A Single Key Pair is Adequate for the Zheng Signcryption

Jia Fan[1,2], Yuliang Zheng[2], and Xiaohu Tang[1]

[1] Southwest Jiaotong University, 610031, P.R.China
[2] University of North Carolina at Charlotte, NC 28223, USA
fanjia@mars.swjtu.edu.cn, yzheng@uncc.edu, xhutang@ieee.org

Abstract. We prove that the original Zheng signcryption scheme published at Crypto'97, with a couple of minor tweaks, requires only a single public/private key pair for each user. That is the user can employ the same public/private key pair for both signcryption and unsigncryption in a provably secure manner. We also prove that the Zheng signcryption scheme allows a user to securely signcrypt a message to himself. Our first result confirms a long-held belief that signcryption reduces the overhead associated with public keys, while our second result foretells potential applications in cloud storage where one with a relatively less resourceful storage device may wish to off-load data to an untrusted remote storage network in a secure and unforgeable way.

Keywords: Public key, Security proof, Signcryption, Single key pair.

1 Introduction

The concept and first instantiation of signcryption were proposed by Zheng in 1997 [9]. As a cryptographic primitive, signcryption combines both the functions of public key encryption and those of digital signature, in such a way that its overhead is far less than that required by performing encryption and signature separately. At PKC'02, Baek, Steinfeld and Zheng [2] successfully established a security model for signcryption, and proved that with a couple of minor tweaks, the original Zheng signcryption scheme was indeed provably secure under commonly accepted computational assumptions. In the journal version [3] of the same paper, their security model was further enhanced and security proofs were made more rigorous. Their papers, however, still leave two interesting questions unanswered.

The first question has to do with the number of public/private key pairs a user has to keep in order to apply the Zheng signcryption in a provably secure manner. The security models and proofs presented in [2,3] all assume that a user holds two separate public/private key pairs. One of the two key pairs serves as a *sender signcryption key pair* for signcrypting messages originated from that user to other users, while the other key pair serves as a *receiver unsigncryption* *key pair* for unsigncrypting ciphertexts received by that user from other users. A natural question is whether the requirement of two separate public/private key pairs can be relaxed to a single key pair. An obvious benefit of the use of a single key pair is that it will minimize the cost associated with the creation and maintenance of public/private key pairs, especially the cost of public key verification prior to the execution of signcryption and unsigncryption.

The second question is whether the signcryption scheme can be employed by a user to securely signcrypt a message to the user himself. An ability to do so would have applications in emerging computing and communicating platforms such as cloud storage. Cloud storage is a model of networked data storage where data is stored on multiple virtual servers, generally hosted by third parties, rather than being hosted on dedicated servers. In practice, users with limited storage may wish to store data on a not always trusted cloud in a secure and unforgeable manner. In such a scenario, the user could signcrypt the data to himself first, then store the signcryptext to the cloud. When this user downloads the signcryptext from the cloud, it may check whether the signcryptext is valid, and decrypt the signcryptext if it is.

It turns out that the security models in [2, 3] in their original forms do not appear to be capable to address the two open questions. This calls for new ideas in security proofs, especially new security models that capture a real-world scenario where a single public/private key pair is used by each user as well as scenario where one wishes to signcrypt messages to oneself.

Our main contributions are to give affirmative answers to both questions outlined above. To this end we define a strengthened security model for signcryption allowing a user to have only a single public/private key pair and also allowing a user to signcrypt a message to himself. We then prove that the Zheng signcryption scheme, with a minor tweak, is indeed secure in that model, under commonly accepted assumptions including the Gap Diffie-Hellman, the Gap Discrete Logarithm and the random oracle assumptions.

2 Overview of the Zheng Signcryption Scheme

We focus our discussions on the SDSS-1 signcryption scheme proposed by Zheng [9]. Our security proofs apply to other schemes in the same family, including SDSS-2 and counterparts of SDSS-1 and SDSS-2 in other groups such as groups of points on an elliptic curve over a finite field [10].

We follow [3] in describing the Zheng scheme. A minor technical difference between our version and the version in [3] is that we add both sender and receiver's public keys as input to the G hash function. This minor tweak is useful during proof reductions which will become clear later in our description of proofs.

The signcryption scheme with the tweak is described in Tables 1 and 2. We use k to indicate a security parameter that determines other parameters such as the size of a key, the output length of a hash function and ultimately, the level of security of a concrete instantiation of a signcryption scheme in practice.

