On the Security of Luo et al.’s Fully Secure Hierarchical Identity Based Encryption Scheme

Kwangsu Lee*  Jong Hwan Park†  Dong Hoon Lee‡

Abstract

Recently, Luo et al. proposed an efficient hierarchical identity based encryption (HIBE) scheme with constant size of ciphertexts, and proved its full security under standard assumptions. To construct the scheme, they used the dual system encryption technique of Waters, and devised a method that compresses the tag values of dual system encryption. In this paper, we show that the security proof of Luo et al. is wrong since there exists an algorithm that distinguishes whether it is a simulation or not.

Keywords: Hierarchical identity based encryption, Full security, Bilinear pairing.

1 Introduction

Hierarchical identity based encryption (HIBE) is an important extension of identity based encryption (IBE) [3] that uses a user’s identity as a public key. In HIBE, the identity of a user is represented as a hierarchical structure, and a high level user can delegate the capability of key generation to a low level user. The concept of HIBE was proposed by Horwitz and Lynn [10] to reduce the burden of the key generation center of IBE, and the first secure construction was presented by Gentry and Silverberg [9] in the random oracle model. The research of HIBE is important not only because it is the extension of IBE, but also because it has many applications like chosen ciphertext secure public key encryption, forward secure public key encryption, public key broadcast encryption, and searchable encryption with keywords [4–7].

An important problem of HIBE is to construct an efficient HIBE scheme that is fully secure under standard assumptions without random oracles. Many HIBE schemes of previous research were proven to be secure in the selective security model where the capability of an adversary is restricted [1,2,4]. Though there were some HIBE schemes that were proven to be secure in the full security model, they were inefficient or secure under complex assumptions [8,14]. Recently, Waters introduced the dual system encryption method and proposed an efficient HIBE scheme with linear size of ciphertexts that is fully secure under standard assumptions [15]. After that, Lewko and Waters proposed an efficient HIBE scheme with constant size of ciphertexts that is fully secure under static assumptions using the dual system encryption method [11]. However, it is still an open problem to construct an efficient HIBE scheme with constant size of ciphertexts that is fully secure under standard assumptions.

To solve this open problem, Luo et al. proposed an efficient and fully secure HIBE scheme with constant size of ciphertexts under standard assumptions [12]. In this paper, we show that the security proof of Luo

*Korea University, Korea and Columbia University, USA. Email: guspin@korea.ac.kr.
†Korea University, Korea. Email: decartian@korea.ac.kr.
‡Korea University, Korea. Email: donghlee@korea.ac.kr.
et al. is wrong. The HIBE scheme of Luo et al. is constructed by combining the scheme of Waters that uses dual system encryption and the scheme of Boneh et al. that has constant size of ciphertexts \[2,15\]. To combine these two schemes, Luo et al. devised a method that compresses tag values of Waters’ HIBE scheme. We show that there exists an algorithm that decides whether the tag value of a challenge ciphertext in the security proof is random or not, hence the tag compression technique of Luo et al. is flawed.

## 2 Review of Luo-Chen-Hu-Chen HIBE Scheme

We briefly review the bilinear groups and the HIBE scheme of Luo et al. [12].

### 2.1 Bilinear Groups

Let \( G \) and \( G_T \) be multiplicative cyclic groups of prime \( p \) order. Let \( g \) be a generator of \( G \). The bilinear map \( e : G \times G \to G_T \) has the properties:

1. Bilinearity: \( \forall u, v \in G \) and \( \forall a, b \in \mathbb{Z}_p \), \( e(u^a, v^b) = e(u,v)^{ab} \).
2. Non-degeneracy: \( \exists g \) such that \( e(g, g) \neq 1 \), that is, \( e(g, g) \) is a generator of \( G_T \).

We say that \( G, G_T \) are bilinear groups if the group operations in \( G \) and \( G_T \) as well as the bilinear map \( e \) are all efficiently computable.

