Security Analysis and Modification of ID-Based Encryption with Equality Test from ACISP 2017

Hyung Tae Lee[1], Huaxiong Wang[2], and Kai Zhang[34]

[1] Division of Computer Science and Engineering,
College of Engineering, Chonbuk National University, Republic of Korea
hyungtaelee@chonbuk.ac.kr
[2] Division of Mathematical Sciences, School of Physical and Mathemathcial Sciences,
Nanyang Technological University, Singapore
hxwang@ntu.edu.sg
[3] Department of Information Security, College of Computer Science and Technology,
Shanghai University of Electric Power, China
kzhang@shiep.edu.cn
[4] Co-Innovation Center for Information Supply & Assurance Technology,
Anhui University, Hefei, China

Abstract. At ACISP 2017, Wu et al. presented an identity-based encryption with equality test (IBEET) that considers to prevent insider attacks. To analyze its security, they proposed a new security notion for IBEET, which is slightly weaker than the indistinguishability under adaptive identity and chosen ciphertext attacks (IND-ID-CCA2) for traditional identity-based encryption. Then, they claimed that their proposed scheme achieves this new security notion under the Bilinear Diffie-Hellman (BDH) assumption in the random oracle model. In this paper, we demonstrate that their scheme does not achieve the claimed security requirement by presenting an attack. Our attack algorithm is very simple: It requires only a pair of message and ciphertext, and takes one exponentiation and two bilinear map evaluations. Subsequently, we present a modification of their IBEET construction and show that it satisfies their security notion under the BDH assumption and the existence of strong pseudorandom permutation and existentially unforgeable message authentication code in the random oracle model. We remark that our modification has better efficiency than the original construction.

Keywords: Identity based encryption with equality test, insider attacks, chosen ciphertext attacks, modification

1 Introduction

Identity-based encryption with equality test (IBEET) is a special kind of identity-based encryption (IBE) that allows to perform equality tests between ciphertexts under different identities as well as the same identity. More specifically, an IBEET system consists of a sender(s), a receiver(s), and a tester(s): A sender encrypts a message using a receiver's identity and delivers a generated ciphertext to the receiver. The receiver may decrypt the ciphertext using his/her secret key and/or store it at the server. Once a need arises, the receiver issues a trapdoor for equality tests to a tester. Since then, the tester can perform equality tests on ciphertexts under identities of receivers who already passed trapdoors to the tester. This feature enables us to apply IBEET to various scenarios in practice, such as keyword search on encrypted databases and efficient encrypted data management on the cloud. Due to wide availability in practice, several IBEET constructions [2–4, 6] have been proposed.

On the other hand, supporting equality tests makes the security of IBEET schemes weaken. Since the tester can have a trapdoor for equality test on the target ciphertext, she can generate a ciphertext of any message by herself and perform equality tests between the target ciphertext and the ciphertext generated by herself. We call this type of attacks *insider attack* [7]. Due to insider attacks, we cannot expect that IBEET schemes achieve indistinguishability-based security notions against testers who aim to distinguish whether the challenge ciphertext contains which message between two candidates. Particularly, if the message space is sufficiently small or the min-entropy of message distribution is not as high as the security parameter, the tester can recover a message from the target ciphertext by executing insider attacks. Therefore, to prevent insider attacks, in the previous IBEET schemes, it is assumed that the size of message space is exponential in the security parameter and the min-entropy of message distribution is as high as the security parameter.

Very recently, at ACISP 2017, Wu et al. [7] proposed an IBEET scheme which considers to prevent insider attacks. To this end, they first established a variant of traditional IBEET model: In their IBEET system, anyone can perform equality tests between any two ciphertexts publicly without trapdoors. Instead, only group members who have a token for a receiver's identity can generate a ciphertext. Hence, testers who do not have a token cannot perform insider attacks. Thereafter, they constructed an IBEET scheme using bilinear map groups under the proposed model. To analyze the security of their scheme, they introduced a new security notion, which is slightly weaker than the indistinguishability under adaptive identity and chosen ciphertext attacks (IND-ID-CCA2) for traditional IBE; a main difference between two security models is that messages m_0, m_1 submitted by the adversary at the challenge phase cannot be queried to the encryption oracle before and after the challenge phase in the security game for the new model. (Note that the challenger in the security game for the new model should provide an encryption oracle to the adversary because he does not have a token required for encryption, whereas the adversary for the traditional security model of IBE can encrypt a message by himself.) They claimed that their scheme achieves this new security notion under the Bilinear Diffie-Hellman (BDH) assumption in the random oracle model.

In this paper, we demonstrate that their construction does not satisfy their security notion by presenting an attack. Our attack algorithm is very simple: Once the adversary has the challenge ciphertext and a pair of message and ciphertext after the challenge phase, he generates a valid part for equality test of submitted messages at the challenge phase by manipulating the received ciphertext. Then, he can distinguish which message is contained in the challenge ciphertext between two candidates by performing an equality test between the challenge ciphertext and the ciphertext manipulated by himself. It takes one exponentiation to manipulate a ciphertext to obtain a valid part for equality test and two bilinear map evaluations to perform an equality test. Furthermore, our attack requires one access to the encryption oracle only. Hence, it also seems hard that the Wu et al.'s original construction achieves more weaker indistinguishability-based security notions, e.g., the indistinguishability under chosen plaintext attacks and known plaintext attacks.

Next, we modify Wu et al.'s construction so that it achieves the security notion which was presented in the original paper [7]. To avoid our attack presented in this paper, we exploit a keyed permutation, and let group users share the same key for the exploited

keyed permutation and use it as a token for encryption. Moreover, we also employ a message authentication code (MAC) to prevent an adversary from reusing an output of the exploited keyed permutation by manipulating other parts. As a result, we obtain a modification that achieves Wu et al.'s security notion if the exploited keyed permutation is strong pseudorandom, the employed MAC is existentially unforgeable, and the BDH assumption holds in the random oracle model. We remark that our modification has better efficiency than the original construction in terms of computational costs and ciphertext sizes, as a by-product of exploiting a keyed permutation, instead of bilinear maps for the test algorithm.

Organization of the Paper. In Section 2, we introduce formal definitions for IBEET system against insider attack, proposed by Wu et al. and give a description of their construction in [7]. Section 3 presents our attack algorithm for their IBEET scheme. Our modification and its security analysis are provided and discussed in Section 4. The details of security proof are given in Appendix.

2 ID-Based Encryption with Equality Test Against Insider Attack

In this section, we introduce formal definitions for IBEET system against insider attack, proposed by Wu et al. [7]. Then, we review the description of their IBEET construction presented in [7].

Notations. For an algorithm A, $a \leftarrow A$ denotes that a is an output of A. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for all polynomials $p(\cdot)$ and sufficiently large λ, $f(\lambda) \leq \frac{1}{p(\lambda)}$.

2.1 System for IBEET Against Insider Attack

Now, we look at the IBEET system model that considers to prevent insider attack, proposed by Wu et al. [7]. The IBEET system against insider attack consists of a group of users, tester(s), and other users outside of the group. Designated senders who have a token for a receiver in the group can encrypt a message under the receiver's identity and sends it to the receiver. Testers and other users can not generate ciphertexts of users in the group and only can conduct equality tests between ciphertexts of users in the group. Note that Wu et al.'s system model regards testers and other users outside of the group as insiders and attempts to prevent their attacks. Their model does not consider the security against other users in the group who have a token for encryption under the target receiver's identity, but do not generate the target ciphertext.

The IBEET scheme against insider attack consists of the following five polynomial-time algorithms:

- $\mathsf{Setup}(\lambda)$: It takes a security parameter λ as an input and returns the system public parameter PP, the master secret key MSK, and the master token key MTK.
  We note that all other algorithms take PP as an input, though it is not explicitly stated.
- $\mathsf{Extract}(\mathsf{ID}, \mathrm{MSK}, \mathrm{MTK})$: It takes an identity ID, the master secret key MSK, and the master token key MTK as inputs and returns the private key d_{ID} and token $\mathsf{tok}_{\mathsf{ID}}$ for identity ID.