Table 1. Setup & KeyGen Algorithms

```
$Setup(1^k)$ by Trusted Authority $TA$:
1. Choose a random prime $q$ of $l_q$ bits.
2. Choose a random prime $p$ of $l_p$ bits such that $q|(p-1)$.
3. Choose an element $g \in Z_p^*$ such that $Ord_{Z_p^*}(g) = q$.
4. Choose a one-way hash function $G : \{0,1\}^* \rightarrow \{0,1\}^{l_G}$.
5. Choose a one-way hash function $H : \{0,1\}^* \rightarrow Z_q$.
6. Choose a symmetric key encryption scheme $\mathcal{SKE} = (E, D)$.
7. Let $cp = (k, p, q, g, G, H, \mathcal{SKE})$ be the common parameter.
```

```
$KeyGen(cp)$ by User $U$:
1. Choose $x_U \in Z_q^*$ uniformly at random.
2. Compute $y_U \leftarrow g^{x_U} \mod p$.
3. Let the public key $pk_U$ be $y_U$ and the private key $sk_U$ be $(x_U, y_U)$.
```

In this table, $l_p : N \rightarrow N$, $l_q : N \rightarrow N$ and $l_G : N \rightarrow N$ are functions of k determining the lengths in bits of p, q and an output of G respectively. $Ord_{Z_p^*}(g) = q$ means that the order of g in the multiplicative group of Z_p^* is q, and $\mathcal{SKE} = (E, D)$ is a one-time symmetric key encryption scheme with message, key and ciphertext spaces being $\mathcal{SP}_m$, $\{0,1\}^{l_G}$ and $\mathcal{SP}_c$ respectively.

Table 2. Signcryption & Unsigncryption Algorithms

```
$Signcryption(cp, m, sk_S, pk_R)$
by Sender $S$:
1. Parse $sk_S$ as $(x_S, y_S)$, $pk_R$ as $y_R$.
2. Choose $x \in Z_q^*$ uniformly at random.
3. Compute $K \leftarrow {y_R}^x \mod p$.
4. Compute $\tau \leftarrow G(y_S, y_R, K)$.
5. Compute $c \leftarrow E_\tau(m)$.
6. Compute $r \leftarrow H(m, y_S, y_R, K)$.
7. If $r + x_S = 0 \mod q$, return to Step 2.
8. Compute $s \leftarrow x/(r + x_S) \mod q$.
9. Output $\sigma \leftarrow (c, r, s)$ as the signcryptext.
```

```
$Unsigncryption(cp, \sigma, pk_S, sk_R)$
by Receiver $R$:
1. Parse $sk_R$ as $(x_R, y_R)$, $pk_S$ as $y_S$.
2. Parse $\sigma$ as $(c, r, s)$.
3. Compute $w \leftarrow (y_S \cdot g^r)^s \mod p$.
4. Compute $K \leftarrow w^{x_R} \mod p$.
5. Compute $\tau \leftarrow G(y_S, y_R, K)$.
6. Compute $m \leftarrow D_\tau(c)$.
7. If $H(m, y_S, y_R, K) = r$, return $m$;
otherwise return $Reject$.
```

We assume $m \in \mathcal{SP}_m$ in the signcryption algorithm, and $\sigma \in \mathcal{SP}_c \times Z_q \times Z_q^*$ in the unsigncryption algorithm. In practice appropriate tests are carried out first to ensure that these conditions are met. With the unsigncryption algorithm, *Reject* is interpreted as a special symbol indicating that the signcryptext is invalid.

3 Security Model

We now introduce a stronger security model that is extended from the model proposed by Baek *et al.* [3]. Major differences between the two security models are outlined below.

First, our new model allows the use of a single public/private key pair by a user. This is achieved by permitting a target user (with a single public/private key pair) in an attack game to be both a sender and a receiver. This modification makes it possible for an adversary to make signcryption queries with any target user as a sender and unsigncryption queries with any target user as a receiver. We note that in the original security model by Baek *et al*, a target user always has a fixed role, being either a sender or a receiver.

Second, our model adds a security consideration for the case where one signcrypts a message to oneself. In an attack game for confidentiality, an adversary is given two target users, A and B. We allow the adversary to attack on $(S^*, R^*) \in \{(A, B), (A, A), (B, A), (B, B)\}$, where S^* is the sender and R^* is the receiver. By contrast, the model by Baek *et al.* allows only $(S^*, R^*) = (A, B)$. And in an attack game for unforgeability, an adversary is given one target user A. We allow the adversary to attack on (S^*, R^*) where $S^* = A$ and R^* can be an arbitrary user including $R^* = A$, while the model by Baek *et al.* does not allow $R^* = A$.

According to the adversary's capability, An *et al.* [1] divide the security model into two classes, called the *insider* setting and the *outsider* setting respectively. In the outsider setting, an adversary has access to neither sk_{S^*} nor sk_{R^*}. In comparison, the only restriction on an adversary in the insider setting is that it is not allowed to have access to sk_{R^*}. Since our main goal in this paper is to prove the Zheng signcryption scheme is secure in the "outsider" setting for confidentiality, we will define unforgeability in the insider setting, and confidentiality in the outsider setting.

