Generic Transforms to Acquire CCA-Security for Identity Based Encryption: the Cases of FOPKC and REACT

Takashi Kitagawa* Peng Yang† Goichiro Hanaoka* Rui Zhang*
Kanta Matsuura† Hideki Imai*

Abstract

Fujisaki-Okamoto (FOPKC) conversion [14] and REACT conversion [18] are widely known to be able to generically convert a weak public key encryption scheme to a strong encryption scheme, i.e., indistinguishable against adaptive chosen ciphertext attacks (IND-CCA). In this paper, we discuss applications of Fujisaki-Okamoto (FOPKC) conversion and REACT conversion to Identity Based Encryptions (IBE). It has not been formally verified yet that whether these conversions are generic in the IBE setting, in other words, capable to upgrade the security of underlying IBE schemes, although many identity based encryption schemes are in fact built on these conversions. Our results show that both conversions are effective in the IBE case: plain REACT already achieves a good security reduction while the plain FOPKC conversion results in bad running time of the simulator. We further propose a simple modification to the plain FOPKC that solves this problem. Interestingly, our results may also show a separation between these two different attack models. Finally, we choose some concrete parameters to explain (visually) the effect of how the modified FOPKC substantially improves reduction cost regarding the plain conversion.

*Research Center for Information Security (RCIS), National Institute of Advanced Industrial Science and Technology (AIST). {t-kitagawa,hanaoka-goichiro,r-zhang,h-imai}@aist.go.jp

†Institute of Industrial Science, The University of Tokyo. {pengyang,kanta}@iis.u-tokyo.ac.jp

Contents

1 Introduction

Identity based encryption (IBE) [19] is a public key encryption scheme where the encryption public key can be an arbitrary string, such as the recipient's identity, thus the distribution of public key certificates can be avoided for an IBE scheme. However, only recently did the first practical full-functional IBE schemes [8, 10] came into birth, which are proven secure in the random oracle model [11, 4]. Subsequent research further extended to the standard model [9, 6, 5, 20].

It has been shown [2] that the strongest security notion for IBE is *indistinguishability against adaptive chosen* ID *and adaptive chosen ciphertext attacks* (IND-ID-CCA). Nevertheless, in the random oracle model, many IND-ID-CCA IBE schemes (like [8, 15]) are often built with the following strategy:

$$\text{Full Scheme} = \text{Basic Scheme} + \text{Transform}$$

where the basic scheme is the one appeared in [7, 8] and the transform may be those proposed previously for public key encryptions, like FOcrypto [13], FOpkc [14] and REACT [18]. We have confirmed that the FOcrypto conversion is generic for any CPA-secure IBE [21]. Here, we would like to consider the FOpkc and REACT conversions.

More exactly, we not only try to confirm the feasibility of such transforms in the IBE setting, but also focus on gaining *tight* security reduction of such a proof, which is never a trivial job. It is worth reminding that a loose security reduction usually means lower security level with the same key size, and one has to adopt longer keys to compensate this security loss.

1.1 Related Work

As mentioned already, Boneh and Franklin proposed the first practical IBE scheme which became the base for many related researches. The proof given by Boneh and Franklin was quite loose, however, nobody had considered how to improve the security reduction cost of their proof, even nobody had challenged the correctness of their proof until recently, Galindo [15] noticed a small flawed step in the proof of Boneh-Franklin's paper, however, the reduction in Galindo's corrected proof was even looser. We also note that in fact, the proof given in [8, 15] did not take account of applying generic FO transforms but has mainly considered how to reduce the security of the "full" scheme to that of an IND-CCA public key encryption scheme.

Another variant of Boneh-Franklin IBE scheme with tighter security reduction was given by Libert and Quisquater [17], with a REACT-like appearance.Again, there is no clear discussion on generic transforms for IBE in their paper, since this is not the theme of their work.

1.2 Our Results

We prove that these conversions (FOpkc and REACT) can be applied to IBE generically with good reduction costs. More precisely, applying the plain FOpkc to Boneh-Franklin's basic scheme, one immediately acquires the scheme proposed recently by Galindo [15], yet with a very loose security reduction. Recall that in the public key setting, the FOpkc conversion can be proven with a "tight" security reduction to its underlying primitives.

However, we can partially overcome this problem with a tiny modification to the plain FOpkc. The modification is itself very simple and computationally efficient: just hash the ID with other inputs to the random oracle. However, this simple idea actually works! The modified FOpkc conversion admits exactly tight reduction reduction as its public key counterpart.

On the other hand, the plain REACT already gives a good reduction cost, without any modification. Interestingly, these results may indicate a separation between the chosen plaintext attack (CPA) and plaintext checking attack (PCA) in the IBE setting.

We further choose some concrete parameters to explain how the modified FOPKC improves the reduction cost, by estimating the average running time of the simulator. For chosen parameters, using a single PC (or a single dedicated hardware), an IND-ID-CCA adversary breaks the IND-ID-CCA security of "the basic Boneh-Franklin scheme + the plain FOPKC conversion" with about 10^{24} years in addition to break the IND-ID-CPA security of the basic Boneh-Franklin scheme. However, it needs only additional 10^8 or 10^9 years in the case of the modified FOPKC conversion. Consider possible paralleled computing, say 1 million PCs, this becomes $10^2 \sim 10^3$ years. Furthermore, applying Moore's law, in 15 years, this will become only 1.30 years!

Remark 1. *Besides the* IND-ID-CCA *security, there is an another weaker security notion for IBE, called security against* selective ID *(*sID*) attack [9]. This security notion is also very useful and has applications in constructing CCA-secure public key encryptions. However, with similar discussions of this paper, one can reach similar results for* sID *secure IBE: the FO*PKC *(REACT) are also feasible to upgrade weak* sID *secure IBE to acquire* CCA *security, and the reduction costs follow naturally the discussions above.*

2 Preliminary

In this section we review the definition of IBE, several conventions and security notions.

2.1 Identity Based Encryption

Formally, an IBE scheme $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ consists of the four algorithms.

- $\mathcal{S}$, the setup algorithm, takes as input security parameter $k \in \mathbb{Z}$, and outputs system parameters params and the master-key master-key. $\mathcal{S}$ is a probabilistic algorithm. params has a description of a finite message space MSPC and a finite ciphertext space CSPC.
- $\mathcal{X}$, the extraction algorithm, takes as input params, master-key and an arbitrary $\mathsf{ID} \in \{0,1\}^*$, and outputs a private key d. ID is an arbitrary string and it is used as a public key. d is the corresponding private key (decryption key). This algorithm extracts a private key from ID.
- $\mathcal{E}$, the encryption algorithm, takes as input params, ID and $M \in \mathsf{MSPC}$, and outputs a ciphertext $C \in \mathsf{CSPC}$. Let $\mathsf{COIN}(k) \in \{0,1\}^*$ be a finite set. $\mathcal{E}$ is a probabilistic algorithm. $\mathcal{E}$ chooses a random bit string r from $\mathsf{COIN}(k)$. We write the result of running this algorithm as $\mathcal{E}(\mathsf{params}, \mathsf{ID}, M; r)$.
- $\mathcal{D}$, the decryption algorithm takes as input params, a ciphertext $C \in \mathsf{CSPC}$ and a private key d, and outputs the corresponding plaintext $M \in \mathsf{MSPC}$. We write the result of running this algorithm as $\mathcal{D}(\mathsf{params}, d, C)$.