It is assumed that d_{ID} and $\mathsf{tok}_{\mathsf{ID}}$ are delivered to the user of identity ID and all group users, respectively, via secure channel.

- $\mathsf{Enc}(\mathrm{PP}, m, \mathsf{ID}, \mathsf{tok}_{\mathsf{ID}})$: It takes the system public parameter PP, a message m, an identity ID and the token $\mathsf{tok}_{\mathsf{ID}}$ for identity ID as inputs and returns a ciphertext CT.
- $\mathsf{Test}(\mathrm{CT}_A, \mathrm{CT}_B)$: It takes two ciphertexts CT_A and CT_B for identities ID_A and ID_B, respectively, as inputs and returns 1 which indicates that CT_A and CT_B contain the same message, or 0 which indicates that they contain different messages.
- $\mathsf{Dec}(\mathrm{CT}, d_{\mathsf{ID}}, \mathsf{tok}_{\mathsf{ID}})$: It takes a ciphertext CT, the decryption key d_{ID}, and the token $\mathsf{tok}_{\mathsf{ID}}$ for identity ID as inputs and returns a message m or $\perp$.

We say that the above IBEET scheme is *correct* if it satisfies the following conditions:

1. For any security parameter λ, identity ID, and message m, it holds that

$$\Pr[m \leftarrow \mathsf{Dec}(\mathrm{CT}, d_{\mathsf{ID}}, \mathsf{tok}_{\mathsf{ID}})] = 1$$

where $(\mathrm{PP}, \mathrm{MSK}, \mathrm{MTK}) \leftarrow \mathsf{Setup}(\lambda)$, $(d_{\mathsf{ID}}, \mathsf{tok}_{\mathsf{ID}}) \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathrm{MSK}, \mathrm{MTK})$, and $\mathrm{CT} \leftarrow \mathsf{Enc}(\mathrm{PP}, m, \mathsf{ID}, \mathsf{tok}_{\mathsf{ID}})$.

2. For any security parameter λ, identities $\mathsf{ID}_A, \mathsf{ID}_B$, and messages m_A, m_B, it holds that

$$\Pr[1 \leftarrow \mathsf{Test}(\mathrm{CT}_A, \mathrm{CT}_B)]$$

is 1 if $m_A = m_B$ and negligible in the security parameter λ if $m_A \neq m_B$, where $(\mathrm{PP}, \mathrm{MSK}, \mathrm{MTK}) \leftarrow \mathsf{Setup}(\lambda)$, $(d_{\mathsf{ID}_A}, \mathsf{tok}_{\mathsf{ID}_A}) \leftarrow \mathsf{Extract}(\mathsf{ID}_A, \mathrm{MSK}, \mathrm{MTK})$, $(d_{\mathsf{ID}_B}, \mathsf{tok}_{\mathsf{ID}_B}) \leftarrow \mathsf{Extract}(\mathsf{ID}_B, \mathrm{MSK}, \mathrm{MTK})$, $\mathrm{CT}_A \leftarrow \mathsf{Enc}(\mathrm{PP}, m_A, \mathsf{ID}_A, \mathsf{tok}_{\mathsf{ID}_A})$, and $\mathrm{CT}_B \leftarrow \mathsf{Enc}(\mathrm{PP}, m_B, \mathsf{ID}_B, \mathsf{tok}_{\mathsf{ID}_B})$.

We note that the first condition is for the correctness of the decryption algorithm, whereas the second condition is for the correctness of the test algorithm.

2.2 Security Definition for IBEET Against Insider Attack

The authors of [7] introduced a new security notion for IBEET, which is slightly weaker than the formal IND-ID-CCA2 security for traditional IBE. Informally, a main difference between their new security notion and traditional IND-ID-CCA2 security for IBE is that messages m_0, m_1 submitted by the adversary at the challenge phase were never queried to the encryption oracle before the challenge phase and should not be queried to the encryption oracle after the challenge phase in the security game. (In fact, an adversary can generate a ciphertext by himself and thus the encryption oracle does not exist in the security game for traditional IBE schemes.)

The formal definition for their security notion is as follows. We say that an IBEET scheme $\mathsf{IBEET} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Test}, \mathsf{Dec})$ achieves the weak indistinguishability under adaptive identity and chosen ciphertext attacks ($\mathsf{wIND\text{-}ID\text{-}CCA2}$) if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, its advantage in the following game played with the challenger $\mathcal{C}$ is negligible in the security parameter λ:

1. **Setup:** $\mathcal{C}$ obtains the system public parameter PP, the master secret key MSK, and the master token key MTK by running $(\mathrm{PP}, \mathrm{MSK}, \mathrm{MTK}) \leftarrow \mathsf{Setup}(\lambda)$. $\mathcal{C}$ passes PP to $\mathcal{A}$.

2. **Phase 1**: $\mathcal{A}$ may issue queries to the following oracles adaptively and polynomially many times in any order:
   - $\mathcal{O}^{\mathsf{Extract}(\cdot)}$: On input ID_i, the key extraction oracle returns d_{ID_i}, where $(d_{\mathsf{ID}_i}, \mathsf{tok}_{\mathsf{ID}_i}) \leftarrow \mathsf{Extract}(\mathsf{ID}_i, \mathrm{MSK}, \mathrm{MTK})$.
   - $\mathcal{O}^{\mathsf{Enc}(\cdot,\cdot)}$: On input a pair of message and identity $(m_{\mathsf{ID}_i}, \mathsf{ID}_i)$, the encryption oracle returns $\mathrm{CT}_{\mathsf{ID}_i}$ for $\mathrm{CT}_{\mathsf{ID}_i} \leftarrow \mathsf{Enc}(\mathrm{PP}, m_{\mathsf{ID}_i}, \mathsf{ID}_i, \mathsf{tok}_{\mathsf{ID}_i})$.
   - $\mathcal{O}^{\mathsf{Dec}(\cdot,\cdot)}$: On input a pair of identity and ciphertext $(\mathsf{ID}_i, \mathrm{CT}_{\mathsf{ID}_i})$, the decryption oracle returns m by running $m \leftarrow \mathsf{Dec}(\mathrm{CT}_{\mathsf{ID}_i}, d_{\mathsf{ID}_i}, \mathsf{tok}_{\mathsf{ID}_i})$ for $(d_{\mathsf{ID}_i}, \mathsf{tok}_{\mathsf{ID}_i}) \leftarrow \mathsf{Extract}(\mathsf{ID}_i, \mathrm{MSK}, \mathrm{MTK})$.
3. **Challenge**: $\mathcal{A}$ submits a target identity ID^* and two messages m_0, m_1 of the same-length to $\mathcal{C}$, where ID^* was never queried to $\mathcal{O}^{\mathsf{Extract}(\cdot)}$ and m_0, m_1 were never queried to $\mathcal{O}^{\mathsf{Enc}(\cdot,\cdot)}$ in **Phase 1**. $\mathcal{C}$ picks a random bit $b \in \{0,1\}$, runs $\mathrm{CT}^*_{\mathsf{ID}^*,b} \leftarrow \mathsf{Enc}(\mathrm{PP}, m_b, \mathsf{ID}^*, \mathsf{tok}_{\mathsf{ID}^*})$, and sends $\mathrm{CT}^*_{\mathsf{ID}^*,b}$ to $\mathcal{A}$.
4. **Phase 2:** As in **Phase 1**, $\mathcal{A}$ may issue queries to the oracles adaptively and polynomially many times in any order. The constraints for $\mathcal{A}$'s queries are as follows:
   (a) The challenge identity ID^* cannot be queried to $\mathcal{O}^{\mathsf{Extract}(\cdot)}$.
   (b) The submitted messages m_0, m_1 cannot be queried to $\mathcal{O}^{\mathsf{Enc}(\cdot,\cdot)}$.
   (c) The pair of the challenge identity and ciphertext $(\mathsf{ID}^*, \mathrm{CT}^*_{\mathsf{ID}^*,b})$ cannot be queried to $\mathcal{O}^{\mathsf{Dec}(\cdot,\cdot)}$.
5. **Guess:** $\mathcal{A}$ returns a random guess b'.