### 2.2 Luo-Chen-Hu-Chen HIBE Scheme

**Setup\((1^\lambda, I)\):** The setup algorithm generates the bilinear groups \( G, G_T \) of prime order \( p \) of bit size \( \Theta(\lambda) \). Next, it chooses random elements \( g, v, v_1, v_2, w, u_1, u_2, h_1, \ldots, h_l \in G \) and exponents \( a_1, a_2, b, \alpha \in \mathbb{Z}_p \). Then it sets \( \tau_1 = v_1^{a_1}, \tau_2 = v_2^{a_2} \). It outputs the master key \( MK = (g^\alpha, g^{a_1}, g^{a_2}) \) and the public parameters \( PK \) (with the description of \( (p, G, G_T, e) \)) as

\[
PK = (g, g^b, g^a, g^{a_1}, g^{a_2}, G, \tau_1, \tau_2, \tau_1^b \tau_2, v, v_1, v_2, w, u_1, \ldots, u_l, h_1, \ldots, h_l, \Omega = e(g, g)^{\alpha a_1 b}).
\]

**KeyGen\((ID, MK, PK)\):** The key generation algorithm takes as input an identity vector \( ID = (I_1, \ldots, I_n) \in \mathbb{Z}_p^n \) and the master key \( MK \). It selects random exponents \( r_1, r_2, z_1, z_2, \tau_1, \tau_2, \ldots, \tau_n, \tag{k}, \tag{n+1}, \ldots, \tag{n} \in \mathbb{Z}_p \) and computes

\[
D_1 = g^{\alpha a_1 r_1}, D_2 = g^{-a_1 r_1 + r_2} g^{z_1}, D_3 = (g^b)^{-z_2}, D_4 = v_1^{r_1 + r_2} g^{-z_1}, D_5 = (g^b)^{-z_2}, D_6 = (g^b)^{r_2}, D_7 = g^{z_1},
\]

\[
K = (u_1^{l_1} u_2^{l_2} \cdots u_n^{l_n} w^{\tag{n+1} h_1 h_2 \cdots h_n} r_1),
\]

\[
K_{n+1} = (u_1^{l_1} w^{r_1 \tag{n+1}}), \ldots, K_l = (u_1^{l_1} w^{\tag{n} h_1} r_1).
\]

It outputs the private key as

\[
SK_ID = (D_1, \ldots, D_7, K, K_{n+1}, \ldots, K_l, \tag{k}, \tag{n+1}, \ldots, \tag{n}).
\]

**Delegate\((ID', SK_ID, PK)\):** The delegation algorithm takes as input an identity vector \( ID' = (I_1, \ldots, I_{n+1}) \in \mathbb{Z}_p^{n+1} \) and a private key \( SK_ID = (D_1', \ldots, D_l', K_{n+1}', \ldots, K_l', \tag{k}', \tag{n+1}', \ldots, \tag{n}) \) for an identity vector \( ID = (I_1, \ldots, I_n) \in \mathbb{Z}_p^n \) where \( ID \) is a prefix of \( ID' \). It first sets \( \tag{k} = \tag{k}' + I_{n+1} \tag{n+1} \). Next, it
Encrypt follows:

Phase 1: A to PK

The HIBE scheme of Luo et al. was proved in the complete security model that traces paths of delegation

Decrypt chooses random exponents \( r_1, r_2, z_1, z_2 \in \mathbb{Z}_p \) and computes

\[
D_1 = D'_1 v^{r_1 + z_1}, \quad D_2 = D'_2 v^{r_2 + z_1}, \quad D_3 = D'_3 (g^b)^{z_1}, \\
D_4 = D'_4 v^{r_1 + r_2} g^{z_2}, \quad D_5 = D'_5 (g^b)^{-z_2}, \\
D_6 = D'_6 (g^b)^{z_2}, \quad D_7 = D'_7 g^{r_1}, \\
K = K'_i (K'_{n+1})^{l_{n+1}} (u_1 h_1^{l_i} \cdots u_{n+1} h_1^{l_i} \cdots h_{n+1}^{l_i})^{r_1}, \\
K_{n+2} = K'_{n+2} (u_{n+2} h_{n+2}^{l_i})^{r_1}, \ldots, \\
K_i = K'_i (u_i h_1^{l_i} \cdots h_{n}^{l_i})^{r_1}.
\]