Throughout this paper we will use the term of a *negligible* function to indicate any function in an appropriate security parameter k that vanishes faster than the inverse of any integer-valued polynomial in the same parameter k when k is sufficiently large.

3.1 Syntax of Signcryption

A generic signcryption system $\mathcal{SC}$ consists of four algorithms as follows:

- $Setup(1^k)$**:** It takes as input a security parameter 1^k and generates a common parameter cp for an entire system under consideration. It is run by a trusted authority.
- $KeyGen(cp)$**:** It takes as input a system-wide common parameter cp, outputs a pair of public/private keys (pk_U, sk_U) for a user U. This algorithm is run by users within the system, independently of one another.
- $Signcryption(cp, m, sk_S, pk_R)$**:** When a sender S plans to communicate a message $m \in \mathcal{SP}_m$ to a receiver R, where $\mathcal{SP}_m$ is the message space, he runs this algorithm to generate a signcryptext σ from m, a common parameter cp, his private key sk_S and the receiver R's public key pk_R.
- $Unsigncryption(cp, \sigma, pk_S, sk_R)$: When a receiver R receives a signcryptext σ from a sender S, he runs this algorithm with σ, the public parameter cp, the sender S's public key pk_S, and his private key sk_R as input. The algorithm outputs a message m if σ is valid, or a special symbol *Reject* otherwise.

For a signcryption scheme to be useful in practice, we further require that for any plaintext m, any sender S and any receiver R, we have

$$m = Unsigncryption(cp, Signcryption(cp, m, sk_S, pk_R), pk_S, sk_R).$$

3.2 Definition of Confidentiality

We follow an established definition, called indistinguishability under chosen ciphertext attack (IND-CCA) to define confidentiality for signcryption as indistinguishability under chosen signcryptext and plaintext attack (IND-CSPA). This is done by defining an attack game, called an IND-CSPA game.

Let k be the security parameter of the scheme, A and B be two target users. The IND-CSPA game is played between an IND-CSPA adversary and its environment Σ which contains an IND-CSPA challenger and two oracles, namely a signcryption oracle and an unsigncryption oracle. Specifically, the IND-CSPA game proceeds as follows:

– Stage 1: The challenger computes $cp \leftarrow Setup(1^k)$; $(pk_A, sk_A) \leftarrow KeyGen(cp)$; $(pk_B, sk_B) \leftarrow KeyGen(cp)$. It then equips the signcryption and unsigncryption oracles with (sk_A, sk_B) and gives (cp, pk_A, pk_B) to the adversary.
– Stage 2: The adversary makes a sequence of adaptive queries. Each query is one of two types:
  1. Signcryption query: the adversary submits (m, pk_S, pk_R) to the challenger, where $m \in \mathcal{SP}_m$, $pk_S \in \{pk_A, pk_B\}$ and pk_R can be an arbitrary public key in the system including $pk_R \in \{pk_A, pk_B\}$. The challenger forwards
  (m, pk_S, pk_R) to the signcryption oracle which then returns to the challenger with an outcome of $Signcryption(cp, m, sk_S, pk_R)$. Finally, the challenger passes this answer to the adversary.
  2. Unsigncryption query: the adversary submits (σ, pk_S, pk_R) to the challenger, where σ is a signcryptext, $pk_R \in \{pk_A, pk_B\}$, and pk_S can be an arbitrary public key in the system including $pk_S \in \{pk_A, pk_B\}$. The challenger forwards (σ, pk_S, pk_R) to the unsigncryption oracle which then returns to the challenger with an outcome of $Unsigncryption$ (cp, σ, pk_S, sk_R). Finally, the challenger passes this answer to the adversary.
– Stage 3: The adversary submits $(m_0, m_1, pk_{S^*}, pk_{R^*})$ to the challenger where $m_0, m_1 \in \mathcal{SP}_m$ are of equal length, and $pk_{S^*}, pk_{R^*} \in \{pk_A, pk_B\}$. The challenger chooses a random bit $\beta \in \{0, 1\}$. Then it forwards $(m_\beta, pk_{S^*}, pk_{R^*})$ to the signcryption oracle which then returns to the challenger with a signcryptext σ^* which is an outcome of $Signcryption(cp, m_\beta, sk_{S^*}, pk_{R^*})$. The challenger then passes σ^* to the adversary as a challenge signcryptext.
– Stage 4: This is identical to Stage 2, except that the adversary can not query an unsigncryption with $(\sigma^*, pk_{S^*}, pk_{R^*})$.
– Stage 5: The adversary outputs a bit β' as his guess for β and pass it over to the challenger. The challenger then checks whether $\beta = \beta'$. If it is, the adversary wins the challenge.