The advantage of $\mathcal{A}$ in the above game is defined to $\mathsf{Adv}^{\mathsf{wIND\text{-}ID\text{-}CCA2}}_{\mathcal{A},\mathsf{IBEET}}(\lambda) := \left|\Pr[b' = b] - \frac{1}{2}\right|$.

Remark 1. The extraction oracle $\mathcal{O}^{\mathsf{Extract}(\cdot)}$ does not return $\mathsf{tok}_{\mathsf{ID}_i}$ though it executes the extraction algorithm. This is because we assume that the adversary cannot generate ciphertexts of group users. Otherwise, he can generate ciphertexts of messages m_0 and m_1 and perform equality tests by himself and thus it is impossible to achieve wIND-ID-CCA2 security inherently.

2.3 Wu et al.'s IBEET Scheme

In this subsection, we review Wu et al.'s IBEET scheme [7]. The description of their IBEET construction is as follows.

- $\mathsf{Setup}(\lambda)$: On input a security parameter λ, generate two multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order $p = p(\lambda)$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Pick a random generator g of $\mathbb{G}_1$. Select two random elements α, β from $\mathbb{Z}_p^*$, and set a master secret key MSK and a master token key MTK as

$$\mathrm{MSK} = \alpha \quad \text{and} \quad \mathrm{MTK} = \beta.$$

  Compute $P_{pub} = g^\alpha$. Generate three cryptographic hash functions

$$\mathsf{H} : \{0,1\}^t \to \mathbb{Z}_p^*, \quad \mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1, \text{ and } \mathsf{H}_2 : \mathbb{G}_1^3 \times \mathbb{G}_2 \to \{0,1\}^{t+\ell},$$

where t denotes the bit-length of messages and ℓ denotes the bit-length of randomness utilized in the encryption algorithm, i.e., $\ell = \lceil \log_2 p \rceil$ where $\lceil a \rceil$ denotes the smallest integer that is larger than or equal to a for $a \in \mathbb{R}$. Finally, output a system public parameter

$$\text{PP} = (\lambda, p, t, \ell, g, \mathbb{G}_1, \mathbb{G}_2, P_{pub}, e, \mathsf{H}, \mathsf{H}_1, \mathsf{H}_2)$$

and a pair of the master secret and master token keys (MSK, MTK).

– $\mathsf{Extract}(\mathsf{ID}, \text{MSK}, \text{MTK})$: On input an identity ID, the master secret key $\text{MSK} = \alpha$, and the master token key $\text{MTK} = \beta$, the key generation center (KGC) computes

$$g_{\mathsf{ID}} = \mathsf{H}_1(\mathsf{ID}), \quad d_{\mathsf{ID}} = g_{\mathsf{ID}}^{\alpha} \text{ and } \mathsf{tok}_{\mathsf{ID}} = g_{\mathsf{ID}}^{\beta},$$

and outputs $(d_{\mathsf{ID}}, \mathsf{tok}_{\mathsf{ID}})$.

– $\mathsf{Enc}(\text{PP}, m, \mathsf{ID}, \mathsf{tok}_{\mathsf{ID}})$: It takes the system public parameter PP, a message m, an identity ID, and the token $\mathsf{tok}_{\mathsf{ID}}$ for identity ID as inputs and picks two random elements r_1, r_2 from $\mathbb{Z}_p^*$. Then, it computes

$$C_1 = \mathsf{tok}_{\mathsf{ID}}^{r_1 \mathsf{H}(m)}, \quad C_2 = g_{\mathsf{ID}}^{r_1}, \quad C_3 = g^{r_2},$$
$$C_4 = (m \| r_1) \oplus \mathsf{H}_2(C_1 \| C_2 \| C_3 \| e(P_{pub}, g_{\mathsf{ID}})^{r_2})$$

where $g_{\mathsf{ID}} = \mathsf{H}_1(\mathsf{ID})$. Finally, it outputs a ciphertext $\text{CT} = (C_1, C_2, C_3, C_4)$.

– $\mathsf{Test}(\text{CT}_A, \text{CT}_B)$: It takes two ciphertexts $\text{CT}_A = (C_{A,1}, C_{A,2}, C_{A,3}, C_{A,4})$ and $\text{CT}_B = (C_{B,1}, C_{B,2}, C_{B,3}, C_{B,4})$ for identities ID_A and ID_B, respectively, as inputs. Check whether

$$e(C_{A,1}, C_{B,2}) = e(C_{B,1}, C_{A,2}). \tag{1}$$

If it holds, output 1. Otherwise, output 0.

– $\mathsf{Dec}(\text{CT}, d_{\mathsf{ID}}, \mathsf{tok}_{\mathsf{ID}})$: It takes a ciphertext $\text{CT} = (C_1, C_2, C_3, C_4)$, a decryption key d_{ID} and a token $\mathsf{tok}_{\mathsf{ID}}$ for user ID as inputs and computes

$$m' \| r_1' = C_4 \oplus \mathsf{H}_2(C_1 \| C_2 \| C_3 \| e(C_3, d_{\mathsf{ID}})).$$

Then, check whether

$$C_1 = \mathsf{tok}_{\mathsf{ID}}^{r_1' \mathsf{H}(m')} \text{ and } C_2 = g_{\mathsf{ID}}^{r_1'}.$$

where $g_{\mathsf{ID}} = \mathsf{H}_1(\mathsf{ID})$. If both hold, return m'. Otherwise, return $\perp$.

Because our attack algorithm mainly exploits the test algorithm, we specially look into the correctness of the test algorithm. Let CT_A and CT_B be valid ciphertexts of messages m_A and m_B under identities ID_A and ID_B, respectively. That is,

$$\begin{aligned}\text{CT}_A &= (C_{A,1}, C_{A,2}, C_{A,3}, C_{A,4}) \\ &= (\mathsf{tok}_{\mathsf{ID}_A}^{r_{A,1} \mathsf{H}(m_A)}, g_{\mathsf{ID}_A}^{r_{A,1}}, g^{r_{A,2}}, (m_A \| r_{A,1}) \oplus \mathsf{H}_2(C_{A,1} \| C_{A,2} \| C_{A,3} \| e(P_{pub}, g_{\mathsf{ID}_A})^{r_{A,2}}))\end{aligned}$$

and

$$\begin{aligned}\text{CT}_B &= (C_{B,1}, C_{B,2}, C_{B,3}, C_{B,4}) \\ &= (\mathsf{tok}_{\mathsf{ID}_B}^{r_{B,1} \mathsf{H}(m_B)}, g_{\mathsf{ID}_B}^{r_{B,1}}, g^{r_{B,2}}, (m_B \| r_{B,1}) \oplus \mathsf{H}_2(C_{B,1} \| C_{B,2} \| C_{B,3} \| e(P_{pub}, g_{\mathsf{ID}_B})^{r_{B,2}}))\end{aligned}$$

for randomness $r_{A,1}, r_{A,2}, r_{B,1}, r_{B,2} \in \mathbb{Z}_p^*$ chosen by the encryption algorithm. Thus,

$$e(C_{A,1}, C_{B,2}) = e(\mathsf{tok}_{\mathsf{ID}_A}^{r_{A,1}\mathsf{H}(m_A)}, g_{\mathsf{ID}_B}^{r_{B,1}}) = e(\mathsf{tok}_{\mathsf{ID}_A}, g_{\mathsf{ID}_B})^{r_{A,1}r_{B,1}\mathsf{H}(m_A)} \tag{2}$$

and

$$e(C_{B,1}, C_{A,2}) = e(\mathsf{tok}_{\mathsf{ID}_B}^{r_{B,1}\mathsf{H}(m_B)}, g_{\mathsf{ID}_A}^{r_{A,1}}) = e(\mathsf{tok}_{\mathsf{ID}_B}, g_{\mathsf{ID}_A})^{r_{A,1}r_{B,1}\mathsf{H}(m_B)}. \tag{3}$$

Since $e(\mathsf{tok}_{\mathsf{ID}_A}, g_{\mathsf{ID}_B}) = e(g_{\mathsf{ID}_A}, g_{\mathsf{ID}_B})^{\beta} = e(\mathsf{tok}_{\mathsf{ID}_B}, g_{\mathsf{ID}_A})$ for the master token key $\mathrm{MTK} = \beta$, Equation (2) is equal to Equation (3) if $m_A = m_B$. Otherwise, they are the same with a negligible probability under assuming that the exploited hash function H is collision-resistant. Therefore, the test algorithm correctly outputs the result of equality test on the input ciphertexts.