Finally, it outputs the delegated private key as

\[
SK_{ID'} = (D_1, \ldots, D_7, K, K_{n+2}, \ldots, K_i, \text{tag}_k, \text{tag}_{n+2}, \ldots, \text{tag}_t).
\]

Encrypt(\(ID, M, PK\)): The encryption algorithm takes as input an identity vector \(ID = (I_1, \ldots, I_n) \in \mathbb{Z}_p^n\), a message \(M \in \mathbb{G}_T\), and the public parameters \(PK\). It selects random exponents \(s_1, s_2, t, \text{tag}_c \in \mathbb{Z}_p\) and computes

\[
C_0 = \Omega^{s_2} M, \quad C_1 = (g^b)^{s_1 + s_2}, \quad C_2 = (g^{b a_1})^{s_1}, \\
C_3 = (g^{a_1})^{s_1}, \quad C_4 = (g^{b a_2})^{s_1}, \quad C_5 = (a_2)^{s_2}, \\
C_6 = \tau_1^{s_1} \tau_2^{s_2}, \quad C_7 = (\tau_1^{s_1} (\tau_2^{s_2})^{-1} w^{-t}, \\
E_1 = (u_1 h_1^{l_i} \cdots u_{n} h_1^{l_i} \cdots h_{n}^{l_i})^{t}, \quad E_2 = g^t.
\]

It outputs the ciphertext as

\[
CT = (C_0, C_1, \ldots, C_7, E_1, E_2, \text{tag}_c).
\]

Decrypt(\(CT, SK_{ID}, PK\)): The decryption algorithm takes as input a ciphertext \(CT\) and a private key \(SK_{ID}\) for an identity vector \(ID = (I_1, \ldots, I_n)\). If \(\text{tag}_c \neq \text{tag}_k\), then it computes

\[
A_1 = e(C_1, D_1) \cdot e(C_2, D_2) \cdot e(C_3, D_3) \cdot e(C_4, D_4), \\
A_2 = e(C_5, D_6) \cdot e(C_7, D_7), \quad A_3 = A_1 / A_2, \\
A_4 = (e(E_1, D_7) / e(E_2, K))^{1/(\text{tag}_c - \text{tag}_k)}.
\]

Finally, it outputs the encrypted message by computing \(C_0 / (A_3 / A_4) = M\).

### 2.3 Security Model

The HIBE scheme of Luo et al. was proved in the complete security model that traces paths of delegation \([13][15]\). It is described as the following game between a challenger \(C\) and an adversary \(A\):

**Setup**: \(C\) first runs \(\text{Setup}(1^k, l)\) and keeps the master key \(MK\) to itself, then it gives the public parameters \(PK\) to \(A\). It also initializes a set \(S\).

**Phase 1**: \(A\) adaptively requests create, delegate, and reveal queries. For each query, \(C\) handles this as follows:
• **Create**(*ID*): It creates a private key for *ID* by running **KeyGen**(ID, MK, PK) and adds this to the set $S$. It gives a reference to $A$.

• **Delegate**(*ID*, *ID’*): It obtains a delegated private key for *ID’* from *ID* by running **Delegate**(ID’, SK$_{ID}$, PK) and adds this to the set $S$. It gives a reference to $A$.

• **Reveal**(*ID*): It removes the private key for *ID* from the set $S$ and gives $A$ the private key.

**Challenge:** $A$ submits a challenge identity vector $ID^*$ and two messages $M^*_0, M^*_1$ subject to the restriction that $ID$ of reveal queries in the phase 1 is not a prefix of $ID^*$. $C$ flips a random coin $\gamma \in \{0, 1\}$ and obtains a ciphertext by running **Encrypt**(ID*, $M^*_\gamma$, PK). It gives the ciphertext to $A$.