These algorithms must satisfy the standard consistency constraint,

$$\forall M \in \mathsf{MSPC} : \mathcal{D}(\mathsf{params}, d, C) = M, \text{ where } C = \mathcal{E}(\mathsf{params}, \mathsf{ID}, M; r).$$

2.2 Notions and Notations

γ-uniformity. A property γ-uniformity is originally defined for conventional public key encryption schemes [13]. Here, we define γ-uniformity for IBE schemes.

Definition 1 (γ-uniformity)**.** *Let* $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ *be an IBE scheme. For a given* $\mathsf{ID} \in \{0,1\}^*$*, the corresponding decryption key* d*,* $M \in \mathsf{MSPC}$ *and* $C \in \mathsf{CSPC}$*, we define* $\gamma(\mathsf{params}, \mathsf{ID}, M, C) = \Pr[h \leftarrow_R \mathsf{COIN} : C = \mathcal{E}(\mathsf{params}, \mathsf{ID}, M; h)]$*. We say that* Π *is* γ*-uniform, if for any* $\mathsf{ID} \in \{0,1\}^*$*, any* $M \in \mathsf{MSPC}$ *and any* $C \in \mathsf{CSPC}$*, we have* $\gamma(\mathsf{params}, \mathsf{ID}, M, C) \leq \gamma$*.*

Negligible Function. We say a function $\epsilon : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every constant $c \geq 0$ there exits an integer k_c such that $\epsilon(k) < k^{-c}$ for all $k > k_c$.

2.3 IND-ID-CCA Security and IND-ID-CPA Security

Boneh and Franklin [8] defined the indistinguishability (IND) for IBE schemes. An IBE which is indistinguishable against adaptive chosen identity attack (IND-ID-CCA) has the strongest security and one which is indistinguishable against adaptive chosen identity and chosen plaintext attack (IND-ID-CPA) has even weaker security.

In their model, the two security notions were defined in the following games. In each game, there exist two parties, one malicious adversary and one neutral simulator.

2.3.1 IND-ID-CCA Game

This game is described as follows:

Setup: The challenger takes a security parameter k and runs the setup algorithm $\mathcal{S}$. It gives the adversary the resulting system parameters params and keeps the master-key to itself.

Phase 1: The adversary issues queries $q_1, \cdots, q_m$, where query q_i is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$. The challenger responds by running algorithm $\mathcal{X}$ to generate decryption key d_i which corresponds to the public key ID_i. It sends d_i to the adversary.
- Decryption query $\langle \mathsf{ID}_i, C_i \rangle$. The challenger responds by running algorithm $\mathcal{X}$ to generate the decryption key d_i corresponding to the public key ID_i. Then it runs algorithm $\mathcal{D}$ to decrypt the ciphertext C_i using the decryption key d_i. It sends the adversary the resulting plaintext.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once the adversary decides that Phase 1 is over it outputs a pair of messages $M_0, M_1 \in \mathsf{MSPC}$ of equal length and an ID^* on which it wishes to be challenged. ID^* must not have appeared in any Extraction query in Phase 1. The challenger picks a random bit $b \in \{0,1\}$ and sets $C = \mathcal{E}(\mathsf{params}, \mathsf{ID}, M_b)$. It sends C to the adversary.

Phase 2: The adversary issues more queries $q_{m+1}, \cdots, q_n$, where query q_i is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$ where $\mathsf{ID}_i \neq \mathsf{ID}^*$. The challenger responds as in Phase 1.
- Decryption query $\langle \mathsf{ID}_i, C_i \rangle$ where $\langle \mathsf{ID}_i, C_i \rangle \neq \langle \mathsf{ID}, C \rangle$. The challenger responds as in Phase 1.

These queries may be asked adaptively.

Guess: Finally, the adversary outputs a result $b' \in \{0,1\}$ and wins the game if $b' = b$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. The advantage of IND-ID-CCA adversary $\mathcal{A}$ is defined as follows: $Adv_{\mathcal{A}}(k) = |\Pr[b = b'] - \frac{1}{2}|$. The probability is over the random bits used by the challenger and the adversary.

Definition 2 (IND-ID-CCA). *We say that an IBE scheme is secure in the sense of* IND-ID-CCA *if* $Adv_{\mathcal{A}}$ *is negligible for any polynomial time algorithm* $\mathcal{A}$.

2.3.2 IND-ID-CPA Game

This game is similar to but easier than IND-ID-CCA game except that the adversary $\mathcal{A}$ is more limited. $\mathcal{A}$ cannot make decryption queries to simulator in neither Phase 1 nor Phase 2. The advantage of IND-ID-CPA adversary $\mathcal{A}$ is also defined as follows: $Adv_{\mathcal{A}}(k) = |\Pr[b = b'] - \frac{1}{2}|$. The probability is over the random bits used by the challenger and the adversary.

Definition 3 (IND-ID-CPA). *We say that an IBE scheme is secure in the sense of* IND-ID-CPA *if* $Adv_{\mathcal{A}}$ *is negligible for any polynomial time algorithm* $\mathcal{A}$.

2.4 OW-ID-PCA Security

A notion of security called onewayness against plaintext checking attack (OW-PCA) is an even weaker notion. The original concept is defined for conventional public key encryption schemes [18]. Roughly speaking, the security goal, onewayness, means that if given the encryption of a random plaintext the adversary cannot produce the plaintext in its entirety. In the plaintext checking attack model, the adversary can access to the PC(plaintext checking) oracle which takes as input a public key ID, a plaintext M and a ciphertext C and outputs "yes" or "no" whether C is the ciphertext of M.

Here, we define this security notion for IBE schemes in the following OW-ID-PCA game:

Setup: It is the same as the Setup step in the IND-ID-CCA game.

Phase 1: The adversary issues queries $q_1, \ldots, q_m$, where q_i is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$. It is the same as the Extraction query in the IND-ID-CCA game
- PC query $\langle \mathsf{ID}_i, M_i, C_i \rangle$. The PC oracle responds "yes" if C_i is the corresponding ciphertext of M_i. Otherwise the oracle responds "no".

These queries may be asked adaptively.

Challenge: Once the adversary decides that Phase 1 is over, it outputs a public key ID^* on which it wishes to be challenged. ID^* must not have appeared in any Extraction query in Phase 1. The challenger picks a random $M \in \mathsf{MSPC}$ and encrypts M using ID^* as the public key. It then sends the resulting ciphertext C to the adversary.

