Advance Access publication on 13 July 2016
doi:10.1093/comjnl/bxw033

CCA2 Attack and Modification of Huang *et al.*'s Public Key Encryption with Authorized Equality Test

HYUNG TAE LEE[1], SAN LING[1], JAE HONG SEO[2*], AND HUAXIONG WANG[1]

[1]*Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link, 637371, Singapore*
[2]*Department of Mathematics, Myongji University, Yongin, Gyeonggi-do, 17058, Republic of Korea*
[*]*Corresponding author: jaehongseo@mju.ac.kr*

In this article, we identify a flaw in Huang *et al.*'s public key encryption with authorized equality test (*The Computer Journal*, 2015). More precisely, we point out that the proof of the indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) security for their scheme has a serious flaw. We illustrate this flaw by presenting a polynomial time CCA2 attack on their scheme. We also provide a solution to correct this flaw by modifying their scheme slightly. Our solution is quite efficient because it provides security against CCA2 attack by exploiting only the hash computation of a two times longer input without any increase in the sizes of ciphertexts and warrants.

Keywords: public key encryption; authorized equality test; adaptive chosen ciphertext attack

Received 17 March 2016; accepted 3 May 2016
Handling editor: Liqun Chen

1. INTRODUCTION

Public key encryption with equality test (PKEET) was first introduced by Yang *et al.* [1] as an encryption scheme that is able to check whether two ciphertexts under different public keys as well as the same public key contain the same message. This feature can be applied to various scenarios in practice, such as keyword search on encrypted data, to facilitate efficient management by partitioning encrypted data in the cloud, personal health record systems and other systems, and several improved PKEET schemes have been proposed [2–6] to achieve better performance or to support additional functionalities. Recently, Huang *et al.* [7] proposed a public key encryption with authorized equality test (PKE-AET), which allows an authorized tester to perform equality tests on all of the receiver's ciphertexts or a specified ciphertext with respect to the warrant issued by the receiver. It was claimed that their scheme achieves one-wayness under adaptive chosen ciphertext attack against Type I adversaries who have a warrant and the indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) against Type II adversaries who do not have a warrant.

In this article, we identify a flaw in the IND-CCA2 security of Huang *et al.*'s PKE-AET. CCA security games for public key encryption allow attackers to request queries to decryption oracles. In particular, contrary to CCA1, attackers can utilize the challenge ciphertext for queries to the decryption oracle after the challenge phase in the IND-CCA2 security game. We note that the authors of [7] did not carefully consider CCA2 attacks in the proof of the IND-CCA2 security of their scheme. The situation is actually worse because we also illustrate a polynomial time CCA2 attack on their scheme, which shows that the fault in their proof causes an actual threat on the security of the scheme.

Let us briefly explain the idea of our CCA2 attack. The plaintext space of Huang *et al.*'s scheme is a cyclic group $\mathbb{G}$ with a generator g of order q. A ciphertext of a message m has the following form:

$$\begin{aligned} c &= (c_1, c_2, c_3) \\ &= (g^{\gamma}, m^{\gamma} \cdot \mathcal{H}_1((g^{\alpha})^{\gamma}), (m\|\gamma) \oplus \mathcal{H}_2((g^{\beta})^{\gamma})), \end{aligned}$$

where (g^{α}, g^{β}) is the receiver's public key, $\mathcal{H}_1$ and $\mathcal{H}_2$ are hash functions and γ is an element in $\mathbb{Z}_q^*$ chosen by the encryption algorithm. The authors of [7] seem to consider that it is difficult to obtain $\mathcal{H}_1((g^{\alpha})^{\gamma})$ and $\mathcal{H}_2((g^{\beta})^{\gamma})$ values from $(g, g^{\gamma}, g^{\alpha})$ and $(g, g^{\gamma}, g^{\beta})$, respectively, if the computational Diffie–Hellman (CDH) problem is intractable.