**Phase 2:** $A$ continues to request create, delegate, and reveal queries subject to the above restriction.

**Guess:** $A$ outputs a guess $\gamma' \in \{0, 1\}$ of $\gamma$, and wins the game if $\gamma' = \gamma$.

There is a previous security model of HIBE that does not trace paths of delegation [1, 2]. This previous security model of HIBE is only valid if private keys from the delegation algorithm are identically distributed to private keys from the key generation algorithm. The following claim shows that Luo et al.’s HIBE scheme cannot be proven in the previous security model of HIBE.

**Claim 2.1.** In Luo et al.’s HIBE scheme, private keys from the delegation algorithm are not identically distributed to private keys from the key generation algorithm.

**Proof.** The idea of this proof is that $tag_k = tag_k' + I_{n+1} tag_{n+1}$ obtained from the delegation algorithm is not a random value though $tag_k'$ and $tag_{n+1}$ are random values. Suppose that SK$_{ID}$ is a private key for *ID* generated from the key generation algorithm, then it contains random tag values $tag_k', tag_{n+1}, \ldots, tag_l$. For each identity $ID' = ID || I_{n+1,i}$, we obtain a delegated private key SK$_{ID'}$ from SK$_{ID}$ by running the delegation algorithm. Then tag values of delegated private keys are represented as a matrix equation

\[
\begin{pmatrix}
1 \\
I_{n+1,1} \\
I_{n+1,2} \\
\vdots \\
I_{n+1,m}
\end{pmatrix}
\begin{pmatrix}
tag_k' \\
tag_k \\
\vdots \\
tag_k_m
\end{pmatrix}
= \begin{pmatrix}
tag_{k,1} \\
tag_{k,2} \\
\vdots \\
tag_{k,m}
\end{pmatrix}.
\]

It is obvious that the row rank of the left-side $m \times 2$ matrix is two. Therefore, $tag_{k,1}, \ldots, tag_{k,m}$ that are derived from the delegation algorithm are not random values since the $m \times 2$ matrix is singular if $m > 2$.

The authors of Luo et al.’s paper erroneously stated that $tag_k = tag_k' + I_{n+1} tag_{n+1}$ derived from the delegation algorithm is still distributed uniformly in $\mathbb{Z}_p$.

### 3 Review of Waters’ Dual System Encryption

Before we analyze Luo et al.’s HIBE scheme, we first review the dual system encryption method of Waters [15]. To grasp the proof of Luo et al.’s HIBE scheme, we need to understand the proof methodology of dual system encryption since Luo et al.’s HIBE scheme is a variation of the HIBE scheme of Waters.

In dual system encryption, a ciphertext and a private key can be a normal type or a semi-functional type, and it should be hard to distinguish a normal type from a semi-functional type. A normal ciphertext and a normal private key are generated by the encryption algorithm and the key generation algorithm respectively. A semi-functional ciphertext and a semi-functional private key are only used in the security proof. The
normal private key can decrypt the normal ciphertext and the semi-functional ciphertext, whereas the semi-functional private key can only decrypt the normal ciphertext.

The proof of dual system encryption consists of hybrid games. The first game $Game_{Real}$ is the original security game where the challenge ciphertext and private keys are normal. In the next games, the challenge ciphertext is changed from normal to semi-functional, and the private keys are changed from normal to semi-functional one by one. That is, the game $Game_k$ is defined as a game where the challenge ciphertext is semi-functional, a private key with an index less than or equal $k$ is semi-functional, and a private key with an index greater than $k$ is normal. In the final game $Game_{Final}$, the challenge ciphertext and private keys are semi-functional, and the session key is a random value. Thus the proof is completed if no adversary can distinguish these games since the adversary’s advantage of the final game is zero.

In dual system encryption, a paradox of security proof should be solved. The paradox is described as follows. In the proof that distinguishes $Game_{k-1}$ from $Game_k$, a simulator can create a semi-functional ciphertext for any identity and it also can create a private key for any identity. Therefore the simulator can decrypt this semi-functional ciphertext using the $k$-th private key to determine whether it is normal or semi-functional. The $k$-th private key is normal if the decryption is successful, otherwise it is semi-functional. This situation is a contradiction since the simulator can distinguish the type of $k$-th private key without the help of an adversary.