3 Our Attack Against Wu et al.'s IBEET Scheme

In this section, we provide our attack algorithm against Wu et al.'s IBEET construction.

Description of Our Attack Algorithm. The description of our attack algorithm is as follows.

1. At **Phase 1**, $\mathcal{A}$ issues an encryption oracle query with a message m and an identity ID. Then, it returns a ciphertext $\mathrm{CT} = (C_1, C_2, C_3, C_4)$ of message m under identity ID such that

$$C_1 = \mathsf{tok}_{\mathsf{ID}}^{r_1\mathsf{H}(m)}, \quad C_2 = g_{\mathsf{ID}}^{r_1}, \quad C_3 = g^{r_2},$$
$$C_4 = (m \| r_1) \oplus \mathsf{H}_2(C_1 \| C_2 \| C_3 \| e(P_{pub}, g_{\mathsf{ID}})^{r_2})$$

   where $r_1, r_2 \in \mathbb{Z}_p^*$ are random elements chosen by the encryption algorithm and $g_{\mathsf{ID}} = \mathsf{H}_1(\mathsf{ID})$.

2. At the challenge phase, $\mathcal{A}$ submits a target identity ID^* and two messages m_0, m_1 of the same-length such that $\mathsf{H}(m_0) \neq \mathsf{H}(m_1)$. Then, $\mathcal{C}$ returns the challenge ciphertext $\mathrm{CT}^*_{\mathsf{ID}^*,b} = (C_1^*, C_2^*, C_3^*, C_4^*)$ such that

$$C_1^* = \mathsf{tok}_{\mathsf{ID}^*}^{r_1^*\mathsf{H}(m_b)}, \quad C_2^* = g_{\mathsf{ID}^*}^{r_1^*}, \quad C_3^* = g^{r_2^*},$$
$$C_4^* = (m_b \| r_1^*) \oplus \mathsf{H}_2(C_1^* \| C_2^* \| C_3^* \| e(P_{pub}, g_{\mathsf{ID}^*})^{r_2^*})$$

   where b is a random bit chosen by $\mathcal{C}$, $r_1^*, r_2^* \in \mathbb{Z}_p^*$ are random elements chosen by the encryption algorithm and $g_{\mathsf{ID}^*} = \mathsf{H}_1(\mathsf{ID}^*)$.

3. Once receiving the challenge ciphertext $\mathrm{CT}^*_{\mathsf{ID}^*,b} = (C_1^*, C_2^*, C_3^*, C_4^*)$ from $\mathcal{C}$, $\mathcal{A}$ first computes

$$C_1' = (C_1^{\mathsf{H}(m)^{-1} \bmod p})^{\mathsf{H}(m_1)} \tag{4}$$

   using the ciphertext $\mathrm{CT} = (C_1, C_2, C_3, C_4)$ of message m obtained at **Phase 1**. Then, $\mathcal{A}$ checks whether

$$e(C_1', C_2^*) \stackrel{?}{=} e(C_1^*, C_2)$$

   If it holds, it returns 1. Otherwise, it returns 0.

Correctness of Our Attack Algorithm. The correctness of our attack algorithm is straightforward. First, from Equation (4), we have

$$C_1' = (C_1^{\mathsf{H}(m)^{-1} \bmod p})^{\mathsf{H}(m_1)} = ((\mathsf{tok}_{\mathsf{ID}}^{r_1\mathsf{H}(m)})^{\mathsf{H}(m)^{-1} \bmod p})^{\mathsf{H}(m_1)} = \mathsf{tok}_{\mathsf{ID}}^{r_1\mathsf{H}(m_1)}.$$

Thus,

$$e(C_1', C_2^*) = (\mathsf{tok}_{\mathsf{ID}}^{r_1\mathsf{H}(m_1)}, g_{\mathsf{ID}^*}^{r_1^*}) = e(g_{\mathsf{ID}}, g_{\mathsf{ID}^*})^{\beta r_1 r_1^* \mathsf{H}(m_1)}$$

since $\mathsf{tok}_{\mathsf{ID}} = g_{\mathsf{ID}}^{\beta}$. On the other hand,

$$e(C_1^*, C_2) = e(\mathsf{tok}_{\mathsf{ID}^*}^{r_1^*\mathsf{H}(m_b)}, g_{\mathsf{ID}}^{r_1}) = e(g_{\mathsf{ID}^*}, g_{\mathsf{ID}})^{\beta r_1 r_1^* \mathsf{H}(m_b)}$$

since $\mathsf{tok}_{\mathsf{ID}^*} = g_{\mathsf{ID}^*}^{\beta}$. Therefore, they are the same if $b = 1$ and different if $b = 0$ and so our attack algorithm outputs the correct answer with probability 1. We note that our attack algorithm succeeds regardless of whether $\mathsf{ID} = \mathsf{ID}^*$ or not.

Notes on Our Attack. Our attack algorithm only requires a pair of message and ciphertext and takes one exponentiation in $\mathbb{G}_1$ and two bilinear map evaluations. Furthermore, it does not need key extraction oracle and decryption oracle queries. Therefore, it seems also hard that Wu et al.'s scheme achieves more weaker indistinguishability-based security notions, such as the indistinguishability under chosen plaintext attacks and known plaintext attacks.

4 Our Modification

In this section, we present our modification of Wu et al.'s IBEET construction. Before providing the description of our modification, we briefly introduce our strategy first. From our observation, a ciphertext of their scheme consists of two parts: One ((C_3, C_4) in the ciphertext) is for recovering a message in the decryption algorithm and the other ((C_1, C_2) in the ciphertext) is for performing equality tests in the test algorithm. To achieve the security requirement for IBEET against insider attack, only group users who have a token should be allowed to generate a valid part for equality tests in ciphertexts. However, in Wu et al.'s construction, anyone can manipulate that part when another pair of message and ciphertext is given, as described in Section 3 of this paper.

To avoid such a situation, we exploit another cryptographic tool, a keyed permutation $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a key space of F and $n = n(\lambda)$ for the security parameter λ. Then, we let group users share the same key $K \in \mathcal{K}$ for F and use it as a token for encryption. More concretely, a ciphertext of message m for identity ID in our modification has a form

$$C_1 = F(K_1, \mathsf{H}(m)),\ C_2 = g^r,\ C_3 = (m\|r) \oplus \mathsf{H}_2(C_1\|C_2\|e(P_{pub}, g_{\mathsf{ID}})^r)$$

where H and H_2 are cryptographic hash functions, g is a generator of the underlying group, e is a bilinear map, and P_{pub} is the master public key, g_{ID} is a hashed value of ID, and r is a randomness chosen by the encryption algorithm. Here, C_1 is for the test algorithm and a pair of (C_2, C_3) is for the decryption algorithm. However, unfortunately, the above

provisional construction is not secure against adaptive chosen ciphertext attacks since the adversary can still generate another valid ciphertext

$$C_1 = F(K_1, \mathsf{H}(m)),\ C_2' = g^s,\ C_3' = (m\|s) \oplus \mathsf{H}_2(C_1\|C_2'\|e(P_{pub}, g_{\mathsf{ID}})^s)$$

by replacing r with s and request a decryption query on it.

