2017

# ID-based Encryption with Equality Test against Insider Attack

Tong Wu
*University of Wollongong*, tw225@uowmail.edu.au

Sha Ma
*University of Wollongong*, sma@uow.edu.au

Yi Mu
*University of Wollongong*, ymu@uow.edu.au

Shengke Zeng
*University of Wollongong*, shengke@uow.edu.au

## Publication Details

Wu, T., Ma, S., Mu, Y. & Zeng, S. (2017). ID-based Encryption with Equality Test against Insider Attack. Lecture Notes on Computer Science, 10342 168-183. Auckland, New Zealand 22nd Australasian Conference on Information Security and Privacy (ACISP 2017)

# ID-based Encryption with Equality Test against Insider Attack

**Abstract**
Testing if two ciphertexts contain the same plaintext is an interesting cryptographic primitive. It is usually referred to as equality test of encrypted data or equality test.

**Keywords**
insider, equality, against, encryption, attack, test, id-based

**Disciplines**
Engineering | Science and Technology Studies

**Publication Details**
Wu, T., Ma, S., Mu, Y. & Zeng, S. (2017). ID-based Encryption with Equality Test against Insider Attack. Lecture Notes on Computer Science, 10342 168-183. Auckland, New Zealand 22nd Australasian Conference on Information Security and Privacy (ACISP 2017)

# ID-based Encryption with Equality Test against Insider Attack

Tong Wu[1], Sha Ma[1,2], Yi Mu[1] and Shengke Zeng[1,3]

[1] Institute of Cybersecurity and Cryptology,
School of Computing and Information Technology, University of Wollongong
Wollongong, NSW, Australia
[2] College of Mathematics and Informatics, South China Agricultural University,
Guangzhou, Guangdong 510640, China
[3] School of Computer and Software Engineering, Xihua University,
Chengdu, 610039, China.
{tw225,sma,ymu,shengke}@uow.edu.au

**Abstract.** Testing if two ciphertexts contain the same plaintext is an interesting cryptographic primitive. It is usually referred to as equality test of encrypted data or equality test. One of attractive applications of equality test is for encrypted database systems, where the database server hosts the encrypted databases and users can query if the plaintext embedded in a ciphertext on a database is equal to that in the queried ciphertext without decryption. Although it is not hard to achieve with the pairing-based cryptography, the security against the insider attack (by the database server) is a challenging task. In this paper, we propose a novel equality test scheme aiming to solve the problem. Our scheme adopts the identity-based cryptography. We prove the security of our scheme in the random oracle model.

**Keywords:** ID-based encryption, equality test, insider attack

## 1 Introduction

The probabilistic public key encryption with equality test (PKEET) [14] is an interesting technique with wide applications such as in outsourced database systems, which host and manage encrypted data for clients. The merit of the equality test scheme is that one can check whether two ciphertexts contain the same plaintext without decrypting them.

In the original equality test scheme [14], outsourced database servers are usually considered to be semi-trusted because of its curiosity on user data. We call it an "Honest but Curious" (HBC) server. It is practical for the server to obtain the illegal profit from peddling users' private data by simple brute force attacks on the encrypted message. Our scheme should resist this kind of adversaries, even we assume that the adversary has access to all ciphertexts and can test their equality, which is called "insider attack" [11]. An HBC server (the insider), who runs the test algorithm correctly and continuously, can perform any polynomial time computation and then obtain the information beyond its own.

## 1.1 Related Work

Boneh et al. first proposed a public key encryption with keyword search scheme (PEKS) in the random oracle model [2]. When a user conducts a search, he can generate a trapdoor with a keyword and his private key. Taking the generated trapdoor and a ciphertext, the test algorithm will output "accept" if they contain the same keyword; otherwise, "reject". Their work provides a solution to the equality test on encrypted keywords in public key encryption.

To provide a general equality test scheme, Yang et al. proposed the first public key encryption with equality test (PKEET) [14]. In PKEET, given two tags $T_i$ and $T_j$ on ciphertexts $C_i$ and $C_j$ generated with $PK_i$ and $PK_j$ corresponding to message $M_i$ and $M_j$, respectively, there is a function $\mathsf{Test}(\cdot, \cdot)$, which outputs 1 iff $M_i = M_j$. Their work achieves the security against the One-way Chosen Ciphertext Attack (OW-CCA). There are some extensions of PKEET which offer the fine or flexible grain authorization and stronger security [13, 12, 6, 9, 10, 8]. To achieve the stronger security, the authorization mechanism is adopted to these PKEET schemes. Some of them utilize trapdoors generated from private keys which are used to the authorization process. As an instance, Ma et al. in [9], proposed a PKEET with the flexible authorization according to four scenarios. In [10], Ma et al. provided a solution to the PKEET in multi-user setting by delegating the equality test to a fully trusted proxy. Later, Ma [8] proposed an identity-based PKEET (IBEET). These works improved the security of PKEET to IND-CCA security, while private keys are kept secret. However, none of their works can resist the insider attack.

Mayer et al. [11] proposed a verifiable private equality test (VPET) for multi-party computation. The protocol resists attacks launched by HBC entities and active malicious entities who can behave active malicious actions. However, it requires that all users are online during testing and generate a proof for each equality test. It is therefore impractical for the cloud storage management and outsourced database services, which require users to be offline. Constructions presented in [7, 3] also provide solutions to insider attacks for the PEKS schemes, but not for the general equality test. Chen et al. in [3] proposed a PEKS scheme based on the dual-server framework. Peng et al. [7] proposed a PEKS scheme to prevent the trapdoor generated globally by containing identity set in trapdoors by keeping this set secret, so that the insider attack is eliminated.

## 1.2 Our Contribution

The probabilistic public key encryption with equality test was proposed by Yang et al. [14]. However, their scheme is vulnerable to the above insider attack. Since the ciphertext can be generated publicly, the HBC server can test the embedded message in the target ciphertext on its guess. To address this problem, we propose an efficient identity-based equality test scheme with resistance against the insider attack as our contributions. We define a novel security model for the confidentiality of IBEET which allows the adversary to conduct the equality test on all ciphertexts but can not generate ciphertexts. We refer it to as Weak-IND-ID-CCA (W-IND-ID-CCA). Nevertheless, it is stronger than security models for previous works under the same attack.

### 1.3 Organization

The rest of the paper is organized as follows. In Section 2, we provide some preliminaries for our construction. In Section 3, we formulate the notion of IBEET-IA. In Section 4, we present the construction of IBEET-IA and prove its security in Section 5. In Section 6, we construct a secure outsourced database application based on IBEET-IA and present the experimental results. In Section 7, we conclude our paper.