For an IND-CSPA adversary $\mathcal{A}$ running in time t, making at most j_s signcryption queries and j_u unsigncryption queries, we define the advantage of $\mathcal{A}$ in winning the challenge as $Adv_{\mathcal{SC},\mathcal{A}}^{ind-cspa}(t, j_s, j_u) = |Pr[\beta = \beta'] - 1/2|$. And we define $\epsilon_{t,j_s,j_u}^{ind-cspa}$ to be the maximum value of $Adv_{\mathcal{SC},\mathcal{A}}^{ind-cspa}(t, j_s, j_u)$ over all IND-CSPA adversaries with the same *resources* parameter (t, j_s, j_u).

Definition 1. *We say that a signcryption scheme $\mathcal{SC}$ is IND-CSPA secure if for any IND-CSPA adversary that runs in time t, makes at most j_s signcryption queries and j_u unsigncryption queries, the maximum advantage $\epsilon_{t,j_s,j_u}^{ind-cspa}$ is negligible in k, where t, j_s and j_u are all polynomials in k.*

3.3 Definition of Unforgeability

Unforgeability is defined as existential unforgeability against chosen signcryptext and plaintext attack (EUF-CSPA), which follows the established definition of existential unforgeability against chosen message attack (EUF-CMA). This is done by defining an attack game, called an EUF-CSPA game.

Let k be the security parameter of the scheme, A be a target user. The EUF-CSPA game is played between an EUF-CSPA adversary and its environment Σ which contains an EUF-CSPA challenger and two oracles, one being a signcryption oracle and the other an unsigncryption oracle. The EUF-CSPA game proceeds as follows:

- Stage 1: The challenger computes $cp \leftarrow Setup(1^k)$; $(pk_A, sk_A) \leftarrow KeyGen(cp)$. It then equips the signcryption and unsigncryption oracles with sk_A and gives (cp, pk_A) to the adversary.
- Stage 2: It is mostly the same as Stage 2 in the IND-CSPA game described above, except that in this case there is no pk_B.
- Stage 3: The adversary passes $(\sigma^*, pk_{S^*}, pk_{R^*}, sk_{R^*})$ to the challenger, where σ^* is a signcryptext, $pk_{S^*} = y_A$, pk_{R^*} can be an arbitrary public key in the system including $pk_{R^*} = pk_A$, and sk_{R^*} is the corresponding private key of pk_{R^*}. The challenger then checks whether the outcome of $Unsigncryption(cp, \sigma^*, pk_{S^*}, sk_{R^*})$ is a special symbol *Reject* or a message $m^* \in \mathcal{SP}_m$. If the outcome is m^* and the adversary has never made a signcryption query on $(m^*, pk_{S^*}, pk_{R^*})$, then the adversary wins the challenge.

For an adversary $\mathcal{A}$ running in time t, making at most j_s signcryption queries and j_u unsigncryption queries, we define the advantage of $\mathcal{A}$ in winning the challenge as $Adv_{\mathcal{SC},\mathcal{A}}^{euf-cspa}(t, j_s, j_u) = Pr[\mathcal{A}\ wins]$, where "$\mathcal{A}$ *wins*" denotes an event that adversary $\mathcal{A}$ wins the challenge in the above attack game. And we define $\epsilon_{t,j_s,j_u}^{euf-cspa}$ to be the maximum of $Adv_{\mathcal{SC},\mathcal{A}}^{euf-cspa}(t, j_s, j_u)$ over all EUF-CSPA adversaries with the same resource parameter (t, j_s, j_u).

Definition 2. *We say that a signcryption scheme $\mathcal{SC}$ is EUF-CSPA secure if for any EUF-CSPA adversary running in time t, and making at most j_s signcryption queries and at most j_u unsigncryption queries, the maximum advantage $\epsilon_{t,j_s,j_u}^{euf-cspa}$ is negligible in k, where t, j_s and j_u are all polynomials in k.*

4 Assumptions and Primitives

4.1 Problems and Assumptions

Let $\mathcal{G}$ be a finite multiplicative group with g being a generator of the group. The Discrete Logarithm (DL) problem is one where an attacker is given $(g, y) \in \mathcal{G}^2$, asked to find an x such that $y = g^x$ in $\mathcal{G}$. The well-known Diffie-Hellman (DH) problem has two different flavors: a computational one and a decisional one. With the Computational Diffie-Hellman (CDH) problem, an attacker is given three elements $(g, g^a, g^b) \in \mathcal{G}^3$ for unknown a and b, and asked to compute g^{ab}. In contrast, with the Decisional Diffie-Hellman (DDH) problem an attacker is given four elements $(g, g^a, g^b, z) \in \mathcal{G}^4$, for unknown a and b, and asked to tell whether $z = g^{ab}$.