However, the adversary chooses two messages with their discrete logarithm to the base g in the IND-CCA2 security game, so he can recover $\mathcal{H}_1((g^{\alpha})^{\gamma})$ from c_2 using c_1 and the discrete logarithms of the challenge messages with high probability even though he does not know γ. Furthermore, since he does not know the exact value of γ, he cannot obtain the exact value of $\mathcal{H}_2((g^{\beta})^{\gamma})$. However, he can obtain $(0\|\gamma) \oplus \mathcal{H}_2((g^{\beta})^{\gamma})$ with high probability, and this is sufficient to generate a new valid ciphertext. Therefore, their intuitive method is insufficient to prevent CCA2 attacks, and our suggested attack works well.

We also provide a rectified version of their scheme to achieve the IND-CCA2 security. Our modification is quite simple, and the role of the modified part is clear, where the key feature of our attack is to generate a valid ciphertext from the challenge ciphertext without hash queries. Therefore, we make a link between c_2 and c_3 to prevent reuse of part of the challenge ciphertext for queries to the decryption oracle. Thus, we modify the encryption algorithm for a message m by inserting m^{γ} into the input of the hash function for the third component of a ciphertext. As a consequence, we only change the third component of the ciphertext of a message m to

$$(m\|\gamma) \oplus \mathcal{H}_2((g^{\beta})^{\gamma}\|m^{\gamma})$$

such that c_2 and c_3 share m^{γ}. Therefore, the adversary is forced to request the hash value of the hash function $\mathcal{H}_2$ to the hash oracle in order to generate another valid ciphertext, and thus we can correct their scheme and its security proof.

We note that our solution additionally requires only one hash computation of a two times longer input for both the encryption and decryption algorithms. The output sizes of all algorithms including the hash function $\mathcal{H}_2$ are the same as before, so the sizes of ciphertexts and warrants are also exactly the same as before. Therefore, our solution has practically the same performance as the original scheme in terms of its efficiency and storage.

Organization of the Article. In Section 2, we introduce Huang *et al.*'s PKE-AET scheme and its security argument. Section 3 presents our CCA2 attack against their scheme. We provide a modification of their scheme to achieve the IND-CCA2 security in Section 4.

2. HUANG *ET AL.*'S PKE-AET SCHEME

In this section, we look at Huang *et al.*'s PKE-AET scheme [7] and security definitions of PKE-AET scheme by focusing on the IND-CCA2 security against Type II adversaries who have no warrant for equality test.

Notation. Throughout the article, $a \xleftarrow{\$} A$ denotes that a is uniformly and randomly chosen from A for a set A and an element a in the set A. For a real number a, $\lceil a \rceil$ denotes the smallest integer that is larger than or equal to a. $\oplus$ stands for the bitwise XOR operation and $\|$ represents for concatenation.

2.1. The description of Huang *et al.*'s scheme

In this section, we provide the system model and the description of Huang *et al.*'s PKE-AET [7].

The system of PKE-AET consists of a sender, a receiver and a tester: A sender encrypts a message using a receiver's public key and passes a ciphertext to the receiver. After receiving the ciphertext, the receiver may decrypt it using his/her secret key. Once the receiver issues a warrant to a tester by encrypting with a tester's public key, the tester obtains the issued warrant by decrypting it, can verify it and may perform equality test with it. We note that there are two types of warrants in Huang *et al.*'s PKE-AET; a receiver's warrant that enables to check equality for all of receiver's ciphertexts and a ciphertext's warrant that enables to check equality for the specified ciphertext.

