7$ which is composed of tuples, and the format of the tuple is $(M, V, \gamma)$. The response is acquired from the List$H_7$ which is initially empty and can be updated by the following steps:

(i) $B$ returns $\gamma$ as the response if $(M, V)$ exists in a tuple $(M, V, \gamma)$ from the List$H_7$

(ii) Otherwise, $B$ picks a random value $\gamma \in \mathbb{Z}_q^*$ and adds the tuple $(M, V, \gamma)$ to the List$H_7$. Then, $B$ returns $\gamma$ to $\mathcal{A}_1$.

(h) $H_8$\text{query}(N): $\mathcal{A}_1$ can utilize $N$ to obtain a response to the random oracle $H_8$ from the challenger $B$. ...
To obtain the response, $B$ maintains a list, called $\text{List}H_8$ which is composed of tuples, and the format of the tuple is $\langle N, S \rangle$. The response is acquired from the $\text{List}H_8$ which is initially empty and can be updated by the following steps

(i) $B$ returns $S$ as the response if $N$ exists in a tuple $\langle N, S \rangle$ from the $\text{List}H_8$

(ii) Otherwise, $B$ picks a random point $S \in G_2$ and adds the tuple $\langle N, S \rangle$ to the $\text{List}H_8$. Then, $B$ returns $S$ to $A_1$

(2) Phase 1: the adversary $A_1$ can, respectively, utilize $ID$, $\langle ID, T \rangle$, $\langle ID, T, CT \rangle$ and $\langle ID, T \rangle$ to issue the InitialKey query, Timekey query, Decryption query, and Trapdoor query. The response to each query can be obtained as follows

(a) InitialKey query($ID$): $A_1$ utilizes $ID$ to issue the query, while $B$, respectively, finds the corresponding tuples $\langle ID, U_{ID}, u, rb \rangle$ and $\langle ID, V_{ID}, v, rb \rangle$ from the $\text{List}H_1$ and the $\text{List}H_4$ according to $ID$. If $rb = 1$, $B$ interrupts this game. If $rb = 0$, $B$ uses $u$ and $v$ to define $IK_{ID} = (IK_{ID1}, IK_{ID2}) = ((g_2^{u})^*, (g_2^{v})^*)$. Then $B$ returns $IK_{ID}$ as the user initial key to $A_1$

(b) Timekey query($ID, T$): $A_1$ utilizes $ID, T$ to issue the query, while $B$, respectively, finds the corresponding tuples $\langle ID, T, U_{IDT}, \eta \rangle$ and $\langle ID, T, V_{IDT}, \zeta \rangle$ from the $\text{List}H_1$ and the $\text{List}H_4$ according to $ID, T$. $B$ uses $\eta$ and $\zeta$ to define $TK_{ID} = (TK_{ID1}, TK_{ID2}) = ((g_2^{\eta})^*, (g_2^{\zeta})^*)$. Then $B$ returns $TK_{ID}$ as the user time key to $A_1$

(c) Decryption query($ID, T, CT$): $A_1$ utilizes $ID, T, CT$ to issue the query, while $B$, respectively, finds the corresponding tuples $\langle ID, U_{IDT}, u, rb \rangle$, $\langle ID, V_{IDT}, v, rb \rangle$, $\langle ID, T, U_{IDT}, \eta \rangle$, and $\langle ID, T, V_{IDT}, \zeta \rangle$ from the $\text{List}H_1$, $\text{List}H_2$, $\text{List}H_3$, and the $\text{List}H_4$ according to $ID$ and $T$. The response of this query is acquired from these lists by performing the following tasks

(i) If $rb = 0$, $B$, respectively, uses $ID$ and $\langle ID, T \rangle$ to run InitialKeyQuery and TimekeyQuery to obtain $IK_{ID}$ and $TK_{ID}$. Then, $B$ utilizes $IK_{ID}$, $TK_{ID}$, and $CT$ to run the Decryption algorithm to produce the message $M$ which is sent to $A_1$

(ii) If $rb = 1$, $B$ utilizes $CT_1$ and $CT_2$, which are from $CT = (CT_1, CT_2, CT_3, CT_4)$, to find the corresponding tuple $\langle W, CT_1, CT_2, \omega \rangle$ from the $\text{List}H_5$. Then, $M' || V' = CT_3 \oplus \omega$ can be computed by using $CT_1$ and $\omega$. Further, $B$ utilizes $M'$ and $V'$ to find the corresponding tuples $\langle M, V, \gamma \rangle$ from the $\text{List}H_6$ and $\langle M, Q \rangle$ from the $\text{List}H_6$. Obviously, $\gamma$ and $Q$ can be obtained. If $S$ can be found in the corresponding tuple $\langle N, S \rangle$ from the $\text{List}H_8$ such that $CT_4 = Q' \cdot S$ holds, $B$ will confirm whether $CT_2 = g_2^\gamma$ holds. If $CT_1 = g_2^\omega$, the message $M'$ is sent to $A_1$

(d) Trapdoor query($ID, T$): $A_1$ utilizes ($ID, T$) to issue the query, while $B$, respectively, uses $ID$ and ($ID, T$) to run InitialKeyQuery and TimekeyQuery to obtain $IK_{ID} = (IK_{ID1}, IK_{ID2})$ and $TK_{ID} = (TK_{ID1}, TK_{ID2})$. Then, $B$ utilizes $IK_{ID2}$ and $TK_{ID2}$ to produce the trapdoor $TD_{ID} = IK_{ID2} \cdot TK_{ID2}$ which is sent to $A_1$

(3) Challenge: when the phase 1 is over, $A_1$ outputs a tuple $\langle ID^*, T^*, M^*_1, M^*_2 \rangle$ as the target of the challenge. $B$ utilizes $ID^*$ to find the corresponding tuples $\langle ID, U_{ID}, u, rb \rangle$ from the $\text{List}H_1$. If $rb = 0$, $B$ interrupts this game. If $rb = 1$, $B$ randomly selects $b \in \{0, 1\}$ and $\gamma \in \{0, 1\}^k$, and a random point $CT_1^* \in G_2$ is chosen. Finally, the challenge ciphertext $CT^* = (CT_1^*, CT_2, CT_3^*, CT_4^*)$ is sent to $A_1$

(4) Phase 2: $A_1$ can issue the same query as phase 1, but it must be under the condition of $ID \neq ID^*$ and $CT \neq CT^*$