Phase 2: The adversary issues more Extraction queries and more PC queries. In this phase, the adversary must not ask Extraction query $\langle \mathsf{ID}^* \rangle$. The challenger responds as in Phase 1.

Guess: Finally, the adversary outputs a result $M' \in \mathsf{MSPC}$. The adversary wins the game if $M' = M$.

We refer to such an adversary $\mathcal{A}$ as an OW-ID-PCA adversary. $\mathcal{A}$'s advantage in attacking the scheme is defined as: $Adv_{\mathcal{A}}(k) = \Pr[M' = M]$. The probability is over the random bits used by the challenger and the adversary.

Definition 4 (OW-ID-PCA). *We say that an IBE scheme is secure in the sense of* OW-ID-PCA *if* $Adv_{\mathcal{A}}$ *is negligible for any polynomial time algorithm* $\mathcal{A}$.

3 The FOPKC Conversion for IBE

It is absorbing to find out an authentic way to enhance weak IBE schemes to strongly secure ones. In the conventional public key cryptosystems, there exist some such conversions, and the FOPKC [12, 14] is an example, besides, it is very efficient and achieves a tight reduction cost. Since an IBE is a different primitive from traditional public key encryption, one may think these conversions are not immediately a solution for IBE.

In this section, we investigate the FOPKC for IBE. More interesting result is that although a most significant merit of the FOPKC is its tight reduction cost for conventional public key encryption schemes, however, this merit could not hold if we apply the FOPKC *straightforwardly* into IBE case. In order to solve the problem, we slightly revise the plain FOPKC. The modification is quite simple in shape, but the right thing that we need.

3.1 The plain FOPKC

Let $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ be an IND-ID-CPA IBE. Then, we can construct an another IBE $\Pi_1 = \{\mathcal{S}_1, \mathcal{X}_1, \mathcal{E}_1, \mathcal{D}_1\}$ as follows: Let l_1 be a bit length of a plaintext of Π, l_2 be a bit length of a plaintext of Π_1 and $\mathsf{COIN}(k)$ be Π's coin-flipping space. The conversion is constructed in Table 1.

The Plain FOPKC
Setup $\mathcal{S}_1$: It is as $\mathcal{S}$. In addition, it picks a hash function $H : \{0,1\}^{l_2} \times \{0,1\}^{l_1-l_2} \rightarrow \mathsf{COIN}(k)$.
Extraction $\mathcal{X}_1$: It is as $\mathcal{X}$.
Encryption $\mathcal{E}_1$: It takes a system parameter params, an encryption key $\mathsf{ID} \in \{0,1\}^*$ and a message $M \in \{0,1\}^{l_2}$. $\mathcal{E}_1(\mathsf{params}, \mathsf{ID}, M; \sigma) = \mathcal{E}\big(\mathsf{params}, \mathsf{ID}, M\|\sigma; H(M, \sigma)\big)$, where σ is a randomly chosen $l_1 - l_2$ bit string.
Decryption $\mathcal{D}_1$: Let C be a ciphertext to decrypt. Algorithm $\mathcal{D}_1$ works in the following steps: 1. Computes $\mathcal{D}(\mathsf{params}, d, C) = M'$ and let $[M']^{l_2} = M$ and $[M']_{l_1-l_2} = \sigma$ where $[a]^b$ and $[a]_b$ denote the first and the last b bits of a string a, respectively. 2. Tests that $\mathcal{E}\big(\mathsf{params}, \mathsf{ID}, M||\sigma; H(M, \sigma)\big) = C$. If not, outputs "reject". 3. Outputs M as the decryption of C.

Table 1: The Plain FOPKC

Theorem 1. *Suppose the hash function H is the random oracle and Π is a γ-uniform IBE encryption scheme. Let $\mathcal{B}$ be an* IND-ID-CCA *adversary which has advantage $\epsilon(k)$ against Π_1 and it runs in time at most $t(k)$. Suppose $\mathcal{B}$ makes at most q_H H queries, q_E Extraction queries and q_D Decryption queries. Suppose that executing $\mathcal{E}$ once needs at most time τ Then there is an* IND-ID-CPA *adversary $\mathcal{A}$ which has advantage at least $(\epsilon(k) + \frac{1}{2} - q_H/(2^{l_1-l_2}))(1 - q_D\gamma) - \frac{1}{2}$ against Π. Its running time is $t(k) + q_H \cdot q_D \cdot \tau$.*

Proof. We show how to construct adversary $\mathcal{A}$ by using adversary $\mathcal{B}$ as an oracle. The challenger starts an IND-ID-CPA game by executing $\mathcal{S}$ and generates params and master-key. The master-key is kept secret by the challenger. $\mathcal{A}$ works by interacting with $\mathcal{B}$ in an IND-ID-CCA game as follows:

Setup: $\mathcal{A}$ gives params to $\mathcal{B}$.

Symbol	Event
Fail	$\mathcal{A}$ fails to answer a decryption query at some point during the game
SuccA	$\mathcal{A}$ wins the IND-ID-CPA game in the case that event Fail does not occur
SuccB	$\mathcal{B}$ wins the IND-ID-CCA game in the case that event Fail does not occur
Ask0	$\mathcal{B}$ queries $H(M_b, \sigma_b)$
Ask1	$\mathcal{B}$ queries $H(M_{\bar{b}}, \sigma_{\bar{b}})$

Table 2: Definitions of Events in Proof of plain FOPKC

Phase 1: Three sorts of queries are answered as follows:

H-queries: $\mathcal{A}$ maintains a list of tuples $\langle M_i, \sigma_i, h_i \rangle$ as explained below. We refer to this list as the H^{list}. The list is initially empty. When $\mathcal{B}$ queries $H(M_i, \sigma_i)$, $\mathcal{A}$ responds as follows:

1. If the query M_i, σ_i already appears on the H^{list} in a tuple $\langle M, \sigma_i, h_i \rangle$ then $\mathcal{A}$ responds with $H(M_i, \sigma_i) = h_i$.
2. Otherwise, $\mathcal{A}$ picks a random element h_i from $\mathsf{COIN}(k)$ of Π.
3. $\mathcal{A}$ adds the tuple $\langle M_i, \sigma_i, h_i \rangle$ to the H^{list} and returns h_i.

Extraction queries: Let $\langle \mathsf{ID}_i \rangle$ be an Extraction query issued by $\mathcal{B}$. $\mathcal{A}$ inputs $\langle \mathsf{ID}_i \rangle$ to its own extraction oracle and gets the corresponding decryption key d_i. $\mathcal{A}$ passes d_i to $\mathcal{B}$ as the answer of the query.

Decryption queries: Let $\langle \mathsf{ID}_i, C_i \rangle$ be a Decryption query issued by $\mathcal{B}$. $\mathcal{A}$ responds as follows:

1. Find a pair of tuples $\langle M, \sigma, h \rangle$ from the H^{list}, such that $\mathcal{E}(\mathsf{params}, \mathsf{ID}_i, M \| \sigma; h) = C_i$.
2. Outputs M if there exists such a pair of tuples, or outputs "reject" otherwise.