The description of Huang *et al.*'s scheme is as follows:

- Setup(λ): On input a security parameter λ, it selects two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order $q = q(\lambda)$ and a bilinear map e: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let g be a generator of $\mathbb{G}$. Let $\mathcal{H}_1$: $\mathbb{G} \to \mathbb{G}$, $\mathcal{H}_2$: $\mathbb{G} \to \{0,1\}^{q+\lceil \log_2 q \rceil}$ and $\mathcal{H}_3$: $\mathbb{G} \to \{0,1\}^{\lceil \log_2 q \rceil}$ be cryptographic hash functions. It outputs a public parameter
$$pp = (\mathbb{G}, \mathbb{G}_T, e, g, q, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3).$$
- KeyGen(pp): On input pp, it selects random elements $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q^*$ and outputs a public key $pk = (g^{\alpha}, g^{\beta})$ and a secret key $sk = (\alpha, \beta)$.
- Enc(pk,m): On input the public key $pk = (g^{\alpha}, g^{\beta})$ and a message $m \in \mathbb{G}$, it selects a random element $\gamma \xleftarrow{\$} \mathbb{Z}_q^*$ and outputs
$$c = (g^{\gamma}, m^{\gamma} \cdot \mathcal{H}_1(g^{\alpha\gamma}), (m\|\gamma) \oplus \mathcal{H}_2(g^{\beta\gamma})).$$
We note that the plaintext space of this algorithm is $\mathbb{G}$.
- Dec(sk,c): On input the secret key $sk = (\alpha, \beta)$ and a ciphertext $c = (c_1, c_2, c_3)$,
  (1) Compute $m'\|\gamma' = c_3 \oplus \mathcal{H}_2(c_1^{\beta})$.
  (2) Check whether $c_1 = g^{\gamma'}$ and $c_2 = m'^{\gamma'} \cdot \mathcal{H}_1(c_1^{\alpha})$.
  (3) Output m' if both of the above verifications pass. Otherwise, output $\perp$.
- Authorization:
  – Aut$_r$(sk,pk_t): On input the user's secret key $sk = (\alpha, \beta)$ and the tester's public key $pk_t = (g^{\alpha_t}, g^{\beta_t})$,
    (1) Select a random element $\theta \xleftarrow{\$} \mathbb{Z}_q^*$ and compute a user's warrant $rw = (g^{\theta}, \alpha \oplus \mathcal{H}_3(g^{\beta_t\theta}))$.
    (2) Send rw to the tester.

– $\mathrm{Aut}_c(sk, pk_t, c)$: On input the user's secret key $sk = (\alpha, \beta)$, the tester's public key $pk_t = (g^{\alpha_t}, g^{\beta_t})$ and a ciphertext $c = (c_1, c_2, c_3)$,
  (1) Check the validity of c by decrypting c using the user's secret key. If c is not valid, output $\perp$.
  (2) Otherwise, select a random element $\theta' \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and compute a ciphertext's warrant $cw = (g^{\theta'}, c_1^{\alpha} \cdot \mathcal{H}_1(g^{\beta_t \theta'}))$.
  (3) Send cw to the tester.

- Verification:
  – $\mathrm{Ver}_r(rw, pk, sk_t)$: On input the receiver's warrant $rw = (w_1, w_2)$, the receiver's public key $pk = (g^{\alpha}, g^{\beta})$, and the tester's secret key $sk_t = (\alpha_t, \beta_t)$,
    (1) Compute $\alpha^* = w_2 \oplus \mathcal{H}_3(c_1^{\beta_t})$.
    (2) If $g^{\alpha} = g^{\alpha^*}$, output 1. Otherwise, output 0.
  – $\mathrm{Ver}_c(cw, pk, sk_t, c)$: On input the ciphertext's warrant $cw = (w_1, w_2)$, the receiver's public key $pk = (g^{\alpha}, g^{\beta})$, the tester's secret key $sk_t = (\alpha_t, \beta_t)$ and a ciphertext $c = (c_1, c_2, c_3)$,
    (1) Compute $z = w_2 / \mathcal{H}_1(w_1^{\beta_t})$.
    (2) Check whether $e(g, z) = e(c_1, g^{\alpha})$. If it holds, output 1; otherwise, output 0.
- $\mathrm{Test}(c, w, c', w', sk_t)$: On input two ciphertexts $c = (c_1, c_2, c_3)$, $c' = (c_1', c_2', c_3')$, two warrants $w \in \{rw, cw\}$, $w' \in \{rw', cw'\}$ and the tester's secret key $sk_t = (\alpha_t, \beta_t)$, it performs as follows:
  (1) Compute z on input c and w by the following:
    (a) For a receiver's warrant $rw = (w_1, w_2)$, compute $z = c_2 / \mathcal{H}_1(c_1^{w_2 \oplus \mathcal{H}_1(w_1^{\beta_t})})$.
    (b) For a ciphertext's warrant $cw = (w_1, w_2)$, compute $z = c_2 / \mathcal{H}_1(w_2 / \mathcal{H}_1(w_1^{\beta_t}))$.
  (2) It also computes z' on input c' and w' by the same way as above.
  (3) Check whether $e(c_1, z') = e(c_1', z)$. If it holds, output 1. Otherwise, output 0.