The CDH problem has a gap based version in which an attacker is granted access to a powerful oracle, named DDH oracle, that solves the DDH problem [7]. This new problem is called the Gap Diffie-Hellman (GDH) problem. A gap based version of the DL problem can be obtained in a similar way. In this paper, we follow [3] to define the Gap Discrete Logarithm (GDL) problem as one in which an attacker has access to a *restricted* oracle for the DDH problem. Similar to the DDH oracle, a restricted DDH oracle also answers whether a given quadruple is a DH quadruple or not. However, the restricted DDH oracle only accepts queries on $(g, y, ., .) \in \mathcal{G}^4$ where (g, y) is the input of the adversary.

The CDH problem has a number of interesting variants. In one variant, an attacker is given $(g, g^a) \in \mathcal{G}^2$ with an unknown a and asked to compute $y = g^{a^2}$. It turns out that this variant is equivalent to the CDH problem [4].

In our proofs we will employ a new variant of the CDH problem in which an attacker is given $(g, g^a, g^b) \in \mathcal{G}^3$ for unknown a and b, and attempts to output one of $(g^{a^2}, g^{b^2}, g^{ab})$. The attacker is considered successful as long as its output is one of the three possible values. We call this new problem the *extended Computational Diffie-Hellman* problem or the eCDH problem for short. Clearly the eCDH problem is computationally equivalent to the CDH problem. A gap based version of the eCDH problem is defined by allowing an attacker to have access to a DDH oracle. Let us call it the *extended GDH* problem or the eGDH problem for short. Naturally, the equivalence of the CDH and eCDH problems is carried over to their gap based versions. That is, the following lemma is true.

Lemma 1. *The GDH problem and the eGDH problem are equivalent.*

Assumptions related to the above mentioned problems are defined by stating that no attacker that runs in polynomial time in the size of the group can successfully solve the respective problem with a non-negligibly success probability. In particular, according to Lemma 1, we claim that the eGDH assumption are equivalent to the GDH assumption.

4.2 One-Time Symmetric Key Encryption

A one-time symmetric key encryption system $\mathcal{SKE}$ [6] consists of two bijective and deterministic algorithms E and D.

– $E_\tau(m)$: On input a key τ, a plaintext m, it outputs a ciphertext $c \leftarrow E_\tau(m)$.
– $D_\tau(c)$: On input a key τ, a ciphertext c, it outputs a plaintext $m \leftarrow D_\tau(c)$.

In the above, $\tau \in \mathcal{SP}_\tau$, $m \in \mathcal{SP}_m$ and $c \in \mathcal{SP}_c$, where the sizes of spaces $\mathcal{SP}_\tau$, $\mathcal{SP}_m$ and $\mathcal{SP}_c$ are all determined by a security parameter k. And it is required that for all $m \in \mathcal{SP}_m$ and $\tau \in \mathcal{SP}_\tau$, $m = D_\tau(E_\tau(m))$.

We will use a one-time symmetric key encryption with the security of passive indistinguishability of $\mathcal{SKE}$ (PI-SKE). In a PI-SKE attack game, a passive attacker is given $(k, \mathcal{SKE})$, and then submits two equal length messages (m_0, m_1) to get a ciphertext c where $c \leftarrow E_\tau(m_\beta)$, β is a random bit. PI-SKE security states that, any passive attacker running in polynomial time cannot determine which of the two messages was chosen.

4.3 One-Way Hash Functions

Our proofs rely on the random oracle methodology [5]. In other words we assume that each one-way hash function used in the Zheng signcryption scheme behaves like a random oracle, a mathematical function mapping every possible query to a random response from its output domain.

5 Security Proofs

Our proofs for confidentiality and unforgeability apply the game based technique. For each proof, we describe a sequence of $n+1$ games, from Game 0 to Game n (n is a constant). Game 0 is the normal attack game in the security definition. We use a sequence of simulators (from Game 1 to Game n) to replace the challenger. Game $i+1$ and Game i ($0 \le i \le n-1$) are mostly the same, except that the simulator's behavior in Game $i+1$ is a little bit different from the simulator's (or the challenger's when $i = 0$) behavior in Game i.

Define S_i to be an event that the adversary wins the challenge in Game i. To analyze the relation between $Pr[S_i]$ and $Pr[S_{i+1}]$, we make use of two techniques introduced by Shoup [8], namely bridging step and transition based on a failure event.