To solve the paradox of dual system encryption, Waters devised a method that attaches random tags in a ciphertext and a private key. In the IBE scheme of Waters, the ciphertext and the private key contain a random value tag$_c$ and tag$_k$ respectively, and the decryption logic of IBE is changed from $ID_c = ID_k$ to $(ID_c = ID_k) \land (\text{tag}_c \neq \text{tag}_k)$ where $ID_c$ and $ID_k$ are identity information in the ciphertext and the private key respectively. The main idea to solve the paradox is that the simulator can only create a semi-functional ciphertext with tag$_c = F(ID_c)$ and the $k$-th private key is created with tag$_k = F(ID_k)$ where $F(x) = Ax + B$ with hidden values $A$ and $B$. If the simulator creates a semi-functional ciphertext for the identity $ID_k$ of $k$-th private key and decrypt this ciphertext using the $k$-th private key, then the decryption would always fail since tag$_c = \text{tag}_k = F(ID_k)$. Therefore, the paradox is solved since the simulator cannot distinguish the type of $k$-th private key without the help of an adversary.

To finish the proof that distinguishes $Game_{k-1}$ from $Game_k$, it should be guaranteed that the tag values tag$_k$, tag$_c$ of the $k$-th private key and the challenge semi-functional ciphertext look like random values to the adversary’s point of view even if these tag values are set as $F(x) = Ax + B$. The idea of achieving the tag randomness is to use the fact that the adversary of the security model cannot request a private key for the identity $ID^t$ of the challenge ciphertext. Thus the adversary can only obtain two different tag values tag$_k = F(ID_k)$ and tag$_c = F(ID^t)$ that are related with $F(x)$. These two tag values tag$_k$ and tag$_c$ are randomly distributed to the adversary’s point of view since $F(x) = Ax + B$ is a pairwise independent function that guarantees $F(ID)$ and $F(ID^t)$ are independent random values if $ID \neq ID^t$.

4 Analysis of Luo et al.'s HIBE Scheme

We first review the original security proof of Luo et al.’s HIBE scheme [12] and analyze their security proof.

4.1 Luo et al.’s Security Proof

We briefly describe Luo et al.’s security proof that distinguishes $Game_{k-1}$ from $Game_k$ focusing on the way to create tag values since analyzing the tag values of private keys and ciphertexts is enough for our analysis.
In the security proof that distinguishes $Game_{k-1}$ from $Game_k$, a simulator $B$ that interacts with an adversary $A$ is described as follows. $B$ first creates public parameters using the elements of an assumption, and it embeds $F_j(x) = A_jx + B_j$ in public parameters for each depth of HIBE to solve the paradox. In the phase 1 and phase 2 steps, $A$ can adaptively request private key queries. If this is the $i$-th query for $ID = (I_1, \ldots, I_n)$, then $B$ handles the query as follows:

- Case $i < k$: $B$ first creates a normal private key with random tag values by running the key generation algorithm using the master key and modifies it to a semi-functional private key. Then it gives the semi-functional private key to $A$.

- Case $i = k$: $B$ creates a private key with tag $w_k = F_1(I_1) + I_2F_2(I_1) + \cdots + I_nF_n(I_1), \tag{1}$ to solve the paradox using the target value of the assumption, and it gives the private key to $A$.

The private key can be normal or semi-functional depending on the target value of the assumption.

- Case $i > k$: $B$ creates a normal private key with random tag values by running the key generation algorithm using the master key and gives the private key to $A$.

In the challenge step, $A$ gives $ID^* = (I_1^*, \ldots, I_n^*)$ and $M_0^*, M_1^*$ to $B$. $B$ creates a semi-functional ciphertext for $ID^*$ and $M_1^*$ with tag $\gamma = F_1(I_1^*) + I_2F_2(I_1^*) + \cdots + I_nF_n(I_n^*)$ to solve the paradox, and gives the ciphertext to $A$. Finally, $A$ outputs a guess $\gamma'$, and $B$ solves the assumption using the guess of $A$.