## 2 Preliminaries

### 2.1 Bilinear Pairing

**Definition 1 (Bilinear Group** [5]**).** *$\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ constitute a bilinear group if there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$.*

The bilinear pairing is an operation conducted on bilinear groups. Informally, two elements in such group are linearly related to the pairing result. The formal description of the bilinear paring is given as follows.

**Bilinear Pairing** Suppose that $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups with the same prime order $p$. Suppose that $g$ and $h$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. A bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ holds properties as follows:

1. Bilinearity: For any $x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$, $e(x^a, y^b) = e(x, y)^{ab}$.
2. Non-degeneration: $e(g, h) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the generator of $\mathbb{G}_T$.
3. Computability: There exists an efficient algorithm to compute $e(x, y)$, for any $x \in \mathbb{G}_1$ and $y \in \mathbb{G}_2$.

We say that a pairing is symmetric if $\mathbb{G}_1 = \mathbb{G}_2$. Our construction is built on such groups.

**Bilinear Diffie-Hellman Problem (BDHP).** Let $\mathbb{G}_1$, $\mathbb{G}_2$ be two groups of prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map and let $g$ be a generator of $\mathbb{G}_1$. The BDH problem in $(\mathbb{G}_1, \mathbb{G}_2, e)$ is as follows: Given $(g, g^a, g^b, g^c)$ for some $a, b, c \in \mathbb{Z}_p^*$ compute $e(g, g)^{abc} \in \mathbb{G}_2$. An polynomial algorithm $\mathcal{A}$ has advantage $\epsilon(\cdot)$ in solving BDH in $(\mathbb{G}_1, \mathbb{G}_2, e)$ if

$$\Pr[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}] \leq \epsilon(\lambda),$$

where the probability is over the random choice of $(a, b, c) \in \mathbb{Z}_p^*$, the random choice of $g \in \mathbb{G}_1$, and the random bits consumed by $\mathcal{A}$.

**Definition 2 (Bilinear Diffie-Hellman Assumption).** *We say that the BDH assumption holds if for any randomized polynomial time in $\lambda$ for some sufficiently large $\lambda$ algorithm $\mathcal{A}$ solves the BDH problem with the negligible advantage $\epsilon(\lambda)$.*

### 2.2 Short Signature without Random Oracle

Recall that the short signature scheme in plain model is proposed by Boneh and Boyen [1]. The security of their scheme depends on the $q$-Strong Diffie-Hellman assumption.

***q*-Strong Diffie-Hellman Problem (*q*-SDHP)** The $q$-SDHP in $(\mathbb{G}_1, \mathbb{G}_2, e)$ is as follows: Given $q+2$-tuple $(g_1, g_2, g_2^x, g_2^{x^2}, \cdots, g_2^{x^q})$ as input, output a pair $(c, g_1^{\frac{1}{x+c}})$, where $c \in \mathbb{Z}_p^*$. An polynomial algorithm $\mathcal{A}$ has advantage $\epsilon(\cdot)$ in solving $q$-SDHP in $(\mathbb{G}_1, \mathbb{G}_2, e)$ if

$$\Pr[\mathcal{A}(g_1, g_2, g_2^x, g_2^{x^2}, \cdots, g_2^{x^q}) = (c, g_1^{\frac{1}{x+c}})] \leq \epsilon(\lambda),$$

where the probability is over the random choice of $x$ in $\mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$.

**Definition 3 ($q$-Strong Diffie-Hellman Assumption).** *We say that the $q$-SDH assumption holds if for any randomized polynomial time in $\lambda$ for some sufficiently large $\lambda$ algorithm $\mathcal{A}$ solves the $q$-SDH problem with the negligible advantage $\epsilon(\lambda)$.*

**Construction.** We adopt the weak construction of their short signature. It's sufficient to guarantee the secure of our scheme from the attack by forging an invalid pair of elements and testing with other valid ciphertexts. We recall their construction which consists of three algorithm (KeyGen, Sign, Vefiry).

- KeyGen. Pick random $x \in \mathbb{Z}_p$, and compute $v = g_2^x \in \mathbb{G}_2$. The public key is $(g_1, g_2, v)$. The secret key is $x$.
- Sign. Given a secret key $x$ and a message $m \in \mathbb{Z}_p$, output the signature $\sigma \leftarrow g_1^{1/(x+m)} \in \mathbb{G}_1$. Here $1/(x+m)$ is computed modulo $p$. By convention in this context we define $1/0$ to be $0$ so that in the unlikely event that $x+m=0$ we have $\leftarrow 1$.
- Verify. Given a public key $(g_1, g_2, v)$, a message $m \in \mathbb{Z}_p$, and a signature $\sigma \in \mathbb{G}_1$, verify that
$$e(\sigma, v \cdot g_2^m) = e(g_1, g_2)$$
If equality holds output valid. If $\sigma = 1$ and $vg_2^m = 1$ output valid. Otherwise, output invalid.

## 3 Definitions

In this section, we give formal definitions of our scheme and security model. Our scheme achieves chosen ciphertext security (i.e. W-IND-ID-CCA) under the defined security model.

## 3.1 ID-based Encryption with Equality Test against Insider Attack

We propose an ID-based encryption with equality test. The scheme $\Omega$ consists of a set of algorithms: $\Omega = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Join}, \mathsf{Enc}, \mathsf{Test}, \mathsf{Dec})$.

- $\mathsf{Setup}(1^\lambda)$: It takes the secure parameter $\lambda$ and outputs the system public parameters $\mathsf{pp}$, the master secret key $msk$.
- $\mathsf{Extract}(\mathsf{ID}, msk)$: It takes $(\mathsf{ID}, msk)$ and $\mathsf{pp}$ and outputs the private key $d_{\mathsf{ID}}$.
- $\mathsf{Join}(\{\mathsf{ID}\})$: It takes a set of identities $\{\mathsf{ID}\}$ and outputs a group token $\mathsf{tok}$ via a secure protocol.
- $\mathsf{Enc}(m, \mathsf{ID}, P_{pub}, \mathsf{tok})$: It takes $(m, \mathsf{ID}, \mathsf{tok}, P_{pub})$ and outputs the ciphertext $C = (c_1, c_2, c_3, c_4)$.
- $\mathsf{Test}(C_A, C_B)$: It takes ciphertexts $C_A$ and $C_B$ produced by user A and user B, respectively. It outputs 1 if messages associated with $C_A$ and $C_B$ are equal. Otherwise, it outputs 0.
- $\mathsf{Dec}(C, d_{\mathsf{ID}}, \mathsf{tok})$: It takes the ciphertext $C$, $d_{\mathsf{ID}}$ and $\mathsf{tok}$ and outputs the message $m$, if $C$ is a valid ciphertext under $\mathsf{ID}$. Otherwise, it outputs $\perp$ .