Challenge: Once $\mathcal{B}$ decides that Phase 1 is over it outputs a public key ID^* ($\mathsf{ID}^* \neq \mathsf{ID}_i$) and two messages M_0, M_1 on which it wishes to be challenged. $\mathcal{A}$ randomly chooses two $l_1 - l_2$ bit strings σ_0 and σ_1. $\mathcal{A}$ sends $\langle \mathsf{ID}^*, M_0 \| \sigma_0, M_1 \| \sigma_1 \rangle$ to the challenger.

The challenger picks a random bit $b \in \{0, 1\}$ and sets $C = \mathcal{E}(\mathsf{params}, \mathsf{ID}^*, M_b || \sigma_b)$. Then $\mathcal{A}$ gives C as the challenge to $\mathcal{B}$.

Phase 2: Three sorts of queries are answered as the same as in Phase 1.

Guess: Once $\mathcal{B}$ decides that Phase 2 is over it outputs a guess b'.

After $\mathcal{B}$ outputs the guess b', $\mathcal{A}$ outputs this bit b' as the answer of the IND-ID-CPA game.

In order to calculate the reduction cost, we first define the five events in Table 2.
Then, we have,

$$\begin{aligned}\Pr[\mathsf{SuccB}] &= \Pr[\mathsf{SuccB}|\mathsf{Ask0}]\Pr[\mathsf{Ask0}] \\ &+ \Pr[\mathsf{SuccB}|\neg\mathsf{Ask0} \wedge \mathsf{Ask1}]\Pr[\neg\mathsf{Ask0} \wedge \mathsf{Ask1}] \\ &+ \Pr[\mathsf{SuccB}|\neg\mathsf{Ask0} \wedge \neg\mathsf{Ask1}]\Pr[\neg\mathsf{Ask0} \wedge \neg\mathsf{Ask1}]\end{aligned}$$

and

$$\begin{aligned}\Pr[\mathsf{SuccA}] &= \Pr[\mathsf{SuccA}|\mathsf{Ask0}]\Pr[\mathsf{Ask0}] \\ &+ \Pr[\mathsf{SuccA}|\neg\mathsf{Ask0} \wedge \mathsf{Ask1}]\Pr[\neg\mathsf{Ask0} \wedge \mathsf{Ask1}] \\ &+ \Pr[\mathsf{SuccA}|\neg\mathsf{Ask0} \wedge \neg\mathsf{Ask1}]\Pr[\neg\mathsf{Ask0} \wedge \neg\mathsf{Ask1}].\end{aligned}$$

From the specification of $\mathcal{A}$, the following equations holds:

$$\Pr[\mathsf{SuccA}|\mathsf{Ask0}] = 1, \quad \Pr[\mathsf{SuccA}|\neg\mathsf{Ask0} \wedge \mathsf{Ask1}] = 0$$

and

$$\Pr[\mathsf{SuccB}|\neg\mathsf{Ask0} \wedge \neg\mathsf{Ask1}] = \Pr[\mathsf{SuccA}|\neg\mathsf{Ask0} \wedge \neg\mathsf{Ask1}].$$

Thus we have,

$$\begin{aligned}\Pr[\mathsf{SuccA}] - \Pr[\mathsf{SuccB}] &= (1 - \Pr[\mathsf{SuccB}|\mathsf{Ask0}])\Pr[\mathsf{Ask0}] \\ &\quad - \Pr[\mathsf{SuccB}|\neg\mathsf{Ask0} \wedge \neg\mathsf{Ask1}]\Pr[\neg\mathsf{Ask0} \wedge \neg\mathsf{Ask1}] \\ &\geq -\Pr[\neg\mathsf{Ask0} \wedge \mathsf{Ask1}].\end{aligned}$$

Since $\Pr[\neg\mathsf{Ask0} \wedge \mathsf{Ask1}] \leq q_H/2^{l_1-l_2}$, we have

$$\Pr[\mathsf{SuccA}] \geq \epsilon + \frac{1}{2} - \frac{q_H}{2^{l_1-l_2}}.$$

Next, we estimate $\Pr[\neg\mathsf{Fail}]$. The event Fail occurs only when $\mathcal{B}$ submits a Decryption query $\langle \mathsf{ID}, C\rangle$ such that $C = \mathcal{E}\big(\mathsf{params}, \mathsf{ID}, M\|\sigma; H(M,\sigma)\big)$ without asking $H(M,\sigma)$. This case happens with probability at most γ , and therefore, we have that $\Pr[\neg\mathsf{Fail}] \leq (1-\gamma)^{q_D} \simeq 1 - q_D\gamma$.

Hence, we have that

$$Adv_{\mathcal{A}}(k) \geq (\epsilon + \frac{1}{2} - \frac{q_H}{2^{l_1-l_2}})(1 - q_D\gamma) - \frac{1}{2}.$$

Finally, we estimate $\mathcal{A}$'s running time. Since in addition to $\mathcal{B}$'s running time, $\mathcal{A}$ has to run $\mathcal{E}$ for q_H times for responding to each Decryption query, $\mathcal{A}$'s running time is estimated as

$$t(k) + q_H \cdot q_D \cdot \tau$$

□

Discussion: Running time of $\mathcal{A}$ Bellare and Rogaway [4] proposed the notion of *exact security*, which says, a reduction is meaningful, if given an adversary $\mathcal{B}$ against Π, an adversary $\mathcal{A}$ can be constructed with essentially the same amount of time and success probability against Π_1. Now we focus on the running times of $\mathcal{A}$ and $\mathcal{B}$. As shown in Theorem 1, there exists a polynomial time reduction $\mathcal{B}$ to $\mathcal{A}$: in the reduction given above $\mathcal{A}$'s running time is estimated as

$$t(k) + q_H \cdot q_D \cdot \tau$$

where $t(k)$ is $\mathcal{B}$'s running time. Assuming that q_H and q_D are estimated as 2^{60} and 2^{40} respectively, $\mathcal{A}$ has to run $\mathcal{E}$ for 2^{100} times, which are computationally *infeasible* in practice. (Notice that a Decryption query requires on-line computation, while a G-query only requires off-line hash computation.)

3.2 The Modified FOPKC

Since for the plain FOPKC, the simulator cannot determine the ciphertext when it is asked a hash query, it has to do re-encryption for all the hash queries in the list to extract the plaintext. This is essentially why the reduction cost was bad. Here we shall slightly modify the plain FOPKC: The idea is very simple: just include the ID as part of a hash query. Though it is very simple, we shall go on to show it actually solve the above problem.