To make up for the above issue, we additionally employ a MAC scheme and modify the ciphertext so that it has a form

$$C_1 = F(K_1, \mathsf{H}(m)),\ C_2 = g^r,\ C_3 = (m\|r) \oplus \mathsf{H}_2(T\|C_2\|e(P_{pub}, g_{\mathsf{ID}})^r)$$

where $T \leftarrow \mathsf{S}(K_2, C_1)$ for the signing algorithm S of the employed MAC. For correct encryption and decryption, we add the secret key K_2 to both the master token key MTK and the token for encryption.

Informally, the adversary should generate at least the corresponded tag T or the solution $(e(P_{pub}, g_{\mathsf{ID}})^r)$ to the BDH instance to obtain a valid ciphertext by modifying the given ciphertext. Furthermore, it seems hard for him to obtain a useful information about $F(K_1, \mathsf{H}(m_0))$ or $F(K_1, \mathsf{H}(m_1))$ by manipulating C_1 parts of valid ciphertexts of messages which are not m_0 and m_1. Therefore, our scheme seems secure if F is strong pseudorandom, the exploited MAC is existentially unforgeable, and the BDH assumption holds in the random oracle model. Refer to Section 4.2 and Appendix for the details.

4.1 Description of Our Modification

Building Blocks. We employ a keyed permutation and a MAC for our modification. Their definitions are as follows.

Definition 1 (Keyed Permutation [1]). *Let $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be a length-preserving, keyed function, that is, F is a two input function where the first input is called the key and the second input is called just the input. We say that a keyed function F is a keyed permutation if for every key $k \in \{0,1\}^\kappa$, the function $F_k(\cdot) := F(k, \cdot)$ is one-to-one.*

Definition 2 (Message Authentication Code). *A message authentication code* MAC *consists of the following three polynomial time algorithms:*

- $\mathsf{G}(\lambda)$*: On input a security parameter λ, it returns a secret key K.*
- $\mathsf{S}(K, m)$*: Given the secret key K and a message m, it returns a tag T.*
- $\mathsf{V}(K, m, T)$*: Given the secret key K, a message m, and a tag T, it returns 1 or 0.*

Note that we do not exploit the verification algorithm V in our modification, but we assume that the signing algorithm S is deterministic.

Description of Our Modification. The description of our modification is as follows:

- $\mathsf{Setup}(\lambda)$: It generates parameters p, $\mathbb{G}_1$, $\mathbb{G}_2$, $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, MSK $= \alpha$, and $P_{pub} = g^\alpha$ by the same manner as in Wu et al.'s setup algorithm. Choose a keyed permutation $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ for positive integers $\kappa = \kappa(\lambda)$ and $n = n(\lambda)$. Select a random value K_1 from $\{0,1\}^\kappa$. Generate a MAC scheme $\mathsf{MAC} = (\mathsf{G}, \mathsf{S}, \mathsf{V})$

and obtain K_2 by running $\mathsf{G}(\lambda)$. Set the master token key $\mathrm{MTK} = (K_1, K_2)$. Generate three cryptographic hash functions

$$\mathsf{H} : \{0,1\}^t \to \{0,1\}^n, \quad \mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1, \text{ and } \mathsf{H}_2 : \mathcal{T} \times \mathbb{G}_1 \times \mathbb{G}_2 \to \{0,1\}^{t+\ell},$$

where t denotes the bit-length of messages, ℓ denotes the bit-length of randomness utilized in the encryption algorithm and $\mathcal{T}$ denotes the range of outputs of S. We remark that the image of H and the domain of H_2 are slightly modified from those of the original scheme. Finally, output a system public parameter

$$\mathrm{PP} = (\lambda, p, t, \ell, g, \mathbb{G}_1, \mathbb{G}_2, P_{pub}, e, F, \mathsf{MAC}, \mathsf{H}, \mathsf{H}_1, \mathsf{H}_2)$$

and a pair of the master secret and master token keys $(\mathrm{MSK}, \mathrm{MTK})$.

- $\mathsf{Extract}(\mathsf{ID}, \mathrm{MSK}, \mathrm{MTK})$: While d_{ID} is generated by the same manner as in Wu et al.'s extract algorithm, $\mathsf{tok}_{\mathsf{ID}}$ is set to $\mathrm{MTK} = (K_1, K_2)$, and it outputs $(d_{\mathsf{ID}}, \mathsf{tok}_{\mathsf{ID}})$.
- $\mathsf{Enc}(\mathrm{PP}, m, \mathsf{ID}, \mathsf{tok}_{\mathsf{ID}})$: Given the system public parameter PP, a message m, an identity ID, and the token $\mathsf{tok}_{\mathsf{ID}} = (K_1, K_2)$ for identity ID as inputs, pick a random element r from $\mathbb{Z}_p^*$. Then, it computes

$$C_1 = F_1(K_1, \mathsf{H}(m)),\ C_2 = g^r,\ C_3 = (m\|r) \oplus \mathsf{H}_2(T\|C_2\|e(P_{pub}, g_{\mathsf{ID}})^r) \tag{5}$$

  where $T \leftarrow \mathsf{S}(K_2, C_1)$ and $g_{\mathsf{ID}} = \mathsf{H}_1(\mathsf{ID})$. Finally, it outputs a ciphertext $\mathrm{CT} = (C_1, C_2, C_3)$.
- $\mathsf{Test}(\mathrm{CT}_A, \mathrm{CT}_B)$: On input two ciphertexts $\mathrm{CT}_A = (C_{A,1}, C_{A,2}, C_{A,3})$ and $\mathrm{CT}_B = (C_{B,1}, C_{B,2}, C_{B,3})$ for identities ID_A and ID_B, respectively, check whether $C_{A,1} = C_{B,1}$. If it holds, output 1. Otherwise, output 0.
- $\mathsf{Dec}(\mathrm{CT}, d_{\mathsf{ID}}, \mathsf{tok}_{\mathsf{ID}})$: Given a ciphertext $\mathrm{CT} = (C_1, C_2, C_3)$, a decryption key d_{ID} and a token $\mathsf{tok}_{\mathsf{ID}} = (K_1, K_2)$ for user ID as inputs, compute

$$m'\|r' = C_3 \oplus \mathsf{H}_2(T\|C_2\|e(C_2, d_{\mathsf{ID}})).$$

  where $T \leftarrow \mathsf{S}(K_2, C_1)$. Then, it checks whether $C_1 = F_1(K_1, \mathsf{H}(m'))$ and $C_2 = g^{r'}$. If both hold, return m'. Otherwise, return $\perp$.

Remark 2. While a token $\mathsf{tok}_{\mathsf{ID}}$ is changed per identity ID in the original construction, it is fixed for all group users in our modification. We note that since designated senders should know a receiver's token key to generate a ciphertext, it is not a secret information among all group users. Thus, using the same token key among them does not cause any security issue under Wu et al.'s security model. On the other hand, it enables us to improve the efficiency of our modification by realizing a part of ciphertext for the test algorithm using a keyed permutation.

Correctness of Our Modification. Let $\mathrm{CT} = (C_1, C_2, C_3)$ be a valid ciphertext of message m with respect to identity ID, i.e., it satisfies Equation (5) for some r. Then, for $T \leftarrow \mathsf{S}(K_2, C_1)$ with a deterministic algorithm S,

$$\begin{aligned} m'\|r'_1 &= C_3 \oplus \mathsf{H}_2(T\|C_2\|e(C_2, d_{\mathsf{ID}})) \\ &= (m\|r) \oplus \mathsf{H}_2(T\|C_2\|e(P_{pub}, g_{\mathsf{ID}})^r) \oplus \mathsf{H}_2(T\|C_2\|e(C_2, d_{\mathsf{ID}})) = m\|r \end{aligned}$$

since $e(P_{pub}, g_{\mathsf{ID}})^r = e(g^\alpha, g_{\mathsf{ID}})^r = e(g^r, g_{\mathsf{ID}}^\alpha) = e(C_2, d_{\mathsf{ID}})$. Moreover, it holds both $C_1 = F(K_1, \mathsf{H}(m'))$ and $C_2 = g^{r'}$. Thus, our decryption algorithm returns m correctly.