Note: $\mathsf{pp}$ refers to public parameters and hash functions used in our scheme.

## 3.2 Security Models

**Definition 4 (Weak-IND-ID-CCA (W-IND-ID-CCA)).** *Let* $\Omega = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Join}, \mathsf{Enc}, \mathsf{Test}, \mathsf{Dec})$ *be the scheme and* $\mathcal{A}$ *be a polynomial time adversary.*

- **Setup:** The challenger runs the $\mathsf{Setup}$ algorithm to initialize the system and obtains $P_{pub}$, $msk$ and the challenged group token $\mathsf{tok}$. It gives $P_{pub}$ to the adversary $\mathcal{A}$.
- **Phase 1:** The adversary issues queries $q_1$, $q_2$, $\cdots$, $q_m$ where $q_i$ is one of:
  - $H_1$ Query $(\mathsf{ID}_i)$. The challenger responds by running $H_1(\cdot)$ to generate $g_{\mathsf{ID}_i}$. It sends $g_{\mathsf{ID}_i}$ to the adversary.
  - Extract Query $(\mathsf{ID}_i)$. The challenger responds by running $\mathsf{Extract}$ algorithm to generate the private key $d_{\mathsf{ID}_i}$ corresponding to the public key $\mathsf{ID}_i$. It sends $d_{\mathsf{ID}_i}$ to the adversary.
  - $H_2$ Query $(\mathbb{G}_1^3 \times \mathbb{G}_2)$. The challenger responds by running $H_2(\cdot)$ to generate the corresponding hash value. It sends the hash value to the adversary.
  - Encryption Query $(m_i, \mathsf{ID}_i)$. The challenger responds by running $\mathsf{Enc}$ to generate the ciphertext $C_i$ corresponding to $(m_i, \mathsf{ID}_i)$. It sends the ciphertext $C_i$ to the adversary.
  - Decryption Query $(C_i, \mathsf{ID}_i)$. The challenger responds by running $\mathsf{Extract}$ algorithm to generate $d_{\mathsf{ID}_i}$ corresponding to $\mathsf{ID}_i$. It then runs $\mathsf{Dec}$ to decrypt the ciphertext $C_i$ using $d_{\mathsf{ID}_i}$. It sends the resulting plaintext to the adversary.

- **Challenge:**
  Once $\mathcal{A}$ decides the Phase 1 is over, it sends two equal-length messages $m_0, m_1$ and $\mathsf{ID}^*$ to be challenged to the challenger, where both $m_0, m_1$ are not issued in the Encryption Query and $\mathsf{ID}^*$ is not issued in the Extract Query in the Phase 1. The challenger randomly picks $b \in \{0,1\}$, and responds with $C^* \leftarrow \mathsf{Enc}(m_b, \mathsf{ID}^*, P_{pub}, \mathsf{tok})$.
- **Phase 2:**The adversary issues queries $q_{m+1}$, $q_{m+2}$, $\cdots$, $q_n$ where $q_i$ is one of:
  - $H_1$ Query $(\mathsf{ID}_i)$. The challenger responds as in Phase 1.
  - Extract Query $(\mathsf{ID}_i)$ where $\mathsf{ID}_i \neq \mathsf{ID}^*$. The challenger responds as in Phase 1.
  - $H_2$ Query $(\mathbb{G}_1^3 \times \mathbb{G}_2)$. The challenger responds as in Phase 1.
  - Encryption Query $(m_i, \mathsf{ID}_i)$ where $m_i \notin \{m_0, m_1\}$. The challenger responds as in Phase 1.
  - Decryption Query $(C_i, \mathsf{ID}_i)$ where $(C_i, \mathsf{ID}_i) \neq (C^*, \mathsf{ID}^*)$. The challenger responds as in Phase 1.
- **Output:** Finally, $\mathcal{A}$ gives a guess $b'$ on $b$. If $b' = b$, we say $\mathcal{A}$ wins the game.

We define $\mathcal{A}$'s advantage on breaking the scheme as

$$\mathsf{Adv}^{\text{W-IND-ID-CCA}}_{\Omega,\mathcal{A}(H_1,H_2,\mathsf{Extract},\mathsf{Enc},\mathsf{Dec})} = \left|\Pr[b' = b] - \frac{1}{2}\right| = \epsilon(\lambda),$$

where $\epsilon(\lambda)$ is a polynomial of $\lambda$. $\Omega$ is W-IND-ID-CCA secure if $\epsilon(\lambda)$ is a negligible function. In the W-IND-ID-CCA model, the adversary has access to ciphertexts without any valid $\mathsf{tok}$.

## 4 The Proposed Scheme

### 4.1 ID-based Encryption with Equality Test against Insider Attack

Our scheme aims to provide the service for designated senders. That is, the receiver and its designated senders form a group of users. $\mathsf{tok}$ denotes a secret information shared among group members. The server and other users can only conduct the equality test. Our protocol consists of the following five algorithms:

- $\mathsf{Setup}(1^\lambda)$: Initially, the system takes a security parameter $\lambda$ and returns public system parameters $\mathsf{pp}$, the master secret key $msk$.
  1. The system generates two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with the same prime order $p$ of $\lambda$-length bits and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. The system selects an arbitrary generator $g \in \mathbb{G}_1$.
  2. The system selects $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ as $msk$ and sets $P_{pub} = g^\alpha$.
  3. The system chooses three hash functions:

     $$H : \{0,1\}^t \to \mathbb{Z}_p^*, \quad H_1 : \{0,1\}^* \to \mathbb{G}_1, \quad H_2 : \mathbb{G}_1^3 \times \mathbb{G}_2 \to \{0,1\}^{t+l},$$

     where $l$ is the length of random numbers and $t$ is the length of messages. It publishes $\mathsf{pp} = \{\lambda, p, t, l, g, P_{pub}, e, H, H_1, H_2\}$.

– $\mathsf{Extract}(\mathsf{ID}, msk)$: PKG generates $d_{\mathsf{ID}}$ for each user's $\mathsf{ID}$.

$$g_{\mathsf{ID}} = H_1(\mathsf{ID}), \quad d_{\mathsf{ID}} = g_{\mathsf{ID}}^{\alpha},$$

where $d_{\mathsf{ID}}$ is distributed via a secure channel.

– $\mathsf{Join}(\{\mathsf{ID}\})$. Users organize a small group and share a common secret token $\mathsf{tok} = \beta$ via a secure protocol.