### 4.2 Analysis of Luo et al.’s Security Proof

The security proof of Luo et al. is not complete since it didn’t describe the create, delegate, and reveal queries explicitly. There are two possible methods of handling the private key queries in Luo et al.’s security proof if the create, delegate, and reveal queries are considered.

The first method is that the private key queries in Luo et al.’s security proof are handled as the reveal queries. The simulator handles the private key queries as follows. If this is a create query or a delegate query, then the simulator just gives a reference to the adversary. If this is a reveal query, then the simulator creates a private key as the same way regardless of whether it is a create query or a delegate query. This method is only valid if the distribution of the key generation algorithm and the distribution of the delegation algorithm are the same. However, the distributions of two algorithms are different by the Claim 1.

The second method is that the private key queries in Luo et al.’s security proof are handled as the create queries. The simulator handles the private key queries as follows. If this is a create query, then the simulator creates a private key as the same as in Luo et al.’s security proof, stores it in a set, and gives a reference to the adversary. If this is a delegate query, then the simulator runs the delegation algorithm to the private key in the set and gives a reference to the adversary. If this is a reveal query, then the simulator retrieves a private key from the set, gives it to the adversary, and deletes it from the set. In the hybrid games of dual system encryption, the private keys that are revealed to the adversary are changed from normal to semi-functional. Suppose that the adversary makes $q_C$ create queries, $q_D$ delegate queries, and $q_R$ reveal queries. Then the sequence of hybrid games is defined as $Game_{Real}, Game_0, \ldots, Game_{q_R}, Game_{Final}$ since the number of hybrid games depends on the number of reveal queries $q_R$. Thus the index $k$ in the security proof that distinguishes $Game_{k-1}$ and $Game_k$ is related to the $k$-th reveal query, not the $k$-th create query. To solve the paradox of dual system encryption, the private key from the $k$-th create query in Luo et al.’s security proof should be revealed in the $k$-th reveal query. However, the simulator cannot guarantee that the private key from the $k$-th create key is revealed in the $k$-th reveal query since an adversary can request the $k$-th reveal query for an
identity that is not related to the identity of the \(k\)-th create query. In this case, the security proof fails since the paradox is not solved.

The security proof of Waters’ HIBE scheme solved this problem by selecting a random index \(\kappa\) for the create queries and aborting the simulation if the private key of the \(k\)-th reveal query was not created in the \(\kappa\)-th create query \([15]\). The success probability of this simulator is at least \(1/q_C\) since \(\kappa\) is a random value between 1 and \(q_C\). Therefore the security proof of Luo et al. should be modified as the same as Waters’ security proof by correctly handling the create, delegate, and reveal queries.

### 4.3 Modified Security Proof

The modified security proof of Luo et al.’s HIBE scheme that distinguishes \(\text{Game}_{k-1}\) from \(\text{Game}_k\) is described as follows.

Suppose that an adversary \(\mathcal{A}\) that distinguishes \(\text{Game}_{k-1}\) from \(\text{Game}_k\) requests \(q_C\) create queries, \(q_D\) delegate queries, and \(q_R\) reveal queries. A simulator \(\mathcal{B}\) that solves an assumption using the adversary \(\mathcal{A}\) works as follows. \(\mathcal{B}\) first creates public parameters using the elements of an assumption, and it embeds \(F_j(x) = A_jx + B_j\) in the public parameters for each depth of HIBE to solve the paradox. In the phase 1 and 2 steps, \(\mathcal{A}\) can request create, delegate, and reveal queries for private keys. \(\mathcal{B}\) first selects a random \(\kappa\) such that \(1 \leq \kappa \leq q_C\) to guess that the \(k\)-th revealed private key will be the \(\kappa\)-th created one or a delegated one from the \(\kappa\)-th created one. If this is a create query for \(ID = (I_1, \ldots, I_{\gamma})\) with an index \(j\), then \(\mathcal{B}\) handles this as follows:

- **Case** \(j \neq \kappa\): \(\mathcal{B}\) selects random tag values and stores them. Next it gives a reference to \(\mathcal{A}\).
- **Case** \(j = \kappa\): \(\mathcal{B}\) sets \(\text{tag}_k = F_1(I_1) + I_2F_2(I_1) + \cdots + I_nF_n(I_1), \text{tag}_{n+1} = F_{n+1}(I_1), \ldots, \text{tag}_l = F_l(I_1)\) to solve the paradox and stores them. Next it gives a reference to \(\mathcal{A}\).