Let $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ be an IBE scheme which is secure in the sense of IND-ID-CPA. We denote the new encryption scheme as $\Pi_2 = \{\mathcal{S}_2, \mathcal{X}_2, \mathcal{E}_2, \mathcal{D}_2\}$. Let l_1 be a bit length of a plaintext of Π, l_2 be a bit length of a plaintext of Π_2 and $\mathsf{COIN}(k)$ be Π's coin-flipping space. The conversion is constructed in Table 3.

The Modified FOPKC
Setup $\mathcal{S}_2$: It is as $\mathcal{S}$. In addition, it picks a hash function $H : \{0,1\}^{l_2} \times \{0,1\}^{l_1-l_2} \times \{0,1\}^* \to \mathsf{COIN}(k)$.
Extraction $\mathcal{X}_2$: It is as $\mathcal{X}$.
Encryption $\mathcal{E}_2$: It takes a system parameter params, an encryption key $\mathsf{ID} \in \{0,1\}^*$ and a message $M \in \{0,1\}^{l_2}$. $\mathcal{E}_2(\mathsf{params}, \mathsf{ID}, M; \sigma) = \mathcal{E}\big(\mathsf{params}, \mathsf{ID}, M\|\sigma; H(M, \sigma, \mathsf{ID})\big)$, where σ is a randomly chosen $l_1 - l_2$ bit string.
Decryption $\mathcal{D}_2$: Let C be a ciphertext to decrypt. This algorithm works in the following steps: 1. Computes $\mathcal{D}(\mathsf{params}, d, C) = M'$ and let $[M']^{l_2} = M$, $[M']_{l_1-l_2} = \sigma$. 2. Tests that $\mathcal{E}\big(\mathsf{params}, \mathsf{ID}, M'; H(M, \sigma, \mathsf{ID})\big) = C$. If not, outputs "reject". 3. Outputs M as the decryption of C.

Table 3: The Modified FOPKC

Theorem 2. *Suppose that the hash function H is the random oracle and Π is γ-uniform IBE encryption scheme. Let $\mathcal{B}$ be an* IND-ID-CCA *adversary which has advantage $\epsilon(k)$ against Π_2 and it runs in time at most $t(k)$. Suppose $\mathcal{B}$ makes at most q_H H-queries, q_E Extraction queries and q_D Decryption queries. Suppose that encrypting one message needs time τ. Then there is an* IND-ID-CPA *adversary $\mathcal{A}$ which has advantage at least $(\epsilon(k) + \frac{1}{2} - q_H/(2^{l_1-l_2}))(1 - q_D\gamma) - \frac{1}{2}$ against Π.Its running time is $t(k) + q_H \cdot \tau$*

Proof. Similar strategy of the proof of Theorem 1 applies here, i.e., construct IND-ID-CPA adversary $\mathcal{A}$ for Pi by using IND-ID-CCA adversary $\mathcal{B}$ for Pi_2 as an oracle. Then what remains here is how to answer $\mathcal{B}$'s H-queries and Decryption-queries.

Setup: $\mathcal{A}$ gives params to $\mathcal{B}$.

Phase 1: Three sorts of queries are answered as follows:

H-queries: $\mathcal{A}$ maintains a list of tuples $\langle M_i, \sigma_i, \mathsf{ID}_i, h_i, C_i\rangle$ as explained below. We refer to this list as the H^{list}. The list is initially empty. When $\mathcal{B}$ queries $H(M_i, \sigma_i, \mathsf{ID}_i)$, $\mathcal{A}$ responds as follows:

1. If the query M_i, σ_i and ID_i already appears on the H^{list} in a tuple $\langle M_i, \sigma_i, \mathsf{ID}_i, h_i, C_i\rangle$ then $\mathcal{A}$ responds with $H(M_i, \sigma_i, \mathsf{ID}_i) = h_i$.
2. Otherwise, $\mathcal{A}$ picks a random element h_i from $\mathsf{COIN}(k)$.
3. $\mathcal{A}$ generates a ciphertext $C_i = \mathcal{E}(\mathsf{params}, \mathsf{ID}_i, M_i\|\sigma_i; h_i)$.
4. $\mathcal{A}$ adds the tuple $\langle M_i, \sigma_i, \mathsf{ID}_i, h_i, C_i\rangle$ to the H^{list} and responds to $\mathcal{B}$ with $H(M_i, \sigma_i, \mathsf{ID}_i) = h_i$.

Extraction queries: Let $\langle \mathsf{ID}_i\rangle$ be an Extraction query issued by $\mathcal{B}$. $\mathcal{A}$ inputs $\langle \mathsf{ID}_i\rangle$ to its own extraction oracle and gets the corresponding decryption key d_i. $\mathcal{A}$ passes d_i to $\mathcal{B}$ as the answer of the query.

Decryption queries: Let $\langle \mathsf{ID}_i, C_i\rangle$ be a decryption query issued by $\mathcal{B}$. $\mathcal{A}$ responds this query in the following steps:

1. Finds a tuple $\langle \sigma_j, M_j, \mathsf{ID}_j, g_j, C_j \rangle$ from the H^{list} such that $\mathsf{ID}_i = \mathsf{ID}_j$ and $C_i = C_j$.
2. Outputs M_j if there exists such a tuple, or outputs "reject" otherwise.

Challenge: Once $\mathcal{B}$ decides that Phase 1 is over it outputs a public key ID^* ($\mathsf{ID}^* \neq \mathsf{ID}_i$)and two messages M_0, M_1 on which it wishes to be challenged. $\mathcal{A}$ randomly choose two $l_1 - l_2$ bit strings σ_0 and σ_1. $\mathcal{A}$ sends $\langle \mathsf{ID}^*, M_0 \| \sigma_0, M_1 \| \sigma_1 \rangle$ to the challenger and receives a ciphertext C. Then, $\mathcal{A}$ gives C as the challenge to $\mathcal{B}$.

Phase 2: Three sorts of queries are answered as the same as in Phase 1.

Guess: Once $\mathcal{B}$ decides that Phase 2 is over it outputs a guess b'.

After $\mathcal{B}$ outputs the guess b', $\mathcal{A}$ outputs this bit b' as the answer of the IND-ID-CPA game.

The advantage of $\mathcal{A}$ can be evaluate in the same way as in Theorem 1, so we omit to describe the detail of the evaluation here.

Finally, we estimate the running time of $\mathcal{A}$. Since adding the running time of $\mathcal{B}$, $\mathcal{A}$ has to run $\mathcal{E}$ once when a new H-query is asked. Therefore, $\mathcal{A}$'s running time is estimated as $t(k) + q_H \cdot \tau$.

□

3.3 Comparison: Running Time of $\mathcal{A}$ in The Plain FOPKC and The Modified FOPKC

We compare the running times of simulators for Π_1 and Π_2. In this comparison, we especially focus on times to run the encryption algorithm $\mathcal{E}$ which is required for each simulation. It is believed that if a simulator has to run $\mathcal{E}$ for more than 2^{80} times, then it does not properly work in a realistic time. Now, we have that

$$\#\mathcal{E}(\Pi_1)(\sim 2^{100}) \gg 2^{80} \gg \#\mathcal{E}(\Pi_2)(\sim 2^{60})$$

where $\#\mathcal{E}(\cdot)$ denotes the times to run $\mathcal{E}$ in the simulation. This implies that the running time of the simulator for Π_2 is considered realistic.