– $\mathsf{Enc}(m, \mathsf{ID}, P_{pub}, \mathsf{tok})$: To encrypt $m$, it selects two random numbers $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$, with $|r_1| = l$. Then it computes

$$c_1 = g_{\mathsf{ID}}^{\frac{r_1}{H(m)+\mathsf{tok}}}, \quad c_2 = g_{\mathsf{ID}}^{r_1}, \quad c_3 = g^{r_2},$$

$$c_4 = (m \parallel r_1) \oplus H_2(c_1 \parallel c_2 \parallel c_3 \parallel e(P_{pub}, g_{\mathsf{ID}})^{r_2}).$$

Finally, it returns $C = (c_1, c_2, c_3, c_4)$.

– $\mathsf{Test}(C_A, C_B)$: Suppose that

$$C_A \leftarrow \mathsf{Enc}(m_A, \mathsf{ID}_A, P_{pub}, g_{\mathsf{ID}_A})$$

and

$$C_B \leftarrow \mathsf{Enc}(m_B, \mathsf{ID}_B, P_{pub}, g_{\mathsf{ID}_B})$$

are generated by user A and user B, respectively. With $C_A = (c_{A,1}, c_{A,2}, c_{A,3}, c_{A,4})$ and $C_B = (c_{B,1}, c_{B,2}, c_{B,3}, c_{B,4})$, the test algorithm on $C_A$ and $C_B$ runs as follows:

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

If the equation holds, it explains the equality between $m_A$ and $m_B$, then outputs 1. Otherwise, outputs 0.

– $\mathsf{Dec}(C, d_{\mathsf{ID}}, \mathsf{tok})$: To decrypt the ciphertext $C$ with $d_{\mathsf{ID}}$ and $\mathsf{tok}$, it computes:

$$m \parallel r_1 = c_4 \oplus H_2(c_1 \parallel c_2 \parallel c_3 \parallel e(c_3, d_{\mathsf{ID}})).$$

If

$$c_1 = g_{\mathsf{ID}}^{\frac{r_1}{H(m)+\mathsf{tok}}} \wedge c_2 = g_{\mathsf{ID}}^{r_1},$$

it returns $m$. Otherwise, $\perp$.

## 4.2 Correctness

We say that $\Omega$ has the ciphertext comparability with error $\mu$, for some function $\mu(\cdot)$. For instance, we run the equality test on $C_A \leftarrow \mathsf{Enc}(m_A, \mathsf{ID}_A, P_{pub}, \mathsf{tok})$, $C_B \leftarrow \mathsf{Enc}(m_B, \mathsf{ID}_B, P_{pub}, \mathsf{tok})$ generated by user A and user B, respectively.

$$C_A = (c_{A,1}, c_{A,2}, c_{A,3}, c_{A,4}), \quad C_B = (c_{B,1}, c_{B,2}, c_{B,3}, c_{B,4})$$

We compute the left hand side ($L$) and the right hand side ($R$) of Eq. (1) in the $\mathsf{Test}$ algorithm, respectively. We analyze it in two cases: $m_A = m_B$ and $m_A \neq m_B$.

$$\begin{aligned} L &\coloneqq e(c_{A,1}, c_{B,2}) & R &\coloneqq e(c_{B,1}, c_{A,2}) \\ &= e(g_{\mathsf{ID}_A}^{r_{A,1}/(\mathsf{tok}+H(m_A))}, g_{\mathsf{ID}_B}^{r_{B,1}}) & &= e(g_{\mathsf{ID}_B}^{r_{B,1}/(\mathsf{tok}+H(m_B))}, g_{\mathsf{ID}_A}^{r_{A,1}}) \\ &= e(g_{\mathsf{ID}_A}^{r_{A,1}/(\mathsf{tok}+H(m_A))}, g_{\mathsf{ID}_B}^{r_{B,1}}) & &= e(g_{\mathsf{ID}_B}^{r_{B,1}/(\mathsf{tok}+H(m_B))}, g_{\mathsf{ID}_A}^{r_{A,1}}) \\ &= e(g_{\mathsf{ID}_A}, g_{\mathsf{ID}_B})^{r_{A,1}r_{B,1}/(\mathsf{tok}+H(m_A))} & &= e(g_{\mathsf{ID}_B}, g_{\mathsf{ID}_A})^{r_{A,1}r_{B,1}/(\mathsf{tok}+H(m_B))} \end{aligned}$$

Case 1: If $m_A = m_B$, the equation holds with the probability of 1;
Case 2: If $m_A \neq m_B$, the equation holds when the collision occurs in hash function $H(m)$, that is $H(m_A) = H(m_B)$ while $m_A \neq m_B$. We define $H(m)$ is a collision resistant hash function. $\Pr[H(m_A) = H(m_B)|m_A \neq m_B]$ is a negligible function.

## 5 Security

Our scheme is a chosen ciphertext secure IBEET (i.e. W-IND-ID-CCA), assuming BDH is hard in groups generated by a BDH parameter generator.

**Theorem 1.** *Let $\mathcal{A}$ be a* W-IND-ID-CCA *adversary on IBEET-IA that making at most $q_e$ times extract queries and $q_d$ times decryption queries achieves advantage at least $\epsilon$. Then there is a BDH algorithm $\mathcal{B}$ solving the BDH problem with the advantage at least $\frac{\epsilon}{e(q_e+q_d+1)}$.*

*Proof.* Suppose there is a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ who achieves the advantage $\epsilon$ on breaking $\Omega = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Test}, \mathsf{Dec})$. Given a BDH instance, a PPT adversary $\mathcal{B}$ will take advantage of $\mathcal{A}$ to solve the BDH problem with the probability of $\epsilon'$. Hence, if the BDH assumption holds, then $\epsilon'$ is negligible and consequently $\epsilon$ must be negligible.

Assume $\mathcal{B}$ holds a BDH tuple $(g, U, V, R)$, where $x = \log_g U$, $y = \log_g V$ and $z = \log_g R$ are unkown. Let $g$ be the generator of $\mathbb{G}_1$. Finally, $\mathcal{B}$ is supposed to output $e(g, g)^{xyz} \in \mathbb{G}_2$. The game between $\mathcal{B}$ and $\mathcal{A}$ runs as follows:

**Setup**: $\mathcal{B}$ sets $P_{pub} = g^{x \cdot r} = U^r$, where $r \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $\mathsf{tok} = \beta \xleftarrow{\$} \mathbb{Z}_p^*$. $\mathcal{B}$ gives $P_{pub}$ to $\mathcal{A}$.