We note that the authorization, verification and test algorithms are not affected by the third component of the ciphertext, except for the validity check of the ciphertext using the decryption algorithm. Because our rectification in Section 4 modifies only the third component of the ciphertext (and the decryption algorithm by reflecting it) from the original Huang *et al.*'s scheme, we do not need to modify the authorization, verification and test algorithms.

Remark 2.1. In the original scheme, the receiver's warrant rw, which is the output of the authorization algorithm $\mathrm{Aut}_r(sk, pk_t)$, was defined by

$$rw = (g^{\theta}, \alpha \cdot \mathcal{H}_1(g^{\beta_t \theta})).$$

It is very similar to a ciphertext of the ElGamal encryption scheme [8]. However, while α belongs to $\mathbb{Z}_q^*$, $\mathcal{H}_1(g^{\beta_t \theta})$ belongs to $\mathbb{G}$, and hence the multiplication between α and $\mathcal{H}_1(g^{\beta_t \theta})$ is not defined. To rectify it, we first publish an additional cryptographic hash function $\mathcal{H}_3$ in the setup algorithm and slightly modify authorization, verification and test algorithms for receiver's warrant as we have presented above. We note that this modification is irrelevant to our attack since it does not utilize any warrants.

2.2. Security of Huang *et al.*'s scheme

In PKE-AET, we consider the following two types of adversaries:

- Type I adversary: The adversary who has a warrant and the tester's secret key wants to reveal the message contained in the challenge ciphertext.
- Type II adversary: The adversary who does not have a warrant wants to distinguish whether the challenge ciphertext contains which message between two candidates.

In this article, we focus on the IND-CCA2 security against Type II adversaries only. We define the IND-CCA2 security with the following game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$:

(1) **Setup**: $\mathcal{C}$ generates n pairs of public key and secret key and sends all public keys to the adversary $\mathcal{A}$.
(2) **Phase 1**: $\mathcal{A}$ is allowed to query to the decryption oracle polynomially many times.
(3) **Challenge**: $\mathcal{A}$ first selects one user U_j as the challenge target. He then generates his own public key pk_t and secret key sk_t and selects two messages m_0 and m_1 of the same length. Thereafter, he forwards $(pk_j, pk_t, sk_t, m_0, m_1)$ to $\mathcal{C}$. $\mathcal{C}$ randomly selects $b \in \{0,1\}$, runs $\mathrm{Enc}(pk_j, m_b) \rightarrow c_b^*$ and sends c_b^* to $\mathcal{A}$.
(4) **Phase 2**: $\mathcal{A}$ queries to the oracles as in Step 2 polynomially many times. The constraint is that the challenger ciphertext c_b^* is not allowed to be queried to the decryption oracle.
(5) **Guess**: $\mathcal{A}$ outputs $b' \in \{0,1\}$.