(5) Guess: $A_1$ responds to $B$ with a guess $b' \in \{0, 1\}$. If $b' \neq b$, $B$ responds with failure and terminates. Otherwise, $A_1$ wins the game. Then, $B$ randomly selects a tuple $\langle W^*, CT_1^*, CT_2^*, \omega' \rangle$ from the $\text{List}H_5$ and calculates $H_5(\bar{e}(g_1, g_2)^{abc} \cdot \bar{e}(g_1^\omega, g_2)^{\gamma} \cdot CT_1^* \cdot CT_2^*) = (M_3' || V) \oplus CT_3^*$. where $\bar{e}(g_1, g_2)^{abc} \cdot \bar{e}(g_1^\omega, g_2)^{\gamma} = W^*$. Hence, $B$ can output the BDH solution $A = (W^* \bar{e}(g_1^\omega, g_2)^{\gamma})^{u'-1}$ due to $\bar{e}(g_1, g_2)^{abc} = (W^* \bar{e}(g_1^\omega, g_2)^{\gamma})^{u'-1}$

\[ \square \]

Analysis. Let us start with two cases, namely, the simulation of $H_1$ query for $i = 1, 2, \ldots, 8$ and the simulation of decryption query. For the $H_1, H_2, H_3, H_4, H_5, H_6$, and $H_7$ queries, it is obvious that the simulations are perfect because there exists no relationship between the constructions of these queries and the solution of the BDH problem. For the $H_5$ and $H_6$ queries, we consider two events $E_{H_5}$ and $E_{H_6}$ which, respectively, issues the $H_5$ query with ($\bar{e}$
Then, we have a security game. More precisely, suppose that \( \mathcal{A} \) is the number of decryption query. Then, we obtain \( \Pr[\text{DecErr}^\prime] \leq q_D / q \).

\[
\Pr[b = b'] = \Pr[b = b' | E] \Pr[E] + \Pr[b = b' | \neg E] \Pr[\neg E] \leq \Pr[E] + \left( \frac{1}{2} \right) \Pr[\neg E] = \Pr[E] + \frac{1}{2} \left( 1 - \Pr[E] \right) = \frac{1}{2} \Pr[E] + \frac{1}{2}. \tag{7}
\]

According to above inequality, \( \varepsilon = \Pr[b = b'] - 1/2 \) and \( E = (E_{H_1} \lor E_{H_2} \lor E_{\text{DecErr}}) \neg E_{\text{Abort}} \), we have

\[
\varepsilon = \Pr[b = b'] - \frac{1}{2} \leq \Pr[E] \leq \frac{\Pr[E_{H_1}^1] + \Pr[E_{H_2}^2] + \Pr[\text{DecErr}^\prime]}{\Pr[\neg E_{\text{Abort}}]}.
\tag{8}
\]

Moreover, we obtain

\[
\Pr[E_{H_1}^1] \geq \varepsilon \cdot \Pr[\neg E_{\text{Abort}}] - \Pr[\text{DecErr}^\prime] - \Pr[E_{H_2}^2]. \tag{9}
\]

Since \( \Pr[\neg E_{\text{Abort}}] = \delta^{\text{time}} (1 - \delta) \), we can gain \( \Pr[\neg E_{\text{Abort}}] \geq 1/e(\frac{q_{H_1}}{q} + \frac{q_{T_1}}{q} + 1) \) when \( \delta = 1 - (1/(\frac{q_{H_1}}{q} + \frac{q_{T_1}}{q} + 1)) \). Then, we have

\[
\Pr[E_{H_1}^1] \geq \frac{\varepsilon}{e(\frac{q_{H_1}}{q} + \frac{q_{T_1}}{q} + 1)} - \frac{q_D}{q} - \frac{q_{H_1}}{q}. \tag{10}
\]

Here, the adversary \( \mathcal{A} \) can distinguish the target ciphertext \( C_{T_1} \) is the real one when \( E_{H_1}^1 \) occurs. In addition, the tuple \( \langle \tilde{e}(g_1, g_2)^{abc}, \tilde{e}(g_1^a, g_3)^{abc}, C_{T_1}^1, C_{T_1}^2 \rangle \) has been added in the ListH.5. If the challenger \( \mathcal{B} \) picks the correct tuple from the ListH, \( \mathcal{B} \) wins this security game. Meanwhile, the advantage of solving the BDH problem is

\[
\varepsilon' \geq \frac{1}{q_{H_1}} \Pr[E_{H_1}^1] \geq \frac{1}{q_{H_1}} \left[ \frac{\varepsilon}{e(\frac{q_{H_1}}{q} + \frac{q_{T_1}}{q} + 1)} - \frac{q_D}{q} - \frac{q_{H_1}}{q} \right]. \tag{11}
\]

**Theorem 6.** If the BDH assumption holds, the proposed RIBEEET scheme satisfies the IND-ID-CCA security in the security game. More precisely, suppose that \( \mathcal{A} \) is a PPT type 2 adversary who has at least \( \varepsilon \) advantage to break the RIBEEET scheme. Then, there exists an algorithm \( \mathcal{B} \) to solve the BDH problem with the advantage

\[
\varepsilon' \geq \left( \frac{1}{q_{H_1}} \right) \left[ \frac{\varepsilon}{e(\frac{q_{H_1}}{q} + \frac{q_{T_1}}{q} + 1)} - \frac{q_D}{q} - \frac{q_{H_1}}{q} \right]. \tag{12}
\]

Next, we discuss an event \( E \) which states that the simulation of this security game will not be interrupted. Here, we can obtain \( E = (E_{H_1} \lor E_{H_2} \lor E_{\text{DecErr}}) \neg E_{\text{Abort}} \), where \( E_{\text{Abort}} \) is defined as the event that the challenger \( \mathcal{B} \) interrupts this security game. Since \( \mathcal{B} \) guesses \( b' \) with the advantage \( \leq 1/2 \), the \( \Pr[b = b' | \neg E] \leq 1/2 \) can be obtained if \( E \) does not occur. Further, we have

\[
\Pr[\text{DecErr}^\prime] = \Pr[\text{Abort}] \leq \frac{1}{2} \Pr[E] + \frac{1}{2}. \tag{7}
\]

where \( q_{H_1}, q_{H_2}, q_{T_1}, q_{H_3}, q_D, \) and \( q_e \) respectively, are the number of queries to random oracle \( H_1, \) random oracle \( H_3, \) Timekey query, Trapdoor query, Decryption query, and Euler's number.

**Proof.** An algorithm \( \mathcal{B} \) is constructed to solve the BDH problem. The algorithm \( \mathcal{B} \) is given a BDH tuple \( \langle q, G_1, G_2, G_T, \tilde{e}, g_1, g_2, g_3, g_{ab}, g_{bc}, g_{ac}, g_{abc}, g_{abc}^a \rangle \) which is defined in Section 2. The algorithm \( \mathcal{B} \) can be seen as a challenger to find the answer of the BDH problem. The answer \( A = \tilde{e}(g_1, g_2)^{abc} \) can be found by interacting with the PPT type II adversary \( \mathcal{A}_2 \) in the following security game.