1. Bridging Step: The change from Game i to Game $i+1$ is a bridging step means that the change is only conceptual. From the adversary's point of view, these two games proceed identically. Therefore, in this case we have $Pr[S_i] = Pr[S_{i+1}]$.
2. Transition Based on a Failure Event: The change from Game i to Game $i+1$ is a transition based on a failure event means that from the adversary's point of view, these two games proceed identically unless a certain "failure event" occurs. We can then apply a so-called Difference Lemma [8]:

   Lemma 2. *(Difference Lemma): Let S_1, S_2 and F be events defined on some probability spaces. Suppose that the event $S_1 \wedge \neg F$ occurs if and only if $S_2 \wedge \neg F$ occurs. Then $\mid Pr[S_1] - Pr[S_2] \mid \le Pr[F]$.*

We have in this case $|Pr[S_{i+1}] - Pr[S_i]| \leq Pr[Failure\ Event\ Occurs]$.

In each proof, we make sure that for all i ($0 \leq i \leq n-1$), the change from Game i to Game $i+1$ is either a bridging step or a transition based on a failure event which occurs with at most a negligible probability in k. In Game n, we show that the adversary's advantage in winning the challenge is negligible in k. Finally, from the results in all the games, we can arrive at our desired conclusion that the adversary's advantage in winning Game 0 (the normal attack game) is also negligible in k.

We define $\langle g \rangle$ be a group generated by g. The security proofs of unforgeability and confidentiality for the Zheng signcryption are as follows.

5.1 Proof of Unforgeability

Theorem 1. *Let H and G be two hash functions modeled as random oracles. Then under the GDL assumption in $\langle g \rangle$, the Zheng signcryption scheme is EUF-CSPA secure. Specifically, let k be a security parameter of signcryption, $\mathcal{A}$ be an EUF-CSPA adversary that runs in time t, and makes at most j_s signcryption queries, j_u unsigncryption queries, j_g hash queries to G and j_h hash queries to H, where t, j_s, j_u, j_g and j_h are all polynomials in k. Then the maximum advantage $\epsilon^{euf-cspa}_{t,j_s,j_u}$ of the adversary satisfies the following condition:*

$$\epsilon^{euf-cspa}_{t,j_s,j_u} \leq \frac{j_s(j_g + j_h + 3j_u + 2j_s) + 2j_h + j_u + 1}{q} + 2 \cdot \sqrt{j_h \cdot \epsilon^{gdl}_{t_{gdl},j_{gdl}}}$$

where $\epsilon^{gdl}_{t_{gdl},j_{gdl}}$ is negligible in k for all sufficiently large k.

Before diving into details of the proof of Theorem 1, we review in Table 3 an assumption introduced in [3], which is renamed as the Random Beacon GDL assumption (or rbGDL assumption for short). The following Lemma 3 is also from [3] which shows an equivalence relationship between the rbGDL assumption and the GDL assumption.

Lemma 3. *Any algorithm $\mathcal{A}_{rbgdl}$ attacking the rbGDL assumption with run-time t_{rbgdl}, j_{rbgdl} restricted DDH queries, j_r Random Beacon queries, and success probability $Adv^{rbgdl}_{\mathcal{RBGDL},\mathcal{A}_{rbgdl}}(t_{rbgdl}, j_{rbgdl}, j_r) \geq 2j_r/q$ can be converted into an algorithm $\mathcal{A}_{gdl}$ attacking the GDL assumption with run-time $t_{gdl} = 2t_{rbgdl} + O(q^2)$, $j_{gdl} = 2j_{rbgdl}$ restricted DDH queries, and success probability*

$$Adv^{gdl}_{\mathcal{GDL},\mathcal{A}_{gdl}}(t_{gdl}, j_{gdl}) \geq \frac{1}{j_r}\left(\frac{Adv^{rbgdl}_{\mathcal{RBGDL},\mathcal{A}_{rbgdl}}(t_{rbgdl}, j_{rbgdl}, j_r)}{2} - \frac{j_r}{q}\right)^2.$$

Proof of Theorem 1. We describe our proof in a sequence of seven games, from Game 0 to Game 6 as follows. We define $\mathcal{S}_i$ ($1 \leq i \leq 6$) to be the simulator in Game i.

Table 3. The rbGDL Assumption

Random Beacon Gap Discrete Logarithm (rbGDL) Assumption [3]
Given a pair of elements (g, g^a) in $\mathcal{G}$, g is a generator of $\mathcal{G}$, $Ord_{\mathcal{G}}(g) = q$, $a \in \{0, ..., q-1\}$. With the help of a Restricted DDH Oracle and a Random Beacon,   *A Random Beacon takes as input a pair of elements* $(y[i], K[i]) \in \mathcal{G}^2$ *(*$y[i] \neq 1$*), outputs uniformly a random independent number* $r[i] \in \{0, ..., q-1\}$. $i \in \{1, ..., j_r\}$ where j_r is total number of Random Beacon queries been made.   it is computationally intractable to compute the value of $(r[i^*], s^*, i^*)$, satisfying $K[i^*] = y[i^*]^{s^*(r[i^*]+a)}$.