We define the advantage of $\mathcal{A}$ in the above game by

$$\mathrm{Adv}_{\mathcal{A}}^{\text{IND-CCA2}} = \Pr[b' = b] - \frac{1}{2}.$$

Huang *et al.* claimed that their scheme is IND-CCA2 secure against Type II adversaries under the CDH assumption. We introduce the formal definition of the CDH assumption below. We note that we will also provide a rectified version of Huang *et al.*'s scheme under the CDH assumption.

DEFINITION 2.1. (CDH Assumption). *Let $\mathbb{G}$ be a cyclic group of order $p = p(\lambda)$ with a generator g for a security parameter λ. The CDH problem is defined as follows: Given (g, g^{μ}, g^{ν}) for randomly chosen $\mu, \nu \in \mathbb{Z}_p^*$, a PPT algorithm $\mathcal{A}$ finds the value $g^{\mu\nu}$ with the advantage*

$$\mathrm{Adv}^{\mathrm{CDH}}_{\mathbb{G},\mathcal{A}}(\lambda) := \Pr[\mathcal{A}(g, g^{\mu}, g^{\nu}) = g^{\mu\nu}].$$

We say that the CDH assumption on $\mathbb{G}$ holds if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in the security parameter λ.

3. OUR ADAPTIVE CHOSEN CIPHERTEXT ATTACK

Now, we present a CCA2 attack on Huang *et al.*'s PKE-AET against Type II adversaries and explain the flaw in their security proof.

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

(1) In the challenge phase of the IND-CCA2 security game, the adversary randomly selects x_0 and x_1 from $\mathbb{Z}_q^*$, computes $m_0 = g^{x_0}$ and $m_1 = g^{x_1}$ and sends them to the challenger.
(2) Once the challenge ciphertext

$$\begin{aligned}\hat{c} &= (\hat{c}_1, \hat{c}_2, \hat{c}_3)\\ &= (g^{\gamma}, m_b \cdot \mathcal{H}_1(g^{\alpha\gamma}), (m_b \| \gamma) \oplus \mathcal{H}_2(g^{\beta\gamma}))\end{aligned}$$

is returned for randomly chosen $\gamma \in \mathbb{Z}_q^*$ and $b \in \{0,1\}$ by the challenger, the adversary computes

$$A = \hat{c}_2 / (\hat{c}_1)^{x_0}. \tag{1}$$

Thereafter, he chooses $z \overset{\$}{\leftarrow} \mathbb{Z}_q^*$, sets $\bar{m} = g^z$, and computes

$$\bar{c}_2 = (\hat{c}_1)^z \cdot A = (g^{\gamma})^z \cdot A \tag{2}$$

and

$$\bar{c}_3 = \hat{c}_3 \oplus (m_0 \| 0^{\lceil \log_2 q \rceil}) \oplus (\bar{m} \| 0^{\lceil \log_2 q \rceil}) \tag{3}$$

where $0^{\lceil \log_2 q \rceil}$ is the $(\lceil \log_2 q \rceil)$-bit string of 0's. He queries $(\hat{c}_1, \bar{c}_2, \bar{c}_3)$ to the decryption oracle.
(3) If the decryption oracle responds $\bar{m}$, the adversary outputs 0. Otherwise, output 1.

For the confirmation, the adversary may repeat Step 2 by replacing x_0 and m_0 with x_1 and m_1 in the equations (1) and (3), respectively.

At Step 2, if the challenge ciphertext $\hat{c}$ is an encryption of m_0, i.e. $m_b = m_0$, then

$$\begin{aligned}A &= \hat{c}_2 / (\hat{c}_1)^{x_0} = m_b^{\gamma} \cdot \mathcal{H}_1(g^{\alpha\gamma}) / (g^{\gamma})^{x_0}\\ &= (m_b / m_0)^{\gamma} \cdot \mathcal{H}_1(g^{\alpha\gamma})\\ &= \mathcal{H}_1(g^{\alpha\gamma})\end{aligned}$$