(1) Setup: the challenger \( \mathcal{B} \) utilizes the BDH tuple to set \( P_{pub} = g_1^a \) and then generates the public system parameters \( \text{PSP} = \{ q, G_1, G_2, G_T, \tilde{e}, g_1, g_2, P_{pub}, H_1, H_3, H_4, H_5, H_6, H_7, H_8 \} \), where \( H_i \) is a random oracle for \( i = 1, 2, \ldots, 8 \). In addition, the system life time \( \text{SLT} = \{ T_1, T_2, \ldots, T_8 \} \) can be generated by the challenger \( \mathcal{B} \). Then, \( \mathcal{B} \) gives \( \mathcal{A}_2 \) the public system parameters \( \text{PSP} \) and system life time \( \text{SLT} \). Here, the adversary \( \mathcal{A}_2 \) can issue queries to each random oracle as follows

(a) \( H_i \) query (ID): \( \mathcal{A}_2 \) can utilize ID to obtain a response to the random oracle \( H_i \) from the challenger \( \mathcal{B} \). To obtain the response, \( \mathcal{B} \) maintains a list, called ListH1 which is composed of tuples, and the format of the tuple is \( \langle \text{ID}, \text{U}_\text{ID}, u \rangle \). The response is acquired from the ListH1 which is initially empty

(i) \( \mathcal{B} \) returns \( \text{U}_\text{ID} \) as the response if \( \text{ID} \) exists in a tuple \( \langle \text{ID}, \text{U}_\text{ID}, u \rangle \) from the ListH1

(ii) Otherwise, \( \mathcal{B} \) picks a random value \( u \in Z_q^* \) to compute \( \text{U}_\text{ID} = g_2^u \). Then, \( \mathcal{B} \) adds the tuple \( \langle \text{ID}, \text{U}_\text{ID}, u \rangle \) to the ListH1 and returns \( \text{U}_\text{ID} \) to \( \mathcal{A}_2 \).
(b) $H_2\text{query}(ID)$: $A_2$ can utilize $ID$ to obtain a response to the random oracle $H_2$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_2$ which is composed of tuples, and the format of the tuple is $(ID, V_{ID}, v)$. The response is acquired from the List$H_2$ which is initially empty and can be updated by the following steps

(i) $B$ returns $V_{ID}$ as the response if $ID$ exists in a tuple $(ID, V_{ID}, v)$ from the List$H_2$

(ii) Otherwise, $B$ picks a random value $v \in Z_q^*$ to compute $V_{ID} = g_v^r$. Then, $B$ adds the tuple $(ID, V_{ID}, v)$ to the List$H_2$ and returns $V_{ID}$ to $A_2$

(c) $H_3\text{query}(ID, T)$: $A_2$ can utilize $(ID, T)$ to obtain a response to the random oracle $H_3$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_3$ which is composed of tuples, and the format of the tuple is $(ID, T, U_{IDT}, \eta, rb)$. The response is acquired from the List$H_3$ which is initially empty and can be updated by the following steps

(i) $B$ returns $U_{IDT}$ as the response if $(ID, T)$ exists in a tuple $(ID, T, U_{IDT}, \eta, rb)$ from the List$H_3$

(ii) Otherwise, $B$ picks a random value $\eta \in Z_q^*$ and a random bit $rb \in \{0, 1\}$ to compute

$$U_{IDT} = \begin{cases} g_2^\eta, & \text{if } rb = 0, \\ g_2^{\eta rb}, & \text{if } rb = 1, \end{cases}$$

(13)

where $Pr[rb = 0] = \delta$ and $Pr[rb = 1] = 1 - \delta$ (which will be discussed later). Then, $B$ adds the tuple $(ID, T, U_{IDT}, \eta, rb)$ to the List$H_3$ and returns $U_{IDT}$ to $A_2$

(d) $H_4\text{query}(ID, T)$: $A_2$ can utilize $(ID, T)$ to obtain a response to the random oracle $H_4$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_4$ which is composed of tuples, and the format of the tuple is $(ID, T, V_{IDT}, \zeta, rb)$. The response is acquired from the List$H_4$ which is initially empty and can be updated by the following steps

(i) $B$ returns $V_{IDT}$ as the response if $(ID, T)$ exists in a tuple $(ID, T, V_{IDT}, \zeta, rb)$ from the List$H_4$

(ii) Otherwise, $B$ picks a random value $\zeta \in Z_q^*$ and utilizes $(ID, T)$ to find $rb$ in the List$H_4$. Then, $B$ com-

$$V_{IDT} = \begin{cases} g_2^\zeta, & \text{if } rb = 0, \\ g_2^{\zeta rb}, & \text{if } rb = 1, \end{cases}$$

(14)

and add the tuple $(ID, T, V_{IDT}, \zeta, rb)$ to List$H_4$. $B$ returns $V_{IDT}$ to $A_2$

(e) $H_5\text{query}(W, CT_1, CT_2)$: $A_2$ can utilize $(W, CT_1, CT_2)$ to obtain a response to the random oracle $H_5$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_5$ which is composed of tuples, and the format of the tuple is $(W, CT_1, CT_2, \omega)$. The response is acquired from the List$H_5$ which is initially empty and can be updated by the following steps

(i) $B$ returns $\omega$ as the response if $(W, CT_1, CT_2)$ exists in a tuple $(W, CT_1, CT_2, \omega)$ from the List$H_5$

(ii) Otherwise, $B$ picks a random value $\omega \in \{0, 1\}^{l+1}$ and adds the tuple $(W, CT_1, CT_2, \omega)$ to the List$H_5$. Then, $B$ returns $\omega$ to $A_2$

(f) $H_6\text{query}(M)$: $A_2$ can utilize $M$ to obtain a response to the random oracle $H_6$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_6$ which is composed of tuples, and the format of the tuple is $(M, Q)$. The response is acquired from the List$H_6$ which is initially empty and can be updated by the following steps

(i) $B$ returns $Q$ as the response if $M$ exists in a tuple $(M, Q)$ from the List$H_6$

(ii) Otherwise, $B$ picks a random point $Q \in G_2$ and adds the tuple $(M, Q)$ to the List$H_6$. Then, $B$ returns $Q$ to $A_2$

(g) $H_7\text{query}(M, V)$: $A_2$ can utilize $(M, V)$ to obtain a response to the random oracle $H_7$ from the challenger $B$. To obtain the response, $B$ maintains a list, called List$H_7$ which is composed of tuples, and the format of the tuple is $(M, V, \gamma)$. The response is acquired from the List$H_7$ which is initially empty and can be updated by the following steps