Suppose that two valid ciphertexts $\mathrm{CT}_A = (C_{A,1}, C_{A,2}, C_{A,3})$ and $\mathrm{CT}_B = (C_{B,1}, C_{B,2}, C_{B,3})$ of messages m_A and m_B for identities ID_A and ID_B, respectively, are given. Then,

$$C_{A,1} = F(K_1, \mathsf{H}(m_A)) \text{ and } C_{B,1} = F(K_1, \mathsf{H}(m_B))$$

and so the test algorithm always outputs 1 if $m_A = m_B$ and outputs 0 if $m_A \neq m_B$ with overwhelming property when the exploited hash function H is collision-resistant. Therefore, our modification is correct.

Efficiency Comparison of Our Modification with the Original Scheme. We now compare the efficiency of our modification and the original construction. Our encryption algorithm replaces two components (C_1, C_2) in the original construction, which are obtained by two exponentiations, with one component C_1, which is an output of a keyed permutation. It also replaces the first component of an input of H_2 by the output of the signing algorithm of the MAC. In general, it is regarded that an evaluation of keyed permutation and an execution of the signing algorithm of MACs much cheaper than an ex ponentiation and so our encryption algorithm is more efficient. This modification also yields to replacing an exponentiation with an evaluation of keyed permutation and an execution of the signing algorithm of the MAC in the decryption algorithm. Finally, whereas the test algorithm of the original construction requires two bilinear map evaluations, ours consists of checking equality between two components. Thus, our test algorithm saves two bilinear map evaluations.

4.2 Security Analysis

Now, we show that our modification is wIND-ID-CCA2 secure under Wu et al.'s security model. To this end, we first introduce three cryptographic assumptions below which will be utilized for our security proof.

Definition 3 (Bilinear Diffie-Hellman Assumption). *Let $\mathbb{G}_1, \mathbb{G}_2$ be two multiplicative cyclic groups of prime order $p = p(\lambda)$ for the security parameter λ. Let g be a generator of $\mathbb{G}_1$ and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. The Bilinear Diffie-Hellman (BDH) problem on $(g, \mathbb{G}_1, \mathbb{G}_2, e)$ is to find $e(g,g)^{xyz}$ where g^x, g^y, g^z are given for randomly chosen x, y, z from $\mathbb{Z}_p^*$. For a PPT adversary $\mathcal{A}$, its advantage in solving the BDH problem on $(g, \mathbb{G}_1, \mathbb{G}_2, e)$ is defined to $\Pr[\mathcal{A}(g^x, g^y, g^z) = e(g,g)^{xyz}]$.*

We say that the BDH assumption holds if for any PPT adversary $\mathcal{A}$, its advantage is negligible in the security parameter λ.

Definition 4 (Strong Pseudorandom Permutation from Definition 3.28 in [1]). *We say that a keyed permutation $F : \{0,1\}^{\kappa(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$ is a strong pseudorandom permutation if for any PPT adversary $\mathcal{A}$, it holds that*

$$\left| \Pr[\mathcal{A}^{F_k(\cdot), F_k^{-1}(\cdot)}(\lambda) = 1] - \Pr[\mathcal{A}^{f(\cdot), f^{-1}(\cdot)}(\lambda) = 1] \right|$$

is negligible in the security parameter λ where k is chosen uniformly at random from $\{0,1\}^{\kappa(\lambda)}$ and f is chosen uniformly at random from the set of permutations on $n(\lambda)$-bits.

We note that a strong pseudorandom permutation can be implemented using a block cipher in practice. See [1, Chapter 5] for details.

Definition 5 (Existential Unforgeability under Chosen Message Attacks). *We say that a message authentication code* $\mathsf{MAC} = (\mathsf{G}, \mathsf{S}, \mathsf{V})$ *is existentially unforgeable under chosen message attack if for any PPT adversary* $\mathcal{A}$ *who can access to the signing oracle* $\mathcal{O}^{\mathsf{S}(K,\cdot)}$,

$$|\Pr[\mathsf{V}(K, m, T) = 1]|$$

is negligible in the security parameter λ *where* $K \leftarrow \mathsf{G}(\lambda)$, (m, T) *is the output of* $\mathcal{A}$ *and* m *is not queried to* $\mathcal{O}^{\mathsf{S}(K,\cdot)}$.

The following theorem shows that our modification is wIND-ID-CCA2 secure in the random oracle model if the BDH assumption holds, the exploited F is a strong pseudorandom permutation and the employed MAC scheme is existentially unforgeable under chosen message attack. We give the detailed proof of the theorem in Appendix.

Theorem 1. *Our modification presented in Section 4.1 is* wIND-ID-CCA2 *secure if the exploited* F *is a strong pseudorandom permutation, the exploited* MAC *is existentially unforgeable under chosen message attack and the BDH assumption holds in the random oracle model.*

Sketch Proof. We prove this theorem by using the standard hybrid argument. To this end, we first define a series of security games below. Let $\textsc{ct}^* = (C_1^*, C_2^*, C_3^*)$ denote the challenge ciphertext in security games.

$\mathbf{Game}_0$: This is equivalent to the original security game described in Section 2.2.

$\mathbf{Game}_1$: This is almost the same as $\mathbf{Game}_0$, except that F is replaced by a truly random permutation f which is chosen uniformly at random from the set of permutations on n-bits and modelled as a random oracle with the constraint that m_0 and m_1 cannot be queried to this random oracle.

$\mathbf{Game}_2$: This is almost the same as $\mathbf{Game}_1$, except that the adversary cannot request decryption queries on valid ciphertexts $\textsc{ct} = (C_1, C_2, C_3)$ such that $C_1 = C_1^*$, but $\textsc{ct} \neq \textsc{ct}^*$.

We first show that the difference between adversarial advantages in $\mathbf{Game}_0$ and $\mathbf{Game}_1$ is negligible in the security parameter if the exploited F is a strong pseudorandom permutation by using the difference lemma [5]. Next, since if the adversary can generate a valid ciphertext $\textsc{ct} = (C_1, C_2, C_3)$ such that $C_1 = C_1^*$, but $\textsc{ct} \neq \textsc{ct}^*$, then the value T which is the outcome of $\mathsf{S}(K_2, f(m_b))$ for the challenge message m_b, should appear at the H_2 oracle query. Using this property, we prove that the difference between adversarial advantages in $\mathbf{Game}_1$ and $\mathbf{Game}_2$ is negligible if the employed MAC scheme is existentially unforgeable under chosen message attack. Finally, we show that the advantage of the adversary in $\mathbf{Game}_2$ is negligible if the BDH assumption holds by constructing a simulator who solves the BDH problem using the adversary in $\mathbf{Game}_2$, which is almost the same as that in the proof of Theorem 1 in [7]. See Appendix for details. □

5 Conclusion

In this paper, we presented an attack on the identity-based encryption scheme with equality test against insider attack, proposed by Wu et al. [7]. Then, we provided a modification of their scheme and showed that it achieves the weak indistinguishability under adaptive identity and chosen ciphertext attacks, which was defined and claimed to be achieved in the original paper.

From our observation, Wu et al.'s original security model does not capture attacks by group members who can generate valid ciphertexts of group members, but did not generate a target ciphertext. It would be an interesting open problem to establish a security model that captures such an attack, which seems stronger than the current one, and to design a scheme satisfying this security model.