since $m_0 = g^{x_0}$. Hence,

$$\bar{c}_2 = (g^{\gamma})^z \cdot A = (g^{\gamma})^z \cdot \mathcal{H}_1(g^{\alpha\gamma}) = \bar{m}^{\gamma} \cdot \mathcal{H}_1(g^{\alpha\gamma})$$

and

$$\begin{aligned}\bar{c}_3 &= (m_0 \| \gamma) \oplus \mathcal{H}_2(g^{\beta\gamma}) \oplus (m_0 \| 0^{\lceil \log_2 q \rceil}) \oplus (\bar{m} \| 0^{\lceil \log_2 q \rceil})\\ &= (\bar{m} \| \gamma) \oplus \mathcal{H}_2(g^{\beta\gamma}).\end{aligned}$$

Therefore,

$$(\hat{c}_1, \bar{c}_2, \bar{c}_3) = (g^{\gamma}, \bar{m}^{\gamma} \cdot \mathcal{H}_1(g^{\alpha\gamma}), (\bar{m} \| \gamma) \oplus \mathcal{H}_2(g^{\beta\gamma}))$$

is a valid ciphertext of the message $\bar{m}$, and the decryption oracle should respond $\bar{m}$. If the challenge ciphertext $\hat{c}$ is an encryption of m_1, A is not $\mathcal{H}_1(g^{\alpha\gamma})$ and hence $(\hat{c}_1, \bar{c}_2, \bar{c}_3)$ is not a ciphertext of $\bar{m}$. In this case, therefore, the decryption oracle should return other value $m' \neq \bar{m}$ in $\mathbb{G}$ or $\perp$. Furthermore, we note that the adversary may confirm whether the challenge ciphertext is an encryption of m_1 or not by repeating Step 2 with m_1 and x_1.

The Flaw in Their Security Proof. Now, we briefly explain the flaw in the security proof of their scheme. In Game 3 and Game 3-1 of the proof, the challenger who is simultaneously the solver of the CDH problem on the given instance (g, g^{μ}, g^{ν}), sets $g^{\alpha_j} = g^{\nu\delta_j}$ and $g^{\beta_j} = g^{\nu\delta_j'}$ in the setup phase, where δ_j and δ_j' are randomly chosen elements from $\mathbb{Z}_q^*$ by the challenger. Then, the challenger terminates the simulation once $\mathcal{H}_1(g^{\alpha_j\mu})$ or $\mathcal{H}_2(g^{\beta_j\mu})$ is requested to the hash oracles (denoted this event by E in their paper), because he can obtain the solution of the CDH problem on the instance (g, g^{μ}, g^{ν}) from $g^{\alpha_j\mu}$ or $g^{\beta_j\mu}$ by computing $(g^{\alpha_j\mu})^{\delta_j'^{-1}} = g^{\nu\delta_j\mu\delta_j^{-1}} = g^{\nu\mu}$ or $(g^{\beta_j\mu})^{\delta_j'^{-1}} = g^{\nu\delta_j'\mu\delta_j'^{-1}} = g^{\nu\mu}$. Furthermore, he can detect the event E by computing whether

$$e(g, h_1) = e(g^{\mu}, g^{\nu})^{\delta_j} \text{ and } e(g, h_2) = e(g^{\mu}, g^{\nu})^{\delta_j'} \tag{4}$$

for all stored $(h_1, h_1') \in \tau_1$ and $(h_2, h_2') \in \tau_2$, where τ_1 and τ_2 are the hash lists corresponded to the hash queries for hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively.

In our attack, however, the attacker can obtain a valid ciphertext from the challenge ciphertext without hash queries to the hash oracles even though a ciphertext is related to the instance of the CDH problem. Hence, although the decryption oracle should output the right answer for the correct

simulation, the challenger cannot detect it, and so the decryption oracle cannot provide the right answer to the attacker. Therefore, the simulation of the challenger is wrong.