(i) $B$ returns $\gamma$ as the response if $(M, V)$ exists in a tuple $(M, V, \gamma)$ from the List$H_7$
(ii) Otherwise, $\mathcal{B}$ picks a random value $\gamma \in Z_q^*$ and adds the tuple $\langle M, V, \gamma \rangle$ to the List$H_T$. Then, $\mathcal{B}$ returns $\gamma$ to $\mathcal{A}_2$

(h) $H_q$ query($N$): $\mathcal{A}_2$ can utilize $N$ to obtain a response to the random oracle $H_q$ from the challenger $\mathcal{B}$. To obtain the response, $\mathcal{B}$ maintains a list, called List$H_q$ which is composed of tuples, and the format of the tuple is $(N, S)$. The response is acquired from the List$H_q$ which is initially empty and can be updated by the following steps

(i) $\mathcal{B}$ returns $S$ as the response if $N$ exists in a tuple $(N, S)$ from the List$H_q$

(ii) Otherwise, $\mathcal{B}$ picks a random point $S \in G_2$ and adds the tuple $(N, S)$ to the List$H_q$. Then, $\mathcal{B}$ returns $S$ to $\mathcal{A}_2$

(2) Phase 1: the adversary $\mathcal{A}_2$ can, respectively, utilize ID, (ID, T), (ID, T, CT), and (ID, T) to issue the InitialKey query, Timekey query, Decryption query, and Trapdoor query. The response to each query can be obtained as follows

(a) InitialKey query(ID): $\mathcal{A}_2$ utilizes ID to issue the query, while $\mathcal{B}$, respectively, finds the corresponding tuples $(ID, U_{ID}, u)$ and $(ID, V_{ID}, v)$ from the List$H_1$ and the List$H_0$ according to ID. $\mathcal{B}$ use $u$ and $v$ to define $IK_{ID} = (IK_{ID1}, IK_{ID2}) = ((g_2)^u, (g_2)^v)$. Then, $\mathcal{B}$ returns $IK_{ID}$ as the user initial key to $\mathcal{A}_1$

(b) Timekey query(ID, T): $\mathcal{A}_2$ utilizes (ID, T) to issue the query, while $\mathcal{B}$, respectively, finds the corresponding tuples $(ID, T, U_{IDT}, \eta, rb)$ and $(ID, T, V_{IDT}, \zeta, rb)$ from the List$H_3$ and the List$H_4$ according to (ID, T). If $rb = 1$, $\mathcal{B}$ interrupts this game. If $rb = 0$, $\mathcal{B}$ use $\eta$ and $\zeta$ to define $TK_{ID} = (TK_{ID1}, TK_{ID2}) = ((g_2^\eta)^{rb}, (g_2^\zeta)^{rb})$. Then, $\mathcal{B}$ returns $TK_{ID}$ as the user time key to $\mathcal{A}_2$

(c) Decryption query(ID, T, CT): $\mathcal{A}_2$ utilizes (ID, T, CT) to issue the query, while $\mathcal{B}$, respectively, finds the corresponding tuples $(ID, U_{ID}, u)$, $(ID, V_{ID}, v)$, $(ID, T, U_{IDT}, \eta, rb)$, and $(ID, T, V_{IDT}, \zeta, rb)$ from the List$H_1$, List$H_2$, List$H_3$, and List$H_4$ according to ID and T. The response of this query is acquired from these lists by performing the following tasks

(i) If $rb = 0$, $\mathcal{B}$, respectively, uses ID and (ID, T) to run InitialKey query and Timekey query to obtain $IK_{ID}$ and $TK_{ID}$. Then, $\mathcal{B}$ utilizes $IK_{ID}$, $TK_{ID}$, and CT to run the Decryption algorithm to produce the message $M$ which is sent to $\mathcal{A}_2$

(ii) If $rb = 1$, $\mathcal{B}$ utilizes CT$\mathcal{T}_1$ and CT$\mathcal{T}_2$, which are from CT$\mathcal{T}_1 = \langle CT_1, CT_2, CT_3, CT_4 \rangle$, to find the corresponding tuple $(W, CT_1, CT_2, \omega)$ from the List$H_5$. Then, $M || V' = CT_3 \otimes \omega$ can be computed by using CT$\mathcal{T}_3$ and $\omega$. Further, $\mathcal{B}$ utilizes $M'$ and $V'$ to find the corresponding tuples $(M, V, \gamma)$ from the List$H_7$ and $(M, Q)$ from the List$H_6$. Obviously, $\gamma$ and $Q$ can be obtained. If S can be found in the corresponding tuple $(N, S)$ from the List$H_q$ such that $CT_4 = Q'. \ S$ holds, $\mathcal{B}$ will confirm whether $CT_1 = g_{i_1}^2$ holds. If $CT_1 = g_{i_1}^2$, the message $M'$ is sent to $\mathcal{A}_2$

(1) Trapdoor query(ID, T): $\mathcal{A}_2$ utilizes (ID, T) to issue the query, while $\mathcal{B}$, respectively, uses ID and (ID, T) to run InitialKey query and Timekey query to obtain $IK_{ID}$ and $TK_{ID}$. $\mathcal{B}$ uses $IK_{ID}$, $TK_{ID}$, and CT to produce the trapdoor TD$\mathcal{T}_1 = IK_{ID} \otimes TK_{ID}$ which is sent to $\mathcal{A}_2$

(3) Challenge: when phase 1 is over, $\mathcal{A}_2$ outputs a tuple $(ID^*, T^*, M_{i_1}^*, M_{i_2}^*)$ as the target of the challenge. $\mathcal{B}$ utilizes $(ID^*, T^*)$ to find the corresponding tuples $(ID, T, U_{IDT}, U_{ID}, U_{IDT}, \eta, rb)$ from the List$H_1$. If $rb = 0$, $\mathcal{B}$ interrupts this game. If $rb = 1$, $\mathcal{B}$ randomly selects $b \in \{0, 1\}$ and $V \in \{0, 1\}$ to run $H_q$ query with $M^*_b$ and $V$. $\mathcal{B}$, respectively, finds the corresponding tuples $(ID, T, U_{IDT}, \eta, rb)$ from the List$H_3$ and the List$H_4$ according to (ID, T). $\mathcal{B}$ uses $\eta$ and $\zeta$ to define $TK_{ID} = (TK_{ID1}, TK_{ID2}) = ((g_2^\eta)^{rb}, (g_2^\zeta)^{rb})$. Then, $\mathcal{B}$ returns $TK_{ID}$ as the user time key to $\mathcal{A}_2$

(2) Phase 2: $\mathcal{A}_2$ can issue the same query as phase 1, but it must be under the condition of ID $\neq ID^*$ and $CT \neq CT^*$