The difference between a random beacon and a random oracle is that a random beacon returns a random and independent response even for the same input.

Game 0 (EUF-CSPA Game in the Random Oracle Model): This game is the EUF-CSPA game defined in Section 3.3 in the random oracle model. Therefore, we have

$$Adv^{euf-cspa}_{\mathcal{SC},\mathcal{A}}(t, j_s, j_u) = Pr[S_0]. \tag{1}$$

Game 1 (Apply G_{sim} to Simulate the G Random Oracle): In this game, $\mathcal{S}_1$ behaves mostly the same as $\mathcal{C}$, except that $\mathcal{S}_1$ additionally runs an algorithm G_{sim} to simulate the G random oracle. In order to simulate the G random oracle, $\mathcal{S}_1$ holds two lists, called $Glist_1$ and $Glist_2$ respectively, which are both initially empty. Records on $Glist_1$ are generated by G_{sim}, while records on $Glist_2$ are generated by $Signcryption_{sim}$ and $Unsigncryption_{sim}$ which will be applied in later games. The ν-th record on $Glist_1$ is in form of $(y_{S_\nu}, y_{R_\nu}, K_\nu, \tau_\nu)$, and the μ-th record on $Glist_2$ is in form of $(r_\mu, s_\mu, y_{S_\mu}, y_{R_\mu}, \tau_\mu)$. l_{Glist_1} and l_{Glist_2} denote the total number of records on $Glist_1$ and $Glist_2$ respectively.

– When the i-th hash query is made on (y_S, y_R, K) to the G random oracle, G_{sim} runs the following steps:
  1. If $Glist_1$ is not empty, then from $\nu = 1$ to $\nu = l_{Glist_1}$ do
     (a) take the value of $(y_{S_\nu}, y_{R_\nu}, K_\nu, \tau_\nu)$ which is the ν-th record on $Glist_1$;
     (b) if $(y_{S_\nu}, y_{R_\nu}, K_\nu) = (y_S, y_R, K)$, return τ_ν;
     (c) $\nu = \nu + 1$.
  2. If $Glist_2$ is not empty, then from $\mu = 1$ to $\mu = l_{Glist_2}$do
     (a) take the value of $(r_\mu, s_\mu, y_{S_\mu}, y_{R_\mu}, \tau_\mu)$ which is the μ-th record on $Glist_2$;
     (b) if $y_S = y_A$ and $(y_S, y_R) = (y_{S_\mu}, y_{R_\mu})$, check whether a quadruple $(g, y_S, {y_R}^{s_\mu}, \frac{K}{{y_R}^{s_\mu \cdot r_\mu}})$ is a DH quadruple in $\langle g \rangle$; if it is, return τ_μ;
     (c) if $y_R = y_A$ and $(y_S, y_R) = (y_{S_\mu}, y_{R_\mu})$, check whether a quadruple $(g, y_R, (y_S g^{r_\mu})^{s_\mu}, K)$ is a DH quadruple in $\langle g \rangle$; if it is, return τ_μ;
     (d) $\mu = \mu + 1$.
  3. Choose $\tau \in \{0,1\}^{l_G}$ uniformly at random, add (y_S, y_R, K, τ) to the end of $Glist_1$ and return τ.

It is easy to check that the change from Game 0 to Game 1 is a bridging step, therefore,

$$Pr[S_1] = Pr[S_0]. \tag{2}$$

Game 2 (Apply H_{sim} to Simulate the H Random Oracle): In this game, $\mathcal{S}_2$ behaves mostly the same as $\mathcal{S}_1$, except that $\mathcal{S}_2$ additionally runs an algorithm H_{sim} to simulate the H random oracle. In order to simulate the H random oracle, $\mathcal{S}_2$ holds another two lists, called $Hlist_1$ and $Hlist_2$ respectively, which are both initially empty. Records on $Hlist_1$ are generated by H_{sim}, while records on $Hlist_2$ are generated by $Signcryption_{sim}$ which will be applied in later games. l_{Hlist_1} and l_{Hlist_2} denote the total number of records on $Hlist_1$ and $Hlist_2$ respectively. The ν-th record on $Hlist_1$ is in form of $(m_\nu, y_{S_\nu}, y_{R_\nu}, K_\nu, r_\nu)$, and the μ-th record on $Glist_2$ is in form of $(r_\mu, s_\mu, m_\mu, y_{S_\mu}, y_{R_\mu}, r'_\mu)$.