To rectify it, we make the adversary query $g^{\alpha_j \nu}$ or $g^{\beta_j \nu}$ to the hash oracles in order to generate a valid ciphertext by modifying the challenge ciphertext. Considering the above, we attempt to modify the scheme in Section 4.

4. RECTIFICATION OF THE SCHEME

In this section, we provide a rectified version of Huang *et al.*'s scheme to achieve the IND-CCA2 security against Type II adversaries. Our modification is very simple: In the encryption algorithm, we additionally insert m^{γ} into the input of the hash function along with $g^{\beta\gamma}$ for the third component of a ciphertext. Then, we fix the decryption algorithm by reflecting the modification of the encryption algorithm.

The detail description of our modification is as follows:

- $\mathrm{Enc}'(pk, m)$: On input the public key $pk = (g^{\alpha}, g^{\beta})$ and a message $m \in \mathbb{G}$, it selects a random element $\gamma \xleftarrow{\$} \mathbb{Z}_q^*$ and outputs

  $$c = (g^{\gamma}, m^{\gamma} \cdot \mathcal{H}_1(g^{\alpha\gamma}), (m \| \gamma) \oplus \mathcal{H}_2(g^{\beta\gamma} \| m^{\gamma})).$$

- $\mathrm{Dec}'(sk, c)$: On input the secret key $sk = (\alpha, \beta)$ and a ciphertext $c = (c_1, c_2, c_3)$,
  (1) Compute $R' = c_2 / \mathcal{H}_1(c_1^{\alpha})$.
  (2) Compute $m' \| \gamma' = c_3 \oplus \mathcal{H}_2(c_1^{\beta} \| R')$.
  (3) Check whether $c_1 = g^{\gamma'}$ and $R' = m'^{\gamma'}$.
  (4) Output m' if both of them pass; otherwise, output $\perp$.

We note again that the above rectification modifies only the third component of the ciphertext (and the decryption algorithm by reflecting it) from the original Huang *et al.*'s scheme. Since the original authorization, verification and test algorithms are not affected by the third component of the ciphertext, except for the validity check of the ciphertext using the decryption algorithm, we can still utilize the same authorization, verification and test algorithms without changing from the original scheme.

Theorem 4.1. *Our modification is IND-CCA2 secure against Type II adversaries if the CDH assumption holds in the random oracle model.*

Proof. (Sketch) As we already pointed out several times, the flaw in the proof in [7] is only associated with the decryption oracle queries on ciphertexts obtained by modifying the challenge ciphertext. Therefore, the proof of this theorem is almost the same with those of Theorems 6 and 7 in [7], except for the simulation about decryption oracle queries on such ciphertexts. Let us sketch the differences between their proofs and ours by focusing on the simulation for such the decryption oracle queries and its analysis.

- Naturally, at Step (3) of Game 0 in the proof of Theorem 6 in [7], the challenge ciphertext should be changed to

  $$\hat{c} = (\hat{c}_1, \hat{c}_2, \hat{c}_3) = (g^{\gamma}, m_b^{\gamma} \mathcal{H}_1(g^{\alpha_j \gamma}), (m_b \| \gamma) \oplus \mathcal{H}_2(g^{\beta_j \gamma} \| m_b^{\gamma})).$$

- Now at Step (2) of Game 3-1 in the proof of Theorem 7 in [7], for the decryption queries on a ciphertext $c^* = (c_1^*, c_2^*, c_3^*)$,
  (1) If $c_1^* \neq \hat{c}_1$, then γ in c_1^* is different from that in $\hat{c}_1$. Hence, the inputs of $\mathcal{H}_1$ and $\mathcal{H}_2$ for c^* are different from those for $\hat{c}$, and so they should be ever queried to the hash oracles in order to generate a valid ciphertext c^*.
  (2) If $c_1^* = \hat{c}_1$ and $c_2^* \neq \hat{c}_2$, then the input of $\mathcal{H}_1$ for c_2^* is the same as that for $\hat{c}_2$. Hence, the message m in c_2^* should be different from that in $\hat{c}_2$ since $c_2^* \neq \hat{c}_2$. Therefore, the hash query on this input should be ever queried to the hash oracle for the hash function $\mathcal{H}_2$ in order to generate a valid ciphertext c^*.
  (3) If $c_1^* = \hat{c}_1$ and $c_2^* = \hat{c}_2$, then $c_3^* = \hat{c}_3$. Hence, such the ciphertext cannot be queried to the decryption oracle.