(3) Guess: $\mathcal{A}_2$ responds to $\mathcal{B}$ with a guess $b' \in \{0, 1\}$. If $b' \neq b$, $\mathcal{B}$ responds with failure and terminates. Otherwise, $\mathcal{A}_2$ wins the game. Then, $\mathcal{B}$ randomly selects a tuple $(W^*, CT^*_1, CT^*_2, \omega^*)$ from the List$H_5$ and outputs the BDH solution $A = (W^* \overline{\epsilon}(g_2^a, g_2^b))^{n^{-1}}$ due to $\overline{\epsilon}(g_1, g_2)^{abc} = (W^* \overline{\epsilon}(g_1, g_2)^a)^{n^{-1}}$

The security analysis is similar to Theorem 5. We obtain that $\mathcal{B}$’s advantage to solve the BDH problem is $\epsilon' \geq (1/
Theorem 7. If the BDH assumption holds, the proposed RIBEET scheme satisfies the OW-ID-CCA security in the security game. More precisely, suppose that \( \mathcal{A}_3 \) is a PPT type 3 adversary who has at least \( \varepsilon \) advantage to break the RIBEET scheme. Then, there exists an algorithm \( \mathcal{B} \) to solve the BDH problem with the advantage

\[
\epsilon' \geq \left( \frac{1}{q_{H_5}} \right) \left[ \frac{\epsilon - (1/2^k)}{\epsilon'/q_{H_5} + 1} \right] - \frac{q_D}{q},
\]

where \( q_{H_5}, q_{H_5'}, q_{D'}, \) and \( \epsilon \), respectively, are the number of queries to random oracle \( H_5 \), InitialKey query, Decryption query, and Euler's number.

Proof. An algorithm \( \mathcal{B} \) is constructed to solve the BDH problem. The algorithm \( \mathcal{B} \) is given a BDH tuple \( (q, G_1, G_2, \overline{G}_7, \bar{e}, g_1, g_2, g_1', g_2', g_2', g_2') \) which is defined in Section 2. The algorithm \( \mathcal{B} \) can be seen as a challenger to find the answer of the BDH problem. The answer \( A = \bar{e}(g_1, g_2') \) can be found by interacting with the PPT type III adversary \( \mathcal{A}_3 \) in the following security game.

(i) Setup: The challenger \( \mathcal{B} \) utilizes the BDH tuple to set \( P_{pub} = g_1^t \), and then generates the public system parameters \( \text{PSP} = \{ q, G_1, G_2, \overline{G}_7, \bar{e}, g_1, g_2, P_{pub}, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8 \} \), where \( H_i \) is a random oracle for \( i = 1, 2, \ldots, 8 \). In addition, the system life time \( SLT = \{ T_0, T_1, \ldots, T_{8} \} \) can be generated by the challenger \( \mathcal{B} \). Then \( \mathcal{B} \) gives \( \mathcal{A}_3 \) the public system parameters \( \text{PSP} \) and system life time \( SLT \). Here, the adversary \( \mathcal{A}_3 \) can issue queries to each random oracle as below.

(a) \( H_1 \) query (ID): \( \mathcal{B} \) answers \( \mathcal{A}_3 \) in the same form as the proof of Theorem 5

(b) \( H_2 \) query (ID): \( \mathcal{A}_3 \) can utilize ID to obtain a response to the random oracle \( H_2 \) from the challenger \( \mathcal{B} \). To obtain the response, \( \mathcal{B} \) maintains a list, called ListH2, which is composed of tuples, and the format of the tuple is \( \{ ID, V_{ID}, v, rb \} \). The response is acquired from the ListH2 which is initially empty and can be updated by the following steps

(i) \( \mathcal{B} \) returns \( V_{ID} \) as the response if ID exists in a tuple \( (ID, V_{ID}, v, rb) \) from the ListH2

(ii) Otherwise, \( \mathcal{B} \) picks a random value \( v \in \mathbb{Z}_q^* \) and utilizes ID to find rb in the ListH2. Then \( \mathcal{B} \) computes \( V_{ID} = g_2^t \) and adds the tuple \( (ID, V_{ID}, v, rb) \) to ListH2. \( \mathcal{B} \) returns \( V_{ID} \) to \( \mathcal{A}_3 \)

(c) \( H_3 \) query (ID): \( \mathcal{B} \) answers \( \mathcal{A}_3 \) in the same form as the proof of Theorem 5

(ii) Phase 1: The adversary \( \mathcal{A}_3 \) can, respectively, utilize ID, \( (ID, T) \), \( (ID, T, CT) \) and \( (ID, T) \) to issue the InitialKey query, Timekey query, Decryption query and Trapdoor query. The response to each query can be obtained as follows

(1) InitialKey query (ID): \( \mathcal{B} \) answers \( \mathcal{A}_3 \) in the same form as the proof of Theorem 5

(2) Timekey query (ID, T): \( \mathcal{B} \) answers \( \mathcal{A}_3 \) in the same form as the proof of Theorem 5

(3) Decryption query (ID, T, CT): \( \mathcal{A}_3 \) utilizes \( (ID, T, C, T) \) to issue the query, while \( \mathcal{B} \), respectively, finds the corresponding tuples \( \{ ID, U_{ID\gamma}, rb \}, \{ ID, V_{ID}, v, rb \}, \{ ID, T, U_{ID\gamma}, \eta \} \) and \( \{ ID, T, V_{ID\gamma}, \zeta \} \) from the ListH1, ListH2, ListH3 and the ListH4 according to ID and T. The response of this query is acquired from these lists by performing the following tasks

(i) If \( rb = 0 \), \( \mathcal{B} \), respectively, uses ID and (ID, T) to run InitialKey query and Timekey query to obtain \( IK_{ID} \) and \( TK_{ID} \). Then \( \mathcal{B} \) utilizes \( IK_{ID} \), \( TK_{ID} \) and \( CT \) to run Decryption algorithm to produce the message \( M \) which is sent to \( \mathcal{A}_3 \)