**Phase 1**:

- $H_1$ Query. $\mathcal{A}$ can query the random oracle $H_1$ at any time. $\mathcal{A}$ queries $\mathsf{ID}_i$ to obtain $g_{\mathsf{ID}_i}$. $\mathcal{B}$ responds with $g_{\mathsf{ID}_i}$ if $\mathsf{ID}_i$ has been in the $H_1$ table, $(\mathsf{ID}_i, g_{\mathsf{ID}_i}, a_i, \mathtt{coin}_i)$. Otherwise, for each $\mathsf{ID}_i$, $\mathcal{B}$ responds as follows:
  - $\mathcal{B}$ tosses a coin with $\Pr[\mathtt{coin}_i = 0] = \delta$. If $\mathtt{coin}_i = 1$, responds to $\mathcal{A}$ with $g_{\mathsf{ID}_i} = g^{a_i}$, $a_i \xleftarrow{\$} \mathbb{Z}_p^*$. Otherwise, $\mathcal{B}$ sets $g_{\mathsf{ID}_i} = g^{a_i y} = V^{a_i}$.
  - $\mathcal{B}$ responds with $g_{\mathsf{ID}_i}$, then adds $(\mathsf{ID}_i, g_{\mathsf{ID}_i}, a_i, \mathtt{coin}_i)$ in the $H_1$ table, which is initially empty.

- Extract Query. $\mathcal{A}$ queries the private key of $\mathsf{ID}_i$. $\mathcal{B}$ responds as follows:
  - $\mathcal{B}$ obtains $H_1(\mathsf{ID}_i) = g_{\mathsf{ID}_i}$ in the $H_1$ table. If $\text{coin}_i = 0$, $\mathcal{B}$ responds with $\perp$ and terminates the game.
  - Otherwise, $\mathcal{B}$ responds with $d_{\mathsf{ID}_i} = P_{pub}^{a_i} = U^{r \cdot a_i}$, where $a_i, g_{\mathsf{ID}_i}$ is in the $H_1$ table.
  - $\mathcal{B}$ sends $d_{\mathsf{ID}_i}$ to $\mathcal{A}$, then stores $(d_{\mathsf{ID}_i}, a_i, \mathsf{ID}_i)$ in the private key list, which is initially empty.
- $H_2$ Query. $\mathcal{A}$ queries $D_i \in G_1^3 \times G_2$. $\mathcal{B}$ responds with $W_i \in H_2(D_i)$ in the $H_2$ table. Otherwise, for every $D_i$, $\mathcal{B}$ selects a random string $W_i = \{0,1\}^{t+l}$ as the $H_2(D_i)$. $\mathcal{B}$ responds $\mathcal{A}$ with $H_2(D_i)$ and adds $(D_i, W_i)$ in the $H_2$ table, which is initially empty.
- Encryption Query. $\mathcal{A}$ queries $m_i$ encrypted with $\mathsf{ID}_i$. $\mathcal{B}$ responds as follows:
  - $\mathcal{B}$ searches the $H_1$ table to obtain the $g_{\mathsf{ID}_i}$.
  - Then $\mathcal{B}$ selects $r_{i,1}, r_{i,2} \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and computes:

$$c_{i,1} = g_{\mathsf{ID}_i}^{r_{i,1}/(\mathsf{tok}+H(m_i))}, \quad c_{i,2} = g_{\mathsf{ID}_i}^{r_{i,1}}, \quad c_{i,3} = g^{r_{i,2}},$$

$$D_i = c_{i,1} \parallel c_{i,2} \parallel c_{i,3} \parallel e(P_{pub}, g_{\mathsf{ID}_i})^{r_{i,2}}.$$

  - $\mathcal{B}$ queries $\mathcal{O}_{H_2}$ to obtain $W_i = H_2(D_i)$.
  - $\mathcal{B}$ computes $c_{i,4} = (m_i \parallel r_{i,1}) \oplus W_i$.

  $\mathcal{B}$ responds with $C_i = (c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4})$.
- Decryption Query. $\mathcal{A}$ queries $C_i$ to be decrypted in $\mathsf{ID}_i$. $\mathcal{B}$ responds as follows:
  - $\mathcal{B}$ searches the $H_1$ table to obtain the $g_{\mathsf{ID}_i}$. If $\texttt{coin}_i = 1$, obtain $d_{\mathsf{ID}_i}$ of $\mathsf{ID}_i$ in the private key list to decrypt $C_i$. Then $\mathcal{B}$ computes the bilinear map with $d_{\mathsf{ID}_i}$:

$$e(c_{i,3}, d_{\mathsf{ID}_i}) = e(g^{r_{i,2}}, g^{a_i x r}) = e(g, U)^{r_{i,2} a_i r}.$$

  - After that, $\mathcal{B}$ computes $D_i = c_{i,1} \parallel c_{i,2} \parallel c_{i,3} \parallel e(P_{pub}, g_{\mathsf{ID}_i})^{r_{i,2}}$ and obtains $W_i$ in the $H_2$ table. $\mathcal{B}$ obtains $m_i$ and $r_{i,1}$ by $c_{i,4} \oplus W_i$.
  - Eventually, $\mathcal{B}$ computes $c'_{i,1}, c'_{i,2}$ with $m_i$ and $r_{i,1}$ decrypted from $C_i$. If it is a valid ciphertext that $c'_{i,1} = c_{i,1}$ and $c'_{i,2} = c_{i,2}$, $\mathcal{B}$ responds with $m_i$. Otherwise, $\perp$.

**Challenge**: Once $\mathcal{A}$ decides the Phase 1 is over, $\mathcal{A}$ outputs two equal-length messages $m_0$, $m_1$ and $\mathsf{ID}^*$ to be challenged, where both $m_0$, $m_1$ are not issued in Encryption Query and $\mathsf{ID}^*$ is not queried in Extract Query in Phase 1. $\mathcal{B}$ responds as follows:

- $\mathcal{B}$ encrypts $m_0$ and $m_1$ and gets $C_0$ and $C_1$.
- If $\mathcal{B}$ searches the $H_1$ table. If $\texttt{coin}^* = 1$, then $\mathcal{B}$ responds with $\perp$ and terminates the game, since $g_{\mathsf{ID}^*} = g^{a^*}$.
- Otherwise, $\mathcal{B}$ randomly selects $b \in \{0,1\}$. Since $g_{\mathsf{ID}^*} = g^{ya^*} = V^{a^*}$, $\mathcal{B}$ can calculate

$$c_1^* = g_{\mathsf{ID}^*}^{r_1^*/(\mathsf{tok}+H(m_b))}, \quad c_2^* = g_{\mathsf{ID}^*}^{r_1^*}, \quad c_3^* = R = g^z, \quad c_4^* = (m_b || r_1^*) \oplus W^*,$$

  where $W^* = H_2(D^*)$ and $D^* = c_1^* \parallel c_2^* \parallel c_3^* \parallel e(P_{pub}, g_{\mathsf{ID}^*})^z$ (that $e(P_{pub}, g_{\mathsf{ID}^*})^z$ is unknown which $\mathcal{B}$ wants $\mathcal{A}$ to compute). $C^* = (c_1^*, c_2^*, c_3^*, c_4^*)$ is a valid ciphertext for $m_b$.