From the above, we confirm that $\mathcal{H}_2(g^{\beta_j \gamma} \| R)$ for some R should be ever requested to the hash oracle for generating a valid ciphertext from the challenge ciphertext. Hence, the challenger can terminate once $\mathcal{H}_2(g^{\beta_j \gamma} \| R)$ is requested.

Therefore, we can fix the flaw in the proof of the original scheme with our modification. We omit the detail of the proof. □

Efficiency Comparison. We can easily check that the size of the ciphertext obtained with our correct method is exactly the same as that with the original scheme. We make no other modifications except in the encryption and decryption algorithms; so the sizes of the warrants are also exactly the same as those in the original method.

The only difference in the encryption algorithm is the input length of the hash function $\mathcal{H}_2$, where the length in our method is two times longer than that in the original scheme. However, an exponentiation operation is generally much more expensive than computing a hash function and our encryption algorithm takes four exponentiations; hence, we might ignore this difference in length. The decryption algorithm is also modified slightly, but it requires four exponentiations, which has the same cost as the original decryption algorithm.

In summary, our modified version has almost the same performance as the original method in terms of efficiency and storage.

5. CONCLUSION

In this study, we provided a CCA2 attack on the recently proposed public key encryption with authorized equality test, which was claimed to be IND-CCA2 secure against Type II adversaries [7]. We also modified the scheme, so that it achieves the IND-CCA2 security under the CDH assumption in the random oracle model by adding a value related to the message as an input in one of the hash functions in the encryption algorithm. We note that our modified version has almost the same performance as the original method in terms of both efficiency and storage.

FUNDING

H. T. Lee, S. Ling, and H. Wang were supported by Research Grant TL-9014101684-01 and the Singapore Ministry of Education under Research Grant MOE2013- T2-1-041. J. H. Seo was supported by ETRI R&D program (15ZS1500).

REFERENCES

[1] Yang, G., Tan, C.H., Huang, Q. and Wong, D.S. (2010) Probabilistic public key encryption with equality test. In Pieprzyk, J. (ed.), Topics in Cryptology—CT-RSA 2010*, LNCS*, **5985**, pp. 119–131. Springer.

[2] Tang, Q. (2011) Towards public key encryption scheme supporting equality test with fine-grained authorization. In Parampalli, U. and Hawkes, P. (eds.), ACISP 2011*, LNCS*, **6812**, pp. 389–406. Springer.

[3] Tang, Q. (2012) Public key encryption schemes supporting equality test with authorisation of different granularity. *IJACT*, **2**, 304–321.

[4] Tang, Q. (2012) Public key encryption supporting plaintext equality test and user-specified authorization. *Secur. Commun. Netw.*, **5**, 1351–1362.

[5] Ma, S., Zhang, M., Huang, Q. and Yang, B. (2015) Public key encryption with delegated equality test in a multi-user setting. *Comput. J.*, **58**, 986–1002.

[6] Ma, S., Huang, Q., Zhang, M. and Yang, B. (2015) Efficient public key encryption with equality test supporting flexible authorization. *IEEE Trans. Inform. Forensics Secur.*, **10**, 458–470.

[7] Huang, K., Tso, R., Chen, Y. and Rahman, S. M. M., Almogren, A., and Alamri, A. (2015) PKE-AET: public key encryption with authorized equality test. *Comput. J.*, **58**, 2686–2697.

[8] ElGamal, T. (1985) A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, **31**, 469–472.