(ii) If \( rb = 1 \), \( \mathcal{B} \) utilizes \( CT_1 \) and \( CT_2 \), which are from \( CT = \{ CT_1, CT_2, CT_3, CT_4 \} \), to find the corresponding tuple \( \{ W, CT_1, CT_2, \omega \} \) from the ListH2. Then \( M' = V' = CT_1 \otimes \omega \) can be computed by using \( CT_3 \) and \( \omega \). Further, \( \mathcal{B} \) utilizes \( M' \) and \( V' \) to find the corresponding tuples \( \{ M, V, Y \} \) from the ListH7 and \( \{ M, Q \} \) from the ListH8. Obviously, \( y \) and \( Q \) can be obtained. After that, \( \mathcal{B} \) utilizes ID and (ID, T) to run InitialKey query and Timekey query to obtain \( IK_{ID2} \) and \( TK_{ID2} \) and computes \( IK_{ID2} \cdot TK_{ID2} \cdot T \) \( K_{ID2} = g_3^{(v \cdot \zeta)} \). If \( S \) can be found in the corresponding tuple \( \{ \bar{e}(CT_2, g_2^{(v \cdot \zeta)}), S \} \) from the ListH8 such that \( CT_1 = Q' \cdot S \) holds, \( \mathcal{B} \) will confirm whether \( CT_1 = g_2^t \) holds. If \( CT_1 = g_2^t \), the message \( M' \) is sent to \( \mathcal{A}_3 \)

(d) Trapdoor query (ID, T): \( \mathcal{A}_3 \) utilizes (ID, T) to issue the query, while \( \mathcal{B} \), respectively, uses ID and (ID, T) to run InitialKey query and Timekey query to obtain \( IK_{ID} = (IK_{ID2}, IK_{ID2}) \) and \( TK_{ID} = (TK_{ID2}, T) \) \( K_{ID2} \). Then \( \mathcal{B} \) utilizes \( IK_{ID2} \) and \( TK_{ID2} \) to produce
the trapdoor $TD_{ID} = IK_{ID2} \cdot TK_{ID2}$ which is sent to $a_3$.

(e) **Challenge:** When the phase 1 is over, $a_3$ outputs a tuple $\langle ID^*, T^*, M^* \rangle$ as the target of the challenge. $B$ utilizes $ID^*$ to find the corresponding tuples $\langle ID^, U_{ID}, u, rb \rangle$ from the $List_{H_1}$. If $rb = 0$, $B$ interrupts this game. If $rb = 1$, $B$ randomly selects $V \in \{0, 1\}^i$ to run $H_2$ query with $M^*$ and $V$. Then $\gamma$ can be obtained. $B$ utilizes $\gamma$ to set $CT_1^* = g_{f_1}$ and find $H_6$ query ($\bar{e}(CT_2^*, g_1^{(\gamma + \epsilon^\gamma)})$) to get $Q$ and $S$ such that $CT_2^* = Q^\gamma \cdot S$. In addition, $B$ sets $CT_3^* = g_{f_2}^{(\gamma + \epsilon^\gamma)}$, while a random value $CT_3 \in \{0, 1\}^{1/2}$ is chosen. Finally, the challenge ciphertext $CT^* = (C, CT_1^*, CT_2^*, CT_3^*)$ is sent to $a_3$.

(f) **Phase 2:** $a_3$ can issue the same query as the phase 1, but it must be under the condition of $ID \neq ID^*$ and $CT \neq CT^*$.

(g) **Guess:** $a_3$ responds to $B$ with a guess $M' \in \{0, 1\}^\lambda$. If $M' \neq M$, $B$ responds with failure and terminates. Otherwise, $a_3$ wins the game. Then $B$ randomly selects a tuple $\langle W^*, CT_1^*, CT_2^*, \omega^* \rangle$ from the $List_{H_4}$, and outputs the BDH solution $A = (W^* \bar{e}(g_{f_1}, g_2)^{\omega^*})^{1/2}$ due to $\bar{e}(g_1, g_2)^{abc} = (W^* \bar{e}(g_{f_1}^{\gamma}, g_2)^{\omega^*})^{1/2}$.

The security analysis is similar to Theorem 5. We obtain that $B$’s advantage to solve the BDH problem is $\epsilon' \geq (1/q_{H_4}) \Pr[\epsilon_{H_4} \geq (1/2^\lambda)]/\epsilon(q_{IK} + 1) - q_{TD}/q$. □

**Theorem 8.** If the BDH assumption holds, the proposed RIBEET scheme satisfies the OW-ID-CCA security in the security game. More precisely, suppose that $a_4$ is a PPT type 4 adversary who has at least $\epsilon$ advantage to break the RIBEET scheme. Then, there exists an algorithm $B$ to solve the BDH problem with the advantage.

$$\epsilon' \geq \left( 1/q_{H_4} \right) \left[ \frac{\epsilon - (1/2^\lambda)}{\epsilon(q_{IK} + 1)} \right] - \frac{q_{TD}}{q} \quad (16)$$

where $q_{H_4}^*$, $q_{TK}$, $q_{ID}$, and $\epsilon$, respectively, are the number of queries to random oracle $H_4$, Timekey query, Decryption query, and Euler's number.

**Proof.** An algorithm $B$ is constructed to solve the BDH problem. The algorithm $B$ is given a BDH tuple $\langle q, G_1, G_2, G_T, \bar{e}, g_1, g_2, P_{pub}, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8 \rangle$, where $H_i$ is a random oracle for $i = 1, 2, \ldots, 8$. In addition, the system life time $SLT = \{T_0, T_1, \ldots, T_i\}$ can be generated by the challenger $B$. Then $B$ gives $a_4$ the public system parameters $PSP = \{q, G_1, G_2, G_T, \bar{e}, g_1, g_2, P_{pub}, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8 \}$, where $H_i$ is a random oracle for $i = 1, 2, \ldots, 8$. In addition, the system life time $SLT = \{T_0, T_1, \ldots, T_i\}$ can be generated by the challenger $B$. Then $B$ gives $a_4$ the public system parameters $PSP$ and system life time $SLT$. Here, the adversary $a_4$ can issue queries to each random oracle as below.

(a) $H_1 - H_4$ queries: $B$ answers $a_4$ in the same form as the proof of Theorem 6.

(b) $H_4 - H_8$ queries: $B$ answers $a_4$ in the same form as the proof of Theorem 5.

(ii) **Phase 1:** The adversary $a_4$ can, respectively, utilize $ID, (ID, T), (ID, T, CT)$ and $(ID, T)$ to issue the InitialKey query, Timekey query, Decryption query and Trapdoor query. The response to each query can be obtained as follows.

(a) **InitialKey query (ID):** $B$ answers $a_4$ in the same form as the proof of Theorem 6.

(b) **Timekey query (ID, T):** $B$ answers $a_4$ in the same form as the proof of Theorem 6.

(c) **Decryption query (ID, T, CT):** $B$ answers $a_4$ in the same form as the proof of Theorem 7.

(d) **Trapdoor query (ID, T):** $B$ answers $a_4$ in the same form as the proof of Theorem 7.