References

1. J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2008.
2. H. T. Lee, S. Ling, J. H. Seo, and H. Wang. Semi-generic construction of public key encryption and identity-based encryption with equality test. *Inf. Sci.*, 373:419–440, 2016.
3. H. T. Lee, S. Ling, J. H. Seo, H. Wang, and T. Youn. Public key encryption with equality test in the standard model. *IACR Cryptology ePrint Archive*, 2016:1182, 2016.
4. S. Ma. Identity-based encryption with outsourced equality test in cloud computing. *Inf. Sci.*, 328:389–402, 2016.
5. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.
6. L. Wu, Y. Zhang, K. R. Choo, and D. He. Efficient and secure identity-based encryption scheme with equality test in cloud computing. *Future Generation Comp. Syst.*, 73:22–31, 2017.
7. T. Wu, S. Ma, Y. Mu, and S. Zeng. ID-based encryption with equality test against insider attack. In *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part I*, pages 168–183, 2017.

A Proof of Theorem 1

Theorem 1. *Our modification presented in Section 4.1 is* wIND-ID-CCA2 *secure if the exploited F is a strong pseudorandom permutation, the exploited* MAC *is existentially unforgeable under chosen message attack and the BDH assumption holds in the random oracle model.*

Proof. We prove this theorem by using the standard hybrid argument. To this end, we first define a series of security games below. Let $\mathrm{CT}^* = (C_1^*, C_2^*, C_3^*)$ denote the challenge ciphertext in security games.

$\mathbf{Game}_0$: This is equivalent to the original security game described in Section 2.2.

$\mathbf{Game}_1$: This is almost the same as $\mathbf{Game}_0$, except that F is replaced by a truly random permutation f which is chosen uniformly at random from the set of permutations on n-bits and modelled as a random oracle with the constraint that m_0 and m_1 cannot be queried to this random oracle.

$\mathbf{Game}_2$: This is almost the same as $\mathbf{Game}_1$, except that the adversary cannot request decryption queries on valid ciphertexts $\mathrm{CT} = (C_1, C_2, C_3)$ such that $C_1 = C_1^*$, but $\mathrm{CT} \neq \mathrm{CT}^*$.

The difference between $\mathbf{Game}_0$ and $\mathbf{Game}_1$ is the replacement of a keyed permutation only. Thus, by the difference lemma [5], we obtain that the difference between adversarial advantages of those two games is bounded by the advantage of any PPT adversary who breaks strong pseudorandomness of the exploited keyed permutation F. Since we assume that F is a strong pseudorandom permutation, we have that this difference is negligible in the security parameter.

Lemma 1. *The difference between adversarial advantages in* $\mathbf{Game}_1$ *and* $\mathbf{Game}_2$ *is negligible in the security parameter if the exploited MAC scheme is existentially unforgeable under chosen message attack in the random oracle model.*

Proof of Lemma 1. We analyze the difference between adversarial advantages in $\mathbf{Game}_1$ and $\mathbf{Game}_2$. Let $\mathcal{D}$ be the event that the adversary asks a decryption oracle query on a valid ciphertext $\text{CT} = (C_1, C_2, C_3)$ such that $C_1 = C_1^*$, but $\text{CT} \neq \text{CT}^*$. We construct a simulator $\mathcal{B}$ who breaks existential unforgeability of the exploited $\mathsf{MAC} = (\mathsf{G}, \mathsf{S}, \mathsf{V})$ using the adversary $\mathcal{A}$ who generates the event $\mathcal{D}$. Suppose that the signing oracle $\mathcal{O}^{\mathsf{S}(K,\cdot)}$, which takes a message m as an input and returns the outcome of $\mathsf{S}(K, m)$, and the verification oracle $\mathcal{O}^{\mathsf{V}(K,\cdot,\cdot)}$, which takes a pair of message m and tag T as an input and returns the outcome of $\mathsf{V}(K, m, T)$, are given to $\mathcal{B}$ for the fixed target secret key K. Then, $\mathcal{B}$ performs as follows.

1. $\mathcal{B}$ runs $\mathsf{Setup}(\lambda)$ to generate a system public parameter PP, replaces a MAC scheme in PP by the target MAC scheme MAC, and passes PP to $\mathcal{A}$.
2. $\mathcal{B}$ responses $\mathcal{A}$'s queries by the following manner:
   - $\mathcal{O}^{\mathsf{H}}$ query: Given $m \in \{0,1\}^t$ as an input, it first checks whether there exists the same input stored at the H-table, which was initiated as an empty set at the beginning of the game. If exists, it returns the stored value. Otherwise, it generates a random value $h \in \{0,1\}^n$, stores (m, h) at the H-table, and returns h.
   - $\mathcal{O}^{\mathsf{H}_1}$ query: On input $\mathsf{ID} \in \{0,1\}^*$, it first searches the H_1-table, which was initiated as an empty set at the beginning of the game. If the same input is stored, it returns the stored value. Otherwise, it chooses a random value $h_1 \in \mathbb{G}_1$, stores (ID, h_1) at the H_1-table, and returns h_1.
   - $\mathcal{O}^{\mathsf{H}_2}$ query: Given a pair $(T, C_2, E) \in \mathcal{T} \times \mathbb{G}_1 \times \mathbb{G}_2$ as an input, it first searches the H_2-table, which was initiated as an empty set at the beginning of the game. If the same input is stored at the table, it outputs the stored value. Otherwise, it selects a random value $h_2 \in \{0,1\}^{t+\ell}$, adds (T, C_2, E, h_2) to the H_2-table, and returns h_2.
   - $\mathcal{O}^{f}$ query: On input h, it first checks whether h is stored at the f-table, which was initiated as an empty set at the beginning of the game. If exists, it returns the stored value. Otherwise, it selects a random value R_f from $\{0,1\}^n \setminus S_{fout}$ where S_{fout} is the set of outputs stored at the f-table. Then, it stores (h, R_f) at the f-table and returns R_f.
   - $\mathcal{O}^{\mathsf{Extract}}$ query: On input an identity ID, $\mathcal{B}$ returns d_{ID} by computing $d_{\mathsf{ID}} = g_{\mathsf{ID}}^{\alpha}$ using the master secret key $\text{MSK} = \alpha$.
   - $\mathcal{O}^{\mathsf{Enc}}$ query: Given an identity ID and a message m, it first requests $\mathcal{O}^{\mathsf{H}}$ query on input m and receives h. Then, it requests $\mathcal{O}^{f}$ query on input h and receives R_f.

Thereafter, it requests $O^{\mathsf{S}(K,\cdot)}$ query on R_f and receives T. It sets a ciphertext $\text{CT} = (C_1, C_2, C_3)$ so that

$$C_1 = R_f, C_2 = g^r, \text{ and } C_3 = (m\|r) \oplus h_2$$

where r is a random element from $\mathbb{Z}_p^*$ and h_2 is the result of $\mathcal{O}^{\mathsf{H}_2}$ query on $(T, C_2, e(P_{pub}, g_{\mathsf{ID}})^r)$. Finally, it returns $\text{CT} = (C_1, C_2, C_3)$.

– $\mathcal{O}^{\mathsf{Dec}}$ query: Given a ciphertext $\text{CT} = (C_1, C_2, C_3)$, it first asks a signing oracle query $\mathcal{O}^{\mathsf{S}(K,\cdot)}$ on C_1 and receives T. Then, it computes $e(C_2, d_{\mathsf{ID}})$, requests $\mathcal{O}^{\mathsf{H}_2}$ query on $(T, C_2, e(C_2, d_{\mathsf{ID}}))$ and receives h_2. Using h_2 and C_3, it recovers m and r, and then checks whether $(\mathsf{H}(m), C_1)$ is stored at the f-table and $C_2 = g^r$. If both hold, it returns m. Otherwise, it returns $\perp$.

3. Once $\mathcal{A}$ submits the target identity ID^* and two messages m_0, m_1, $\mathcal{B}$ selects a random bit $b \in \{0,1\}$ and asks an f-query on m_b. Then, it sets a challenge ciphertext to

$$C_1^* = f(m_b),\ C_2^* = g^{r^*},\ \text{and } C_3^* = R_2^*$$

for randomly chosen r^* and R_2^* from $\mathbb{Z}_p^*$ and $\{0,1\}^{t+n}$, respectively, and sends $\text{CT}^* = (C_1^*, C_2^*, C_3^*)$ to $\mathcal{A}$.