If this is a delegate query, then \(\mathcal{B}\) changes the tag values the same as the delegation algorithm and stores them. Next it gives a reference to \(\mathcal{A}\). If this is a reveal query for \(ID = (I_1, \ldots, I_{\gamma})\) with an index \(i\), then \(\mathcal{B}\) handles the query as follows:

- **Case** \(i < k\): \(\mathcal{B}\) creates a semi-functional private key with the stored tag values and gives the private key to \(\mathcal{A}\).
- **Case** \(i = k\): Suppose that \(\mathcal{A}\) requested the reveal query for the \(j\)-th created private key or a delegated one from the \(j\)-th created one. If \(j \neq \kappa\), then \(\mathcal{B}\) aborts the simulation since it fails to guess. Otherwise, \(\mathcal{B}\) creates a private key with the stored tag values that are fixed using \(F_j(x)\) and gives the private key to \(\mathcal{A}\).
- **Case** \(i > k\): \(\mathcal{B}\) creates a normal private key with the stored tag values and gives the private key to \(\mathcal{A}\).

In the challenge step, \(\mathcal{B}\) creates a semi-functional ciphertext for \(ID^* = (I_1^*, \ldots, I_{\gamma}^*)\) and \(M_\gamma^*\) with \(\text{tag}_r = F_1(I_1^*) + I_2^*F_2(I_1^*) + \cdots + I_{\gamma}^*F_{\gamma}(I_1^*)\) to solve the paradox and gives the ciphertext to \(\mathcal{A}\). Finally, \(\mathcal{A}\) outputs a guess \(\gamma'\) and \(\mathcal{B}\) solves the assumption using the guess of \(\mathcal{A}\). The probability that \(\mathcal{B}\) succeeds the simulation is \(1/q_C\) since \(\mathcal{B}\) does not abort if it correctly guesses \(\kappa\) such that \(1 \leq \kappa \leq q_C\).

### 4.4 Analysis of the Modified Security Proof

In this section, we show that there is an algorithm that distinguishes two security games by checking the distribution of the tag value in the challenge ciphertext (not in private keys). Note that the encryption
algorithm selects tag, as a random value in \( \mathbb{Z}_p \), whereas the modified simulator (and the original simulator) should set tag, = \( F_1(I'_1) + \cdots + I'_nF_n(I'_1) \) in the challenge ciphertext to solve the paradox.

**Claim 4.1.** There exists an algorithm that can distinguish \( Game_{k-1} \) from \( Game_k \) with non-negligible advantage in the modified security proof.

**Proof.** Before proving this claim, we briefly review the organization of a correct security proof. Suppose that a decisional assumption for the security proof is that no probabilistic polynomial-time algorithm can distinguish the distribution \( D_0 \) from the distribution \( D_1 \) with non-negligible advantage. For the correct security proof, we can define an additional sequence of hybrid games as follows:

- **Game\(_{k-1,0}\):** This game is equal to \( Game_{k-1} \). That is, the challenger ciphertext is semi-functional, a private key with an index less than or equal to \( k-1 \) is semi-functional, and a private key with an index greater than \( k-1 \) is normal. Note that the challenger selects the master key by itself.
- **Game\(_{k-1,1}\):** This game is similar to the \( Game_{k-1,0} \) except that a simulator simulates \( Game_{k-1,0} \) by using \( D_0 \) of the given assumption, instead of selecting the master key.
- **Game\(_{k-1,2}\):** This game is similar to the \( Game_{k,0} \) except that a simulator simulates \( Game_{k,0} \) by using \( D_1 \) of the given assumption, instead of selecting the master key.
- **Game\(_{k,0}\):** This game is equal to \( Game_k \). That is, the challenger ciphertext is semi-functional, a private key with an index less than or equal to \( k \) is semi-functional, and a private key with an index greater than \( k \) is normal. Note that the challenger selects the master key by itself.