(1) **Challenge:** When the phase 1 is over, $a_4$ outputs a tuple $\langle ID^*, T^*, M^* \rangle$ as the target of the challenge. $B$ utilizes $ID^*$ to find the corresponding tuples $\langle ID, U_{ID}, u, rb \rangle$ from the $List_{H_1}$. If $rb = 0$, $B$ interrupts this game. If $rb = 1$, $B$ randomly selects $V \in \{0, 1\}^i$ to run $H_2$ query with $M^*$ and $V$. Then $\gamma$ can be obtained. $B$ utilizes $\gamma$ to set $CT_1^* = g_{f_1}$ and find $H_6$ query ($\bar{e}(CT_2^*, g_1^{(\gamma + \epsilon^\gamma)})$) to get $Q$ and $S$ such that $CT_2^* = Q^\gamma \cdot S$. In addition, $B$ sets $CT_3^* = g_{f_2}^{(\gamma + \epsilon^\gamma)}$, while a random value $CT_3 \in \{0, 1\}^{1/2}$ is chosen. Finally, the challenge ciphertext $CT^* = (C, CT_1^*, CT_2^*, CT_3^*, CT_4^*)$ is sent to $a_4$.

(2) **Phase 2:** $a_4$ can issue the same query as the phase 1, but it must be under the condition of $ID \neq ID^*$ and $CT \neq CT^*$.

(3) **Guess:** $a_4$ responds to $B$ with a guess $M' \in \{0, 1\}^\lambda$. If $M' \neq M$, $B$ responds with failure and terminates. Otherwise, $a_4$ wins the game. Then $B$ randomly
selects a tuple \((W^*, CT_1^t, CT_2^t, \omega^*)\) from the List\(H_5\), and outputs the BDH solution \(A = (W^* \tilde{\epsilon}(g_1^{\omega^*}, g_2)^{W}g_2^{-1})^{e^{-1}}\) due to \(\tilde{\epsilon}(g_1, g_2)^{abc} = (W^* \tilde{\epsilon}(g_1^{\omega^*}, g_2)^{W})^{e^{-1}}\).

The security analysis is similar to Theorem 5. We obtain that \(\mathcal{B}'\)'s advantage to solve the BDH problem is \(e' \geq (1/q_{H_5}) \Pr[E_{H_5}'] \geq (1/q_{H_5})[(\epsilon - (1/2^k))/\epsilon(q_{TK} + 1)] - (q_{ID}/q)\). \(\square\)

**Theorem 9.** The proposed RIBEET scheme is secure for brute force attacks if the discrete logarithm problem is hard.

**Proof.** As mentioned in the concrete RIBEET scheme, the public system parameters are \(\{q, G_1, G_2, G_T, \tilde{\epsilon}, g_1, g_2, P_{pub}, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8\}\), and the master private key is \(mpk = s\) form \(P_{pub} = g_1^s\). In addition, the security of the user initial key \(IK_{ID}\) and user time key \(TK_{ID}\) is also based on the discrete logarithm problem due to \(IK_{ID} = (IK_{ID1}, IK_{ID2}) = (H_1(\text{ID})^{mpk}, H_2(\text{ID})^{mpk}) = (H_1(\text{ID})^t, H_2(\text{ID})^t)\) and \(TK_{ID} = (TK_{ID1}, TK_{ID2}) = (H_3(\text{ID}, T)^{mpk}, H_4(\text{ID}, T)^{mpk}) = (H_3(\text{ID}, T)^{t}, H_4(\text{ID}, T)^{t})\). Hence, the proposed RIBEET scheme can resist brute force attacks. \(\square\)

## 6. Comparison

In this section, we compare the proposed RIBEET scheme with the previous RIBE scheme [21] and IBEET scheme [6]. In order to analyze the cost of performing encryption, decryption and equality test, we first define two notations as follows.

1. (1) Pair: time to perform a bilinear pairing \(\tilde{\epsilon} : G_1 \times G_2 \rightarrow G_T\)

2. (2) Exp: time to perform an exponentiation in \(G_1, G_2\) or \(G_T\)

We gain Pair = 7.8351 ms and Exp = 0.4746 ms from the literature [32]. These two execution times are obtained under the hardware device with Intel Core i7-8550U 1.80 GHz processor. Meanwhile, the prime number \(q\) selected in the cryptosystem setting phase is 256-bit. In addition, three multiplicative cyclic groups \(G_1, G_2, G_T\) of the prime order \(q\) are chosen in the simulation.

In Table 3, we list the comparisons of our proposed RIBEET scheme with the RIBE scheme [21] and several IBEET schemes [6, 10, 11] in terms of the cost of performing encryption, decryption and equality test, and two properties related to user revocation and equality test of ciphertexts. For the cost of performing encryption and decryption, Tseng and Tsai’s RIBE scheme [21] has better performance than the other two schemes. However, Tseng and Tsai’s RIBE scheme does not support equality test of ciphertexts. Although the existing IBEET schemes [6, 10, 11] and our proposed RIBEET scheme support equality test of ciphertexts, the IBEET schemes does not have a mechanism to revoke users. Conversely, our proposed RIBEET scheme not only provides user revocation, but also retains the cost of encryption, decryption and equality test with the existing IBEET schemes. Additionally, Table 4 compares our RIBEET scheme with the RIBE scheme [21] and several IBEET schemes [6, 10, 11] in terms of \(|PK|, |CT|\), and \(|TD|\) which are, respectively, denoted as the bit length of user public key, ciphertext and trapdoor. We observed that the communication cost of our RIBEET scheme is similar to that of other schemes.

As mentioned in Section 1, the data collected from sensors on the patients is finally encrypted by the mobile device and then transmitted to the cloud. For the analysis of energy cost, we employ the ”ampere” app to measure the voltage and current on the mobile device. After running this app, we obtain 14.28 V and 2856 mA on the mobile device. Table 5 lists the energy cost of performing encryption on the mobile device by using the formula \(W = U \cdot I \cdot t\), where \(W, U, I,\) and \(t\), respectively, are watt, voltage, current, and time.

## 7. Conclusions

We considered the existing syntax of IBEET and the property of user revocation to present the new syntax of RIBEET. Under the new syntax, we proposed a concrete RIBEET scheme. Meanwhile, we demonstrated that the proposed scheme has the IND-ID-CCA security for type I and II adversaries and the OW-ID-CCA security for type III and IV adversaries. We compared the proposed scheme with the previous RIBE scheme and IBEET scheme. We showed that the proposed scheme not only supports equality test for ciphertexts but also provides user revocation.

## Data Availability

The data used to support the findings of this study are included within the article.

## Conflicts of Interest

The authors declare no conflicts of interest.

## Acknowledgments

This research was partially supported by the Ministry of Science and Technology, Taiwan, under contract nos. MOST 110-2222-E-019-001-MY2 and MOST 110-2221-E-019-041-MY3.