– $\mathcal{B}$ responds $\mathcal{A}$ with $C^*$.

**Phase 2**:

– $H_1$ Query. $\mathcal{A}$ queries as in Phase 1.
– Extract Query. $\mathcal{A}$ queries as in Phase 1, except that $\mathsf{ID}_i \neq \mathsf{ID}^*$.
– $H_2$ Query. $\mathcal{A}$ issues the query as in Phase 1.
– Encryption Query. $\mathcal{A}$ queries as in Phase 1, except that the message $m_i \notin \{m_0, m_1\}$.
– Decryption Query. $\mathcal{A}$ queries as in Phase 1, except that the ciphertext $(C_i, \mathsf{ID}_i) \neq (C^*, \mathsf{ID}^*)$.

**Output**: $\mathcal{A}$ gives a guess $b'$ on $b$. If $b' \neq b$, $\mathcal{B}$ responds with failure and terminates the game. If $b' = b$, then $\mathcal{B}$ gets the result of the BDH tuple by guessing the inputs of $H_2$ Query. Suppose $D_{out} = D^*$, $\mathcal{B}$ obtains $e(P_{pub}, g_{\mathsf{ID}^*})^z$ directly from $D_{out}$ by removing first $3k$ bits (the elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ is $k$ bits length.) that is $c_1^* \parallel c_2^* \parallel c_3^*$. Then $\mathcal{B}$ obtains $e(g,g)^{xyz} = e(P_{pub}, g_{\mathsf{ID}^*})^{z(a^*r)^{-1}} = (e(U,V)^{za^*r})^{(a^*r)^{-1}}$.

*Claim.* If the algorithm $\mathcal{B}$ does not abort during the simulation then the algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Furthermore, if $\mathcal{B}$ does not abort then $\left|\Pr[b' = b] - \frac{1}{2}\right| \geq \frac{\epsilon}{e(q_e+q_d+1)}$. The probability over the random bits used by $\mathcal{A}$, $\mathcal{B}$ and the challenger.

It remains to bound the probability that $\mathcal{B}$ aborts during the simulation. The algorithm $\mathcal{B}$ could abort for three reasons: (1) a bad private key extraction query from $\mathcal{A}$ during the phase 1 or 2, (2) $\mathcal{A}$ chooses a bad ID to be challenged on, or (3) a bad decryption query from $\mathcal{A}$ during the phase 1 or 2. We define three corresponding events:

$\varepsilon_1$: $\mathcal{B}$ aborts at the Extract Query step.
$\varepsilon_2$: $\mathcal{B}$ aborts at the Decryption Query step.
$\varepsilon_3$: $\mathcal{B}$ aborts at the Challenge step.

We have

$$\Pr[\neg\varepsilon_1 \wedge \neg\varepsilon_2 \wedge \neg\varepsilon_3] \geq (1-\delta)^{q_e+q_d}\delta.$$

We provide the proof on $\Pr[\neg\varepsilon_1 \wedge \neg\varepsilon_2 \wedge \neg\varepsilon_3]$ by induction on the maximum number of queries $q_e + q_d$ made by the adversary. Let $\varepsilon^{0\cdots i}$ be the event that $\varepsilon_1 \vee \varepsilon_2 \vee \varepsilon_3$ happens after $\mathcal{A}$ queries at most $i$ times and let $i = q_e + q_d$. Similarly, let $\varepsilon^i$ be the event that $\varepsilon_1 \vee \varepsilon_2 \vee \varepsilon_3$ happens for the first time when $\mathcal{A}$ queries the $i_{th}$ item. For $i = 0$, it is trivial that $\Pr[\neg\varepsilon^{0\cdots 0}] = \delta$. Suppose that for $i-1$ the $\Pr[\neg\varepsilon^{0\cdots i-1}] = (1-\delta)^{i-1}\delta$ holds. Then for $i$, it holds

$$\begin{aligned}\Pr[\neg\varepsilon^{0\cdots i}] &= \Pr[\neg\varepsilon^{0\cdots i} | \neg\varepsilon^{0\cdots i-1}]\Pr[\neg\varepsilon^{0\cdots i-1}] \\ &= \Pr[\neg\varepsilon^{i} | \neg\varepsilon^{0\cdots i-1}]\Pr[\neg\varepsilon^{0\cdots i-1}] \\ &\geq \Pr[\neg\varepsilon^{i} | \neg\varepsilon^{0\cdots i-1}](1-\delta)^{i-1}\delta.\end{aligned}$$

Hence, we bound the probability of $\varepsilon^i$ not to happen with $\mathcal{A}$'s $i_{th}$ query. The query is either an Extract Query for $\mathsf{ID}_i$ or a Decryption Query for $(C_i, \mathsf{ID}_i)$. Recall that if $coin_i = 1$ it cannot cause $\varepsilon_1$ and $\varepsilon_2$ to happen. We consider three cases:

Case 1 The $i_{th}$ query is the first time $\mathcal{A}$ queries $\mathsf{ID}_i$. In this case, $\Pr[coin_i = 1] = 1 - \delta$ and hence

$$\Pr[\neg\varepsilon^i | \neg\varepsilon^{0\cdots i-1}] \geq 1 - \delta.$$

Case 2 $\mathsf{ID}_i$ was queried in previous Extract Query. Assuming the previous query did not cause $\varepsilon^{0\cdots i-1}$ to happen we have $coin_i = 1$. Hence,

$$\Pr[\neg\varepsilon^i | \neg\varepsilon^{0\cdots i-1}] = 1.$$

Case 3 $\mathsf{ID}_i$ was queried in the previous Decryption Query. Similarly to Case 2, we have $coin_i = 1$, Hence,

$$\Pr[\neg\varepsilon^i | \neg\varepsilon^{0\cdots i-1}] = 1.$$

To summarize, we have

$$\Pr[\neg\varepsilon^i | \neg\varepsilon^{0\cdots i-1}] \geq 1 - \delta$$

whatever the $i_{th}$ query is. Therefore,

$$\Pr[\neg\varepsilon^i] \geq (1-\delta)^i \delta$$

is as required. Since

$$\Pr[\neg\varepsilon_1 \wedge \neg\varepsilon_2 \wedge \neg\varepsilon_3] \geq (1-\delta)^{q_e+q_d}\delta,$$

the success probability is maximum at $\delta_{opt}$. Using $\delta_{opt} = \frac{1}{q_e+q_d+1}$, the probability that $\mathcal{B}$ does not abort is at least $\frac{1}{e(q_e+q_d+1)}$. This shows that $\mathcal{B}$'s advantage is at least $\frac{\epsilon}{e(q_e+q_d+1)}$ as required.