4. $\mathcal{B}$ responses $\mathcal{A}$'s queries as almost the same as in Step 2, except the case that $(C_1^*, \cdot, \cdot)$ is queried to the decryption oracle. In this case, the decryption oracle returns $\perp$.

5. Once $\mathcal{A}$ outputs its guess, $\mathcal{B}$ selects a random element (T, C_2, E, h_2) from the H_2-table and outputs $(f(m_b), T)$.

We note that if the event $\mathcal{D}$ occurs, the above game is the case that $\mathcal{B}$ interacts with $\mathcal{A}$ in $\mathbf{Game}_1$. Otherwise, it is the case that $\mathcal{B}$ interacts with $\mathcal{A}$ in $\mathbf{Game}_2$. On the other hand, in order that the event $\mathcal{D}$ occurs, $\mathcal{A}$ should request an H_2-oracle query on (T, C_2, E) where T is the output of $\mathsf{S}(K, f(m_b))$. So, the H_2-table includes the value T such that $1 \leftarrow \mathsf{V}(K, f(m_b), T)$ and $f(m_b)$ is not queried to the $\mathcal{O}^{\mathsf{S}(K,\cdot)}$ oracle since m_0 and m_1 cannot be queried to the encryption oracle. Thus, $\mathcal{B}$ can break existential unforgeability of MAC with probability $1/q_{\mathsf{H}_2}$ if $\mathcal{D}$ occurs where q_{H_2} is the number of H_2 queries. Therefore, the difference between adversarial advantages of $\mathbf{Game}_1$ and $\mathbf{Game}_2$ is negligible if the exploited MAC is existentially unforgeable under chosen message attack. □

Lemma 2. *The advantage of the adversary in* $\mathbf{Game}_2$ *is negligible in the security parameter if the BDH assumption holds in the random oracle model.*

Proof of Lemma 2. We note that the proof of this lemma is very similar to that of Theorem 1 in [7]. We construct a simulator $\mathcal{B}$ who solves the BDH problem using the adversary $\mathcal{A}$ in $\mathbf{Game}_2$. Suppose that the BDH instance $(X = g^x, Y = g^y, Z = g^z)$ for randomly chosen $x, y, z \in \mathbb{Z}_p^*$ is given. Then, $\mathcal{B}$ performs as follows.

1. $\mathcal{B}$ executes $\mathsf{Setup}(\lambda)$ to obtain a system public parameter PP and replaces P_{pub} by $P_{pub} = X^s = (g^x)^s$ for randomly chosen $s \in \mathbb{Z}_p^*$. $\mathcal{B}$ sends PP to $\mathcal{A}$. We note that $\mathcal{B}$ knows MTK and s, but does not know x.

2. $\mathcal{B}$ responses $\mathcal{A}$'s queries by the following manner:
   – $\mathcal{O}^{\mathsf{H}}$, $\mathcal{O}^{\mathsf{H}_2}$, $\mathcal{O}^f$ queries: They perform as the same as in the proof of Lemma 1.

– $\mathcal{O}^{\mathsf{H}_1}$: On input $\mathsf{ID}_i \in \{0,1\}^*$, it first searches the H_1-table, which was initiated as an empty set at the beginning of the game. If the same input is stored, it returns the stored value. Otherwise, it tosses a coin with $\Pr[\mathsf{coin}_i = 0] = \delta$. If $\mathsf{coin}_i = 1$, it selects a random element a_i from $\mathbb{Z}_p^*$ and computes $g_{\mathsf{ID}_i} = g^{a_i}$. Otherwise, it computes $g_{\mathsf{ID}_i} = Y^{a_i} = (g^y)^{a_i}$. For each case, it stores $(\mathsf{ID}_i, g_{\mathsf{ID}_i}, a_i, \mathsf{coin}_i)$ at the H_1-table.
– $\mathcal{O}^{\mathsf{Extract}}$ query: Given an identity ID_i as an input, it first obtains $\mathsf{H}_1(\mathsf{ID}_i)$ by requesting the H_1 query. If $\mathsf{coin}_i = 0$, $\mathcal{B}$ aborts. Otherwise, it computes $d_{\mathsf{ID}_i} = P_{pub}^{a_i} = (X)^{s \cdot a_i}$, and returns d_{ID_i} to $\mathcal{A}$.
– $\mathcal{O}^{\mathsf{Enc}}$ query: Given an identity ID_i and a message m, it first asks the H_1 query to obtain $\mathsf{H}_1(\mathsf{ID}_i)$ and the f query on m. Then, $\mathcal{B}$ chooses a random element $r \in \mathbb{Z}_p^*$, and computes

$$C_1 = f(m),\ C_2 = g^r,\ C_3 = (m \| r) \oplus h_2$$

where h_2 is the response of the H_2 query on $(T, C_2, e(P_{pub}, g_{\mathsf{ID}_i})^r)$ and $T \leftarrow \mathsf{S}(K_2, C_1)$. It returns $\text{CT} = (C_1, C_2, C_3)$.
– $\mathcal{O}^{\mathsf{Dec}}$ query: Given a ciphertext $\text{CT} = (C_1, C_2, C_3)$ for identity ID_i, it first searches the H_1-table to obtain $(\mathsf{ID}_i, g_{\mathsf{ID}_i}, a_i, \mathsf{coin}_i)$. If $\mathsf{coin}_i = 0$, it aborts. Otherwise, it computes

$$e(X, C_2)^{sa_i} = e(X^s, g^r)^{a_i} = e(P_{pub}, g_{\mathsf{ID}_i})^r$$

where r is the randomness used for generating CT. Then, it runs $T \leftarrow \mathsf{S}(K_2, C_1)$ and obtains h_2 by requesting a H_2 query on $(T, C_2, e(X, C_2)^{sa_i})$. It recovers m and r from $m \| r = C_3 \oplus h_2$, checks whether $C_1 = f(m)$ and $C_2 = g^r$. If both hold, it returns m. Otherwise, it returns $\perp$.

3. Once $\mathcal{A}$ submits the target identity ID^* and two messages m_0, m_1, $\mathcal{B}$ first asks a H_1 query on ID^* to obtain $(\mathsf{ID}^*, g_{\mathsf{ID}^*}, a^*, \mathsf{coin}^*)$. If $\mathsf{coin}^* = 1$, it aborts. Otherwise, it randomly selects $b \in \{0,1\}$ and asks an f query on m_b. Then, it sets a challenge ciphertext to

$$C_1^* = f(m_b),\ C_2^* = Z, \text{ and } C_3^* = R_2^*$$

where R_2^* is randomly chosen from $\{0,1\}^{t+\ell}$. $\mathcal{B}$ sends $\text{CT}^* = (C_1^*, C_2^*, C_3^*)$ to $\mathcal{A}$.
4. $\mathcal{B}$ responses $\mathcal{A}$'s queries as the same as in Step 2.
5. Once $\mathcal{A}$ outputs its guess b', if $b \neq b'$, then $\mathcal{B}$ aborts. Otherwise, it randomly selects a tuple (T, C_2, E) from the H_2-table and returns $E^{(a^* s)^{-1}}$ as a solution to the BDH problem on instance (X, Y, Z) where E is expected to be $e(P_{pub}, g_{\mathsf{ID}^*})^z = e(X^s, Y^{a^*})^z = e(X, Y)^{sa^* z}$.

We note that the above simulator is actually the same as that in the proof of Theorem 1 in [7], except the adjustment by considering that the part for equality test and the first component of an input of H_2 function are modified. Thus, the analysis for the advantage of $\mathcal{B}$ is exactly the same as that in the proof of Theorem 1 in [7]. We omit the details and refer to [7].

From the difference lemma, Lemmas 1 and 2, we proved that our modification in Section 4.1 is wIND-ID-CCA2 secure if the exploited keyed permutation F is strong pseudorandom, the employed MAC scheme is existentially unforgeable under chosen message attack, and the BDH assumption holds. □