In the correct security proof, two games \( Game_{k-1,0} \) and \( Game_{k-1,1} \) should be statistically indistinguishable, and two games \( Game_{k-1,2} \) and \( Game_{k,0} \) should be statistically indistinguishable. In this case, we can informally say that the indistinguishability of two games \( Game_{k-1} \) and \( Game_k \) is related to the security of the decisional assumption.

To prove this claim, we show that there is an algorithm that can distinguish \( Game_{k-1,0} \) from \( Game_{k-1,1} \) with a non-negligible probability. The idea is that it is easy for an algorithm to decide whether tag, in the challenge ciphertext is random or not if three tag values of a pairwise independent function \( G(x) = F_1 + xF_2 \) are given. An algorithm \( A \) that can distinguish \( Game_{k-1,0} \) from \( Game_{k-1,1} \) is described as follows:

1. \( A \) first requests a create query for an identity \( ID_1 = (I_1) \) and receives a reference \( h_1 \).
2. \( A \) requests a delegate query for an identity \( ID_2 = (I_1, I_2) \) from \( h_1 \) and receives a reference \( h_2 \). Next, it requests the \( k \)-th reveal query for \( h_2 \) and receives a private key with tag\(_{k,1}, \tag_{3}, \ldots, \tag_{l} \).
3. In the challenge step, \( A \) gives an identity \( ID_3 = (I_1, I_3) \) and receives a semi-functional ciphertext with tag\(_{c} \).
4. \( A \) requests a delegate query for an identity \( ID_4 = (I_1, I_4) \) from \( h_1 \) and receives a reference \( h_4 \). Next, it requests a reveal query for \( h_4 \) and receives a normal private key with tag\(_{k,1}, \tag_{3}, \ldots, \tag_{l} \).
5. Finally, \( A \) outputs 1 if \( \text{tag,} = \text{tag}_{k} + (I_3 - I_2) \cdot (\text{tag}_{k} - \text{tag}_{l}^{'})/(I_2 - I_4) \). Otherwise, it outputs 0.

The reveal queries of \( A \) are allowed in the complete security model since each identity of the reveal queries is not a prefix of the challenge identity. In \( Game_{k-1,0} \), \( A \) outputs 0 with \( 1 - 1/p \) probability since the above equation does not hold except \( 1/p \) probability if \( \text{tag,} \) is randomly chosen. In \( Game_{k-1,1} \), \( A \) outputs 1 with 1
probability since the above equation always holds if the tag values are set as $\tag_k = F_1(I_1) + I_2 F_2(I_1)$, $\tag_c = F_1(I_1) + I_3 F_2(I_1)$, $\tag'_k = F_1(I_1) + I_4 F_2(I_1)$ to solve the paradox. We can also show that there is an algorithm that can distinguish $\text{Game}_{k-1,2}$ from $\text{Game}_{k,0}$ using the similar argument.

The above algorithm only uses two facts such that 1) the simulator of the security proof sets $\tag_c = F_1(I_1) + I_3 F_2(I_1)$ to solve the paradox, and 2) it is possible to decompose two tag values $\tag'_k$ and $\tag_{n+1}$ from the delegated tag value $\tag_k = \tag'_k + I_{n+1} \tag_{n+1}$ using the delegate query since $\tag'_k$ and $\tag_{n+1}$ are fixed values. As we know, there is no method to solve the paradox of dual system encryption using tags without setting $\tag_c = F_1(I_1) + I_3 F_2(I_1)$ in Luo et al.’s HIBE scheme. Additionally, the decomposability of the delegate tag value is inherent in Luo et al.’s HIBE scheme.

5 Conclusion

In this paper, we first showed that the original security proof of Luo et al.’s HIBE scheme is incomplete since the authors didn’t properly handle the create, delegate, and reveal queries in their security proofs. Next we showed that the modified security proof that correctly handles the create, delegate, and reveal queries is still flawed since the tag compression technique of Luo et al. does not guarantee the tag randomness in the security proof.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 2010-0029121).

References