4 REACT Conversion for IBE

REACT conversion [18] was originally designed for (OW-PCA secure) traditional public key cryptosystems, to have CCA security. Again, it was not known if REACT can be applied to IBE generically. We investigate the fact in this section. Interestingly plain REACT is not only effective for IBE, but also gives a tight reduction cost, as it does for traditional public key cryptosystems.

4.1 The Plain REACT for IBE

Let $\Pi = \{\mathcal{S}, \mathcal{X}, \mathcal{E}, \mathcal{D}\}$ be an OW-ID-PCA IBE. Let MSPC be a message space of Π and CSPC be a ciphertext space of Π. Then we can construct another IBE $\Pi_3 = \{\mathcal{S}_3, \mathcal{X}_3, \mathcal{E}_3, \mathcal{D}_3\}$ which is secure against IND-ID-CCA. Let MSPC′ be a message space of Π_3 and MSPC′ be a ciphertext space of Π_3. A ciphertext of Π_3 consists of three components c_1, c_2 and c_3. We denote the bit length of these components l_1, l_2 and l_3 respectively. The definition of Π_3 is as follows in Table 4.

Theorem 3. *Suppose the hash functions G and H are random oracles. Let $\mathcal{B}$ be an* IND-ID-CCA *adversary which has advantage $\epsilon(k)$ against Π_3 and its running time is at most $t(k)$. Suppose $\mathcal{B}$ makes at most q_G G-queries, q_H H-queries, q_E extraction queries and q_D decryption queries. Then there is an* OW-ID-PCA *adversary $\mathcal{A}$ which has advantage $2\epsilon(k) - q_D(\frac{1}{2^{l_2}} + \frac{1}{2^{l_3}})$. Its running time is $t(k) + (q_G + q_H) \cdot O(1)$.*

The Plain REACT for IBE
Setup $\mathcal{S}_3$: It is as $\mathcal{S}$. In addition it picks two hash functions: $G : \mathsf{MSPC} \rightarrow \{0,1\}^{l_2}, H : \mathsf{MSPC} \times \mathsf{MSPC}' \times \{0,1\}^{l_1} \times \{0,1\}^{l_2} \rightarrow \{0,1\}^{l_3}$.
Extraction $\mathcal{X}_3$: It is as $\mathcal{X}$.
Encryption $\mathcal{E}_3$: For any massage $M \in \mathsf{MSPC}'$ and random values $R \in \mathsf{MSPC}$, it gets $c_1 = \mathcal{E}(\mathsf{paramas}, \mathsf{ID}, R; r)$, then it computes $K = G(R), c_2 = K \oplus M, c_3 = H(R, M, c_1, c_2)$. The ciphertext consists of the triple $C = (c_1, c_2, c_3)$.
Decryption $\mathcal{D}_3$: We assume that the ciphertext to decrypt is $C = (c_1, c_2, c_3)$. First it decrypts c_1 and gets R. Then it computes $K = G(R)$ and $M = c_2 \oplus K$. It returns M if $c_3 = H(R, M, c_1, c_2)$. Otherwise , it outputs "Reject".

Table 4: The Plain REACT for IBE

Proof. We show how to construct adversary $\mathcal{A}$ by using adversary $\mathcal{B}$ as an oracle. The challenger starts an OW-ID-PCA game by executing $\mathcal{S}$ and generates params and master-key. The master-key is kept secret by the challenger. $\mathcal{A}$ works by interacting with $\mathcal{B}$ in an IND-ID-CCA game as follows:

Setup: $\mathcal{A}$ gives params to $\mathcal{B}$.

Phase 1: Four sorts of queries are answered as follows:

G**-queries:** $\mathcal{A}$ maintains a list of tuples $\langle R_i, K_i \rangle$ as explained below. We refer to this list as G^{list}. The list is initially empty. When $\mathcal{B}$ asks $G(R_i)$, $\mathcal{A}$ responds as follows:

1. If query R_i already appears on the G^{list} in a tuple $\langle R_i, K_i \rangle$ then $\mathcal{A}$ responds with $G(R_i) = K_i$.
2. Otherwise, $\mathcal{A}$ picks a random element $K_i \in \{0,1\}^{l_2}$.
3. $\mathcal{A}$ adds the tuple $\langle R_i, K_i \rangle$ to the G^{list} and returns K_i.

H**-queries:** $\mathcal{A}$ maintains a list of tuples $\langle R_i, M_i, c_1, c_2, c_3 \rangle$. We refer this list as the H^{list}. The list is initially empty. When $\mathcal{B}$ queries $H(R, M, c_1, c_2)$, $\mathcal{A}$ responds as follows:

1. If the query $\langle R, M, c_1, c_2 \rangle$ is already appears on the H^{list} in the tuple $\langle R, M, c_1, c_2, c_3 \rangle$ then $\mathcal{A}$ responds with $H(R, M, c_1, c_2) = c_3$.
2. Otherwise, $\mathcal{A}$ randomly picks a string c_3 from $\{0,1\}^{l_3}$.
3. $\mathcal{A}$ adds the tuple $\langle R, M, c_1, c_2, c_3 \rangle$ to the H^{list} and returns c_3.

Extraction queries: Let $\langle \mathsf{ID}_i \rangle$ be an Extraction query issued by $\mathcal{B}$. $\mathcal{A}$ inputs $\langle \mathsf{ID}_i \rangle$ to its own extraction oracle and receives the corresponding decryption key d_i. $\mathcal{A}$ sends the key d_i to $\mathcal{B}$ as the answer of the query.

Decryption queries: Let $\langle \mathsf{ID}_i, c_1, c_2, c_3 \rangle$ be a Decryption query issued by $\mathcal{B}$. $\mathcal{A}$ responds as follows:

1. $\mathcal{A}$ picks up a tuple $\langle R', M', c_1', c_2', c_3' \rangle$ from H^{list} such that $c_3 = c_3'$.
2. $\mathcal{A}$ computes $K' = G(R')$.
3. Checks if $c_2 = M' \oplus K'$. If this holds, $\mathcal{A}$ queries $\langle \mathsf{ID}_i, R', c_1 \rangle$ to the PC oracle.

4. If the PC oracle answers "yes", $\mathcal{A}$ returns M' to $\mathcal{B}$. Otherwise, $\mathcal{A}$ outputs "reject".

Challenge: Once $\mathcal{B}$ decides that Phase 1 is over it outputs a public key ID^* ($\mathsf{ID}^* \neq \mathsf{ID}_i$) and two message M_0, M_1 on which it wishes to be challenged. $\mathcal{A}$ sends ID^* to the challenger and receives a ciphertext C^*. $\mathcal{A}$ generates a l_2 bit random string c_2 and a l_3 bit random string c_3. $\mathcal{A}$ gives $\langle C^*, c_2, c_3 \rangle$ to $\mathcal{B}$ as a challenge ciphertext.