*Remark 1.* Suppose the $q$-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$. Then the signature scheme above is secure against existential forgery under a weak chosen message attack. The proof is referred to [1]. The security of the simple construction in [1] implies the security of our construction. If the construction is secure against existential forgery under a weak chosen message attack, our construction is secure under the W-IND-ID-CCA.

*Remark 2.* Responses to $H_1$ queries are as in the real attack since responses are uniformly and independently distributed in $\mathbb{G}_1^*$. All responses to the private key extraction queries and decryption queries are valid. Finally, the challenge ciphertext $C^*$ given to $\mathcal{A}$ is the encryption of $m_b$ for some random $b \in \{0,1\}$. Therefore, by the definition of $\mathcal{A}$ we have that

$$\left|\Pr[b' = b] - \frac{1}{2}\right| \geq \frac{\epsilon}{e(q_e+q_d+1)}.$$

## 6 Experiments on A Database

### 6.1 Setup of Experiments

The purpose of our experiments is to demonstrate the efficiency and feasibility of our scheme. The instantiation of our novel IBEET-IA scheme is implemented in Java with the Java Pairing Based Cryptography library (JPBC) [4]. In our experiments, type A pairing is invoked for the configuration of our IBEET-IA program, which is symmetrically built from type A super-singular elliptic curve with the embedding degree of two. Precisely, the length of elements is 512 bits for $\mathbb{G}_1$ and 1024 bits for $\mathbb{G}_2$. For the database, MySQL on a virtual machine with 1024MB memory and one processing core is used as the experimental environment. Detailed parameters of our experiments are shown in Table 1. In addition,

**Table 1.** Experimental environment

| **Hardware** | Parameter |
|---|---|
| **Computer** | Mac Pro 13' 2015 |
| **Processor** | 2.7 GHz Intel Core i5 |
| **Operation System** | OS X El Capitan 10.11.6 |
| **Memory** | 8 GB 1867 MHz DDR3 |
| **Cores** | 2 |
| **Software** | Parameter |
| **Virtual Machine** | VMware Fusion 7 |
| **VM OS** | Ubuntu 14.04.3 |
| **VM Memory** | 1024 MB |
| **VM Core** | 1 |
| **Database** | MySQL |
| **Development** | Java 1.8 + Eclipse + JPBC |

experiments were conducted on the practical database with eight tables, which are constructed with columns over rows on $4 \times 273$, $8 \times 23$, $6 \times 326$, $5 \times 2996$, $9 \times 7$, $4 \times 273$, $8 \times 110$ and $3 \times 7$, respectively. After running three experiments on three algorithms, $\mathsf{Enc}$, $\mathsf{Dec}$ and $\mathsf{Test}$, the efficiency of those algorithms are analyzed. The results are shown in the following subsections.

### 6.2 Performance Evaluation

In the following three experiments, the total time cost is linear with the size of entities to be encrypted, decrypted or tested. In $\mathsf{Enc}$ and $\mathsf{Dec}$ algorithms, they conduct a pairing and three exponents, simultaneously. Therefore, the average time consumption is the same, 0.3 second (or 0.3s) to 0.2s for each entity in the encryption and the decryption containing the reading and writing time. The average time consumption is 0.1s for each equality test, which is a reasonable result and can be accepted by practice applications.

**Encryption Performance** The encryption encrypted with the user's ID. For the $i_{th}$ encryption on $m_i$, it inputs `ID` and the value of the current cell and computes

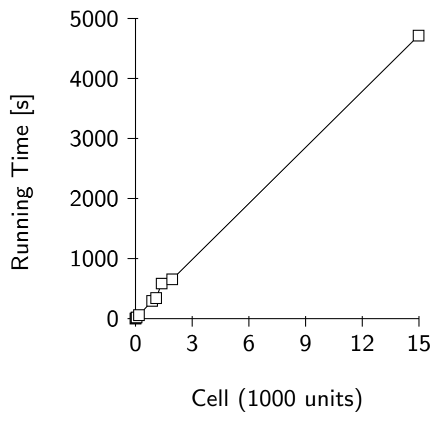
$$c_1^i = g_{\mathsf{ID}}^{r_1^i/(\mathsf{tok}+H(m_i))}, \quad c_2^i = g_{\mathsf{ID}}^{r_1^i}, \quad c_3^i = g^{r_2^i},$$

$$c_4^i = (m_i \parallel r_1^i) \oplus H_2(c_1^i \parallel c_2^i \parallel c_3^i \parallel e(P_{pub}, g_{\mathsf{ID}})^{r_2^i}),$$

where $r_1^i, r_2^i \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, with $|r_1^i| = l$. Finally, it returns $C^i = (c_1^i, c_2^i, c_3^i, c_4^i)$. Results on the total time consumption over the encryption in each experiment are shown as Table 2.

**Table 2.** Encryption performance

| Cells | RT [ms] |
|---|---|
| 7 | 3297 |
| 58 | 19789 |
| 183 | 57940 |
| 880 | 296337 |
| 1092 | 343131 |
| 1384 | 584450 |
| 1942 | 654629 |
| 14980 | 4715098 |



The time consumption of the $\mathsf{Enc}$ algorithm is linear with the number of cells to be encrypted. $\mathsf{Enc}$ contains one pairing and three exponent computations.

**Decryption Performance** We conducted the decryption operation on the random selected column to evaluate the performance of our decryption algorithm. For the $i_{th}$ decryption on captured $(c_1^i, c_2^i, c_3^i, c_4^i)$, it takes $\mathsf{ID}$ and the value of the set and computes

$$m_i \parallel r_1^i = c_4^i \oplus H_2(c_1^i \parallel c_2^i \parallel c_3^i \parallel e(c_3, d_{\mathsf{ID}})).$$

If

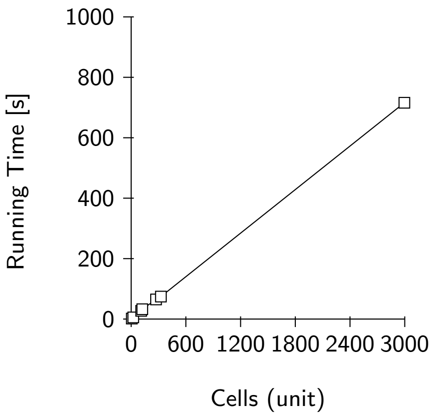
$$c_1^i = g_{\mathsf{ID}}^{r_1/(\mathsf{tok}+H(m_i))} \wedge c_2^i = g_{\mathsf{ID}}^{r_1^i},$$

it returns $m_i$. Otherwise, it returns $\perp$. The results on the total time consumption over the decryption in each experiment are shown as Table 3.