## References

[1] Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan, ”Public-key encryption schemes with auxiliary inputs,” in *Theory of Cryptography*, pp. 361–381, Springer, Berlin, Heidelberg, 2010.
[2] D. Hofheinz and T. Jager, “Tightly secure signatures and public-key encryption,” in Advances in Cryptology–CRYPTO 2012, pp. 590–607, Springer, Berlin, Heidelberg, 2012.

[3] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, “Server-aided public key encryption with keyword search,” IEEE Transactions on Information Forensics and Security, vol. 11, no. 12, pp. 2833–2842, 2016.

[4] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, “Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks,” IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3712–3723, 2018.

[5] D. Boneh and M. Franklin, “Identity-based encryption from the Weil pairing,” in Advances in Cryptology–CRYPTO 2001, pp. 213–229, Springer, Berlin, Heidelberg, 2001.

[6] S. Ma, “Identity-based encryption with outsourced equality test in cloud computing,” Information Sciences, vol. 328, pp. 389–402, 2016.

[7] X. J. Lin, L. Sun, and H. Qu, “Generic construction of public key encryption, identity-based encryption and signcryption with equality test,” Information Sciences, vol. 453, pp. 111–126, 2018.

[8] H. T. Lee, H. Wang, and K. Zhang, “Security analysis and modification of ID-based encryption with equality test from ACISP,” in Information Security and Privacy, pp. 780–786, Springer, Cham, 2018.

[9] D. H. Dung, H. Q. Le, P. S. Roy, and W. Susilo, “Lattice-based IBE with equality test in standard model,” in Provable Security, pp. 19–40, Springer, Cham, 2019.

[10] R. Elhabob, Y. Zhao, N. Eltayieb, A. M. Abdelgader, and H. Xiong, “Identity-based encryption with authorized equivalence test for cloud-assisted IoT,” Cluster Computing, vol. 23, no. 2, pp. 1085–1101, 2020.

[11] X. J. Lin, Q. Wang, L. Sun, and H. Qu, “Identity-based encryption with equality test and datestamp-based authorization mechanism,” Theoretical Computer Science, vol. 861, pp. 117–132, 2021.

[12] H. Alemdar and C. Ersoy, “Wireless sensor networks for healthcare: a survey,” Computer Networks, vol. 54, no. 15, pp. 2688–2710, 2010.

[13] B. Latr, B. Braem, I. Moerman, C. Blondia, and P. Demeester, “A survey on wireless body area networks,” Wireless Networks, vol. 17, no. 1, pp. 1–8, 2011.

[14] J. Zhang, H. Nian, X. Ye, X. Ji, and Y. Heg, “A spatial correlation based partial coverage scheduling scheme in wireless sensor networks,” Journal of Network Intelligence, vol. 5, no. 2, pp. 34–43, 2020.

[15] C. H. Hsieh, J. Lin, C. M. Yu, M. H. Hung, and F. Huang, “A TSP-over-LEACH protocol for energy-efficient wireless sensor networks,” Journal of Network Intelligence, vol. 6, no. 4, pp. 835–846, 2021.

[16] C. M. Chen, Z. Li, S. A. Chaudhry, and L. Li, “Attacks and solutions for a two-factor authentication protocol for wireless body area networks,” Security and Communication Networks, vol. 2021, 12 pages, 2021.

[17] H. Xiong, Y. Hou, X. Huang, Y. Zhao, and C. M. Chen, “Heterogeneous signcryption scheme from IBC to PKI with equality test for WBANs,” IEEE Systems Journal, vol. 16, no. 2, pp. 2391–2400, 2021.

[18] R. Housley, W. Polk, W. Ford, and D. Solo, “Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile,” 2002, IETF RFC 3280.

[19] A. Boldyreva, V. Goyal, and V. Kumar, “Identity-based encryption with efficient revocation,” in Proceedings of the 15th ACM conference on Computer and communications security (CCS ’08), pp. 417–426, Alexandria, Virginia, USA, 2008.

[20] B. Libert and D. Vergnaud, “Adaptive-ID secure revocable identity-based encryption,” in Topics in Cryptology–CT-RSA 2009, pp. 1–15, Springer, Berlin, Heidelberg, 2009.

[21] Y. M. Tseng and T. T. Tsai, “Efficient revocable ID-based encryption with a public channel,” Computer Journal, vol. 55, no. 4, pp. 475–486, 2012.

[22] J. H. Seo and K. Emura, “Revocable identity-based encryption revisited: security model and construction,” in Public-Key Cryptography–PKC 2013, pp. 216–234, Springer, Berlin, Heidelberg, 2013.

[23] Y. Watanabe, K. Emura, and J. H. Seo, “New revocable IBE in prime-order groups: adaptively secure, decryption key exposure resistant, and with short public parameters,” in Topics in Cryptology–CT-RSA 2017, pp. 432–449, Springer, Cham, 2017.

[24] A. Takayasu and Y. Watanabe, “Lattice-based revocable identity-based encryption with bounded decryption key exposure resistance,” in Information Security and Privacy, pp. 184–204, Springer, Cham, 2017.

[25] S. Katsumata, T. Matsuda, and A. Takayasu, “Lattice-based revocable (hierarchical) IBE with decryption key exposure resistance,” in Public-Key Cryptography–PKC 2019, pp. 441–471, Springer, Cham, 2019.

[26] A. Takayasu, “Adaptively secure lattice-based revocable IBE in the QROM: compact parameters, tight security, and anonymity,” Designs, Codes and Cryptography, vol. 89, no. 8, pp. 1965–1992, 2021.

[27] J. Baek, R. Safavi-Naini, and W. Susilo, “Public key encryption with keyword search revisited,” in Computational Science and Its Applications–ICCSA 2008, pp. 1249–1259, Springer, Berlin, Heidelberg, 2008.

[28] S. T. Hsu, C. C. Yang, and M. S. Hwang, “A study of public key encryption with keyword search,” International Journal of Network Security, vol. 15, no. 2, pp. 71–79, 2013.

[29] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong, “Probabilistic public key encryption with equality test,” in Topics in Cryptology–CT-RSA 2010, pp. 119–131, Springer, Berlin, Heidelberg, 2010.

[30] A. Shamir, “Identity-based cryptosystems and signature schemes,” in Advances in Cryptology, pp. 47–53, Springer, Berlin, Heidelberg, 1985.

[31] H. Li, Q. Huang, S. Ma, J. Shen, and W. Susilo, “Authorized equality test on identity-based ciphertexts for secret data sharing via cloud storage,” IEEE Access, vol. 7, pp. 25409–25421, 2019.

[32] Y. Li, Q. Cheng, X. Liu, and X. Li, “A secure anonymous identity-based scheme in new authentication architecture for mobile edge computing,” IEEE Systems Journal, vol. 15, no. 1, pp. 935–946, 2021.