Phase 2: Four sorts of queries are answered as the same as in Phase 1.

Guess: Once $\mathcal{B}$ decides that Phase 2 is over it outputs a guess b'.

After $\mathcal{B}$ outputs a guess b', $\mathcal{A}$ picks all Rs which appear in tuples on the G^{list} and the H^{list}. For each R, $\mathcal{A}$ queries $\langle \mathsf{ID}^*, R, C^* \rangle$ to PC oracle. If PC oracle returns "yes", $\mathcal{A}$ outputs the R as the answer of OW-ID-PCA game.

To estimate the advantage of $\mathcal{A}$, we define the following four events in Table 5.

Symbol	Event
SuccA	$\mathcal{A}$ wins the OW-ID-PCA game
SuccB	$\mathcal{B}$ wins the IND-ID-CCA game
AskB	$\mathcal{B}$ asks a query for $G(R^*)$ or $H(R^*, M_b, c_1, c_2)$ at some point during the game
Fail	the simulation fails before the event AskB occurs

Table 5: Definitions of Events in Proof of REACT for IBE

Then we take the same discussion as in the proof of Theorem 1 and we have that

$$\Pr[\mathsf{SuccB}|\neg\mathsf{Fail}]\Pr[\neg\mathsf{Fail}] \geq \epsilon(k) + \frac{1}{2} - \Pr[\mathsf{Fail}].$$

Since

$$\Pr[\mathsf{SuccB}|\neg\mathsf{Fail} \wedge \neg\mathsf{AskB}] = \frac{1}{2},$$

we also have

$$\begin{aligned} \Pr[\mathsf{SuccB}|\neg\mathsf{Fail}] &= \Pr[\mathsf{SuccB}|\neg\mathsf{Fail} \wedge \mathsf{AskB}]\Pr[\mathsf{AskB}] + \frac{1}{2}(1 - \Pr[\mathsf{AskB}]) \\ &\leq \frac{1}{2}\Pr[\mathsf{AskB}] + \frac{1}{2}. \end{aligned}$$

Hence, we have

$$(\frac{1}{2}\Pr[\mathsf{AskB}] + \frac{1}{2})\Pr[\neg\mathsf{Fail}] \geq \epsilon(k) + \frac{1}{2} - \Pr[\mathsf{Fail}],$$

and therefore,

$$\Pr[\mathsf{AskB}] \geq 2\epsilon(k) - \Pr[\mathsf{Fail}].$$

Next, we estimate $\Pr[\mathsf{Fail}]$. The event Fail occurs only in either

Case 1. $\mathcal{B}$ submits a Decryption query $\langle \mathsf{ID}, c_1, G(R) \oplus M, c_3 \rangle$ such that $c_1 = \mathcal{E}(\mathsf{params}, \mathsf{ID}, R; r)$ and $c_3 = H(R, M, c_1, G(R) \oplus M)$ without asking $G(R)$, or

Case 2. $\mathcal{B}$ submits a Decryption query $\langle \mathsf{ID}, c_1, c_2, H(R, M, c_1, c_2) \rangle$ without asking $H(R, M, c_1, c_2)$.

Case 1 and **2** happen with probability at most 2^{-l_2} and 2^{-l_3}, respectively, and therefore, we have that

$$\Pr[\mathsf{Fail}] \leq 1 - (1 - \frac{1}{2^{l_2}} - \frac{1}{2^{l_3}})^{q_D} \simeq q_D(\frac{1}{2^{l_2}} + \frac{1}{2^{l_3}}).$$

If $\mathcal{B}$ wins the IND-ID-CCA game, then $\mathcal{A}$ also win the OW-ID-PCA game. Therefore, $\Pr[\mathsf{SuccA}] \geq \Pr[\mathsf{SuccB}]$.

Hence, we have that

$$Adv_{\mathcal{A}}(k) \quad = \quad \Pr[\mathsf{SuccA}] \geq \Pr[\mathsf{SuccB}] \simeq 2\epsilon(k) - q_D(\frac{1}{2^{l_2}} + \frac{1}{2^{l_3}}).$$

Finally, we estimate a running time of $\mathcal{A}$. Addition to the running time of $\mathcal{B}$, $\mathcal{A}$ has to answer the G and H queries. Thus $\mathcal{A}$'s running time is $t(k) + (q_H + q_G) \cdot O(1)$.

□

4.2 The Reduction Efficiency of REACT for IBE

As shown in Section 3.2, the reduction cost of the plain FOPKC is inefficient. Using our simple technique which we put ID to a part of the hash function's inputs, the reduction cost could significantly decreases.

Unlike the plain FOPKC, the plain REACT already gives a tight reduction cost for IBE. We remark that this is mainly caused by the PC oracle, which implicitly handles the ID by its definition. The significant differences of reduction costs may indicate a separation between these two attack models: CPA and PCA.

5 Numerical Explanation

In this section, we evaluate how much better on the cost of security reduction the modified FOPKC is than the plain FOPKC by numerical explanation and show what this result means in the real world. As shown in Theorem 1, the plain FOPKC gives a polynomial time reduction and, looking at the coefficient of ϵ, the reduction seems tight, which means that the adversary's advantage against the underlying scheme and that of transformed scheme are close. Therefore, at a glance, merit of the modified FOPKC seems not considerable.

However, we notice that the other terms except for ϵ term have a significant influence on the reduction cost in the plain FOPKC. So, here, we compare the plain FO and our modified FOPKC by strictly estimating T' $(= t'/\epsilon')$ where T' is intuitively the average time for an adversary to succeed in the attack for the basic IBE scheme. We let Boneh and Franklin's scheme (BF-IBE) [8] be the underlying IBE scheme.

5.1 Parameter Setting

Let T' and T be t'/ϵ' and t/ϵ, respectively, where T' and T are the expected computational times to succeed in breaking the underlying IND-ID-CPA IBE scheme and the transformed IND-ID-CCA IBE scheme, respectively, assuming that a t'-time adversary can break the underlying IBE with advantage ϵ' and a t-time adversary can break the transformed IBE with advantage ϵ.

If the value T' is close to T, the reduction is said to be tight. On the other hand, if T' is much larger than T, the reduction is not tight and the adversary might not break the underlying IBE scheme in practical time. We derive the relation between T' and T of both the plain and the modified FOPKC.

In our estimation, we let $q_H = 2^{60}, q_D = 2^{40}$ and $\gamma = 2^{-160}$. (Notice that γ is identical to the inverse of the order of the underlying group in BF-IBE [8].) We set that $l_1 - l_2 = 140$ which is the bit length of the random coin of both FOPKC conversions.