The time consumption of the $\mathsf{Dec}$ algorithm is linear with the number of cells to be decrypted. $\mathsf{Dec}$ contains one pairing and two exponent computations.

**Table 3.** Decryption performance

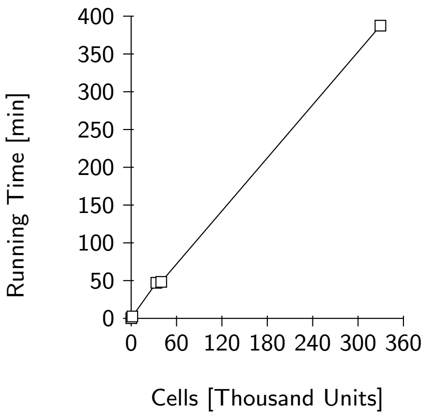
| Cells | RT [ms] |
|---|---|
| 7 | 1640 |
| 23 | 5432 |
| 110 | 27181 |
| 122 | 32544 |
| 273 | 64894 |
| 326 | 74367 |
| 2996 | 715763 |



**Equality Test Performance** We conducted the equality test between two tables who have the same column. For the $i_{th}$ equality test, it takes IDs and two ciphertexts which are $C^i = (c_1^i, c_2^i, c_3^i, c_4^i)$ and $C^j = (c_1^j, c_2^j, c_3^j, c_4^j)$, respectively, and checks whether $e(c_1^i, c_2^j)$ is equal to $e(c_1^j, c_2^i)$. Results on the total time consumption over the equality test in each experiment are shown in Table 4.

**Table 4.** Equality test performance

| Cells | RT [ms] |
|---|---|
| 161 | 16083 |
| 1540 | 149746 |
| 33306 | 2821123 |
| 39772 | 2898100 |
| 329560 | 23247358 |



The time consumption of the Test algorithm is linear with the number of cells to be conducted the equality test. Test contains only two pairing computations. The deviation of the performance might be caused by the instability of the CPU used in our simulation.

### 6.3 Comparison

There are some PKEET variants. We made a comparison on the efficiency of algorithms adopted in these schemes. "Exp" refers to the exponent computation. "P" refers to the pairing computation. "Auth." refers to the authorization.

**Table 5.** Comparing the efficiency of algorithms of variant PKEETs with our scheme

| PKEETs | IA | Enc | Dec | Test | Auth. | Security |
|---|---|---|---|---|---|---|
| [14] | N | 3Exp | 3Exp | 2P | N/A | OW-CCA |
| [12] | N | 4Exp | 2Exp | 4P | 3Exp | OW/IND-CCA |
| [13] | N | 5Exp | 2Exp | 4Exp | N/A | OW/IND-CCA |
| [6] | N | 4Exp | 4Exp | 6Exp+2P | 5Exp | OW/IND-CCA |
| [10] | N | 1P+5Exp | 1P+4Exp | 4P+2Exp | 3Exp | OW/IND-CCA |
| [8] | N | 6Exp | 2P+2Exp | 4P | 2Exp | OW-ID-CCA |
| Ours | Y | 1P+3Exp | 1P+2Exp | 2P | N/A | W-IND-ID-CCA |

The extended PKEET schemes cost three to four steps to conduct the equality test including analyzing trapdoor and inverse-computing trapdoor. In the contrast, our scheme only needs two pairing computations to conduct the equality test. The results in Table 5 indicate the improvement on efficiency in our scheme comparing with other schemes. In addition to efficiency, our scheme has shown the improvement on security and achieved the first W-IND-ID-CCA which is stronger than OW-ID-CCA.

## 7 Conclusions

The probabilistic public key encryption with equality test proposed by Yang et al. in 2010 at CT-RSA and its extended works are vulnerable to the insider attack launched by the semi-trusted server by guessing on the embedding message. The server can test whether the guessed message is equal to that contained in the target ciphertext. To solve this problem, we proposed a novel IBEET-IA scheme which a reasonable efficiency. In order to prove that our scheme is chosen ciphertext secure, we proposed a novel W-IND-ID-CCA security model under the defined insider attack. We also demonstrated its efficiency by experiments on a real database.

## Acknowledgement.

This work was partially supported by the National Natural Science Foundation of China (grant numbers: 61402184 and 61402376).

## References

1. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. pp. 56–73 (2004)
2. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. pp. 506–522 (2004)
3. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X.: Dual-server public-key encryption with keyword search for secure cloud storage. IEEE Trans. Information Forensics and Security 11(4), 789–798 (2016)
4. De Caro, A., Iovino, V.: jpbc: Java pairing based cryptography. In: Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011. pp. 850–855. IEEE, Kerkyra, Corfu, Greece, June 28 - July 1 (2011), http://gas.dia.unisa.it/projects/jpbc/
5. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
6. Huang, K., Tso, R., Chen, Y., Rahman, S.M.M., Almogren, A., Alamri, A.: PKE-AET: public key encryption with authorized equality test. Comput. J. 58(10), 2686–2697 (2015)
7. Jiang, P., Mu, Y., Guo, F., Wang, X., Wen, Q.: Online/offline ciphertext retrieval on resource constrained devices. Comput. J. 59(7), 955–969 (2016)
8. Ma, S.: Identity-based encryption with outsourced equality test in cloud computing. Inf. Sci. 328, 389–402 (2016)
9. Ma, S., Huang, Q., Zhang, M., Yang, B.: Efficient public key encryption with equality test supporting flexible authorization. IEEE Trans. Information Forensics and Security 10(3), 458–470 (2015)
10. Ma, S., Zhang, M., Huang, Q., Yang, B.: Public key encryption with delegated equality test in a multi-user setting. Comput. J. 58(4), 986–1002 (2015)
11. Mayer, D.A., Wetzel, S.: Verifiable private equality test: enabling unbiased 2-party reconciliation on ordered sets in the malicious model. In: 7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012. pp. 46–47 (2012)
12. Tang, Q.: Public key encryption schemes supporting equality test with authorization of different granularity. International Journal of Applied Cryptography 2(4), 304–321 (2012)
13. Tang, Q.: Public key encryption supporting plaintext equality test and user-specified authorization. Security and Communication Networks 5(12), 1351–1362 (2012)
14. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic public key encryption with equality test. In: Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings. pp. 119–131 (2010)