In the evaluation, we consider the case of $\epsilon = 2^{-10}$. Encrypting one message needs one pairing computation in BF-IBE scheme and this is the dominant part. The running time of fastest pairing algorithms in software and hardware are about

$$\begin{cases} 4.33 \text{ miliseconds} & \text{in software implementation (AthlonXP 2GHz)[3]} \\ 0.85 \text{ miliseconds} & \text{in hardware implementation (FPGA 15MHz)[16].} \end{cases}$$

Thus, the running time of the encryption function τ is set to those values.

5.2 T' of The Plain FOPKC

In the above setting, we evaluate T' of the plain FOPKC for BF-IBE:

$$\begin{aligned} T' &\leq \frac{t + q_H q_D \tau}{(\epsilon + 1/2 - q_H/2^{(l_1 - l_2)})(1 - q_D\gamma) - 1/2} \simeq \frac{t + 2^{100} \times \tau}{(\epsilon + 1/2 - 2^{60}/2^{140})(1 - 2^{40} \cdot 2^{-160}) - 1/2} \\ &\simeq \frac{t + 2^{100} \times \tau}{\epsilon - 2^{-80}} \simeq T + 2^{110} \times \tau. \end{aligned}$$

The additional costs to break BF-IBE scheme $(T' - T)$ are

$$\begin{cases} 1.00 \times 2^{102} & \text{seconds } (\simeq 1.32 \times 10^{24} \text{ years}) \text{ in software implementation} \\ 0.85 \times 2^{100} & \text{seconds } (\simeq 0.11 \times 10^{24} \text{ years}) \text{ in hardware implementation,} \end{cases}$$

respectively. Each of them needs too long time to break BF-IBE, of course, it is impossible to calculate in the real world.

5.3 T' of The Modified FOPKC

In the above setting, we evaluate T' of the modified FOPKC for BF-IBE:

$$\begin{aligned} T' &\leq \frac{t + q_H \tau}{(\epsilon + 1/2 - q_H/2^{l_1 - l_2})(1 - q_D\gamma) - 1/2} \simeq \frac{t + 2^{60} \times \tau}{(\epsilon + 1/2 - 2^{60}/2^{140})(1 - 2^{40} \cdot 2^{-160}) - 1/2} \\ &\simeq \frac{t + 2^{60} \times \tau}{\epsilon - 2^{-80}} \simeq T + 2^{70} \times \tau. \end{aligned}$$

The additional costs to break BF-IBE $(T' - T)$ are

$$\begin{cases} 1.00 \times 2^{62} & \text{seconds } (\simeq 1.33 \times 10^{9} \text{ years}) \text{ in software implementation} \\ 0.85 \times 2^{60} & \text{seconds } (\simeq 2.83 \times 10^{8} \text{ years}) \text{ in hardware implementation,} \end{cases}$$

respectively. In this case, the additional cost is much smaller than that in the plain FOPKC case.

The additional cost is almost for pairing calculations in encryptions and these operations are easily parallelized. Nowadays, it is not difficult to gather computing resources like millions order PCs [1] or to produce a number of specialized IC chips. For example, consider the case that an adversary can use a million PCs, that is, it is the software implementation case, the cost is 1.33×10^3 years at present. By Moore's law, this cost will be about 1.30 years in 15 years' time. Thus, it can be said

that this additional cost will be really feasible to compute in the near future even in the software implementation case.

Due to the above discussion, we see that if there exists an adversary who can break the modified FOPKC in a realistic time, then it is also possible to break the underlying IBE scheme in almost the same computational time. On the other hand, it is not clear whether the plain FOPKC provides the same level of security or not. Consequently, we can say that the modified FOPKC achieves *exact security* in a strict sense while the plain FOPKC does not.

	$T - T'$ of The Plain FOPKC case	$T - T'$ of The Modified FOPKC case
Software Implementation	1.32×10^{24} years	1.33×10^{9} years
Hardware Implementation	0.11×10^{23} years	2.83×10^{8} years
Application of Moore's law for SW Impl. using 10^6 PCs in 15 years after	1.32×10^{15} years	1.30 years

Table 6: The Result of the Numerical Explanation

References

[1] RSA security – cryptographic challenges. `http://www.rsasecurity.com/rsalabs/node.asp?id=2091`, RSA Security, available on Feb. 11, 2005.

[2] N. Attrapadung, Y. Cui, D. Galindo, G. Hanaoka, I. Hasuo, H. Imai, K. Matsuura, P. Yang, and R. Zhang. Relations among notions of security for identity based encryption schemes. In *Latin American Theoretical Informatics (LATIN '06)*, volume 3887 of *LNCS*, pages 130–141. Springer, 2006.

[3] P.S.L.M. Barreto. A note on effcient computation of cube roots in characteristic 3. Cryptology ePrint Archive, Report 2004/305, 2004. `http://eprint.iacr.org/2004/305`.

[4] M. Bellare and P. Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416. Springer, 1996.

[5] D. Boneh and X. Boyen. Efficient selective-ID identity based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT '04*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

[6] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology - CRYPTO '04*, volume 3152 of *LNCS*, pages 443–459. Springer, 2004.

[7] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology - CRYPTO '01*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[8] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. Full version of [8].

[9] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology - EUROCRYPT '04*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.

[10] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proc. of the 8th IMA international conference on cryptography and coding*, volume 2260 of *LNCS*, pages 360–363. Springer, 2001.

[11] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[12] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *Proc. of Public Key Cryptography 1999*, volume 1560 of *LNCS*, pages 53–68. Springer, 1999.

[13] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.

[14] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. *IEICE Transactions Fundamentals*, E83-A(1):24–32, 2000. Full version of [13].

[15] D. Galindo. Boneh-Franklin Identity Based Encryption Revisited. In *Proc. of 32nd ICALP*, volume 3580 of *LNCS*, pages 791–802. Springer, 2005.

[16] T. Kerins, W.P. Marnane, E.M. Popovici, and P.S.L.M. Barreto. Efficient hardware for the Tate pairing calculation in characteristic three. In *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *LNCS*, pages 412–426. Springer, 2005. Presentation file is available from `http://islab.oregonstate.edu/ches/ches2005/presentations/`.

[17] B. Libert and J.J. Quisquater. Identity based encryption without redundancy. In *Proc. of ACNS '05*, volume 3531 of *LNCS*, pages 285–300. Springer, 2005.

[18] T. Okamoto and D. Pointcheval. REACT: rapid enhanced-security asymmetric cryptosystem transform. In *Topics in Cryptology – CT-RSA '01*, volume 2020 of *LNCS*, pages 159–174. Springer, 2001.

[19] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO '84*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[20] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT '05*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

[21] P. Yang, T. Kitagawa, G. Hanaoka, R. Zhang, K. Matsuura, and H. Imai. Applying fujisaki-okamoto to identity-based encryption. In *LNCS*, volume 3857, pages 183–192, 2006.