- When the i-th hash query is made on (m, y_S, y_R, K) to the H random oracle, H_{sim} runs the following steps:
  1. If $Hlist_1$ is not empty, then from $\nu = 1$ to $\nu = l_{Hlist_1}$ do
     (a) take the value of $(m_\nu, y_{S_\nu}, y_{R_\nu}, K_\nu, r_\nu)$ which is the ν-th record on $Hlist_1$;
     (b) if $(m_\nu, y_{S_\nu}, y_{R_\nu}, K_\nu) = (m, y_S, y_R, K)$, then return r_ν;
     (c) $\nu = \nu + 1$.
  2. If $Hlist_2$ is not empty, then from $\mu = 1$ to $\mu = l_{Hlist_2}$do
     (a) take the value of $(r_\mu, s_\mu, m_\mu, y_{S_\mu}, y_{R_\mu}, r'_\mu)$ which is the μ-th record on $Hlist_2$;
     (b) if $y_S = y_A$ and $(m, y_S, y_R) = (m_\mu, y_{S_\mu}, y_{R_\mu})$, check whether a quadruple $(g, y_S, {y_R}^{s_\mu}, \frac{K}{{y_R}^{s_\mu \cdot r_\mu}})$ is a DH quadruple in $\langle g \rangle$; if it is, return r'_μ;
     (c) if $y_R = y_A$ and $(m, y_S, y_R) = (m_\mu, y_{S_\mu}, y_{R_\mu})$, check whether a quadruple $(g, y_R, (y_S g^{r_\mu})^{s_\mu}, K)$ is a DH quadruple in $\langle g \rangle$; if it is, return r'_μ;
     (d) $\mu = \mu + 1$.
  3. If $y_S = y_A$, it computes $r \leftarrow R'(y_R, K)$, otherwise it chooses $r \in Z_q$ uniformly at random; add (m, y_S, y_R, K, r) to the end of $Hlist_1$ and return r.
     Here, R' is an algorithm that has the same output distribution as a random beacon R. For any input (even with the same input as before), R' chooses $r \in Z_q$ uniformly at random, and outputs r.

It is also easy to check that the change from Game 1 to Game 2 is a bridging step, therefore,

$$Pr[S_2] = Pr[S_1]. \tag{3}$$

Game 3 (Apply $Signcryption_{sim}$ to Simulate the Signcryption Oracle): In this game $\mathcal{S}_3$ behaves mostly the same as $\mathcal{S}_2$, except that $\mathcal{S}_3$ additionally runs an algorithm $Signcryption_{sim}$ to simulate the signcryption oracle.

– When the i-th signcryption query is made on (m, pk_S, pk_R) to the signcryption oracle, $Signcryption_{sim}$ runs as follows:
  1. Parse pk_S as y_S, pk_R as y_R.
  2. Choose $r \in Z_q$, $s \in Z_q^*$, $\tau \in \{0,1\}^{l_G}$ uniformly at random.
  3. If $g^r y_S = 1 \mod p$, jump to Step 2.
  4. If $Glist_2$ is not empty, then from $\mu = 1$ to $\mu = l_{Glist_2}$ do
     (a) take the value of $(r_\mu, s_\mu, y_{S_\mu}, y_{R_\mu}, \tau_\mu)$ which is the μ-th record on $Glist_2$;
     (b) if $(y_{S_\mu}, y_{R_\mu}) = (y_S, y_R)$, $(y_S g^r)^s = (y_S g^{r_\mu})^{s_\mu}$ and $\tau_\mu \neq \tau$, then return *Reject*;
     (c) $\mu \leftarrow \mu + 1$.
  5. If $Hlist_2$ is not empty, then from $\mu = 1$ to $\mu = l_{Hlist_2}$ do
     (a) take the value of $(r_\mu, s_\mu, m_\mu, y_{S_\mu}, y_{R_\mu}, r'_\mu)$ which is the μ-th record on $Hlist_2$;
     (b) if $(m_\mu, y_{S_\mu}, y_{R_\mu}) = (m, y_S, y_R)$, $(y_S g^r)^s = (y_S g^{r_\mu})^{s_\mu}$ and $r'_\mu \neq r$, then return *Reject*;
     (c) $\mu = \mu + 1$.
  6. Add (r, s, y_S, y_R, τ) to the end of $Glist_2$, (r, s, m, y_S, y_R, r) to the end of $Hlist_2$;
  7. Compute $c \leftarrow E_\tau(m)$;
  8. Return $\sigma = (c, r, s)$.

If the following four conditions are all satisfied, then it is easy to check that $Signcryption_{sim}$ has the same output distribution as the signcryption oracle:

– $Signcryption_{sim}$ does not change the output distribution of G_{sim} and H_{sim}.
– $Signcryption_{sim}$ does not return *Reject*.
– $\tau = G_{sim}(y_S, y_R, K)$ with $K = g^{(x_S + r)\cdot s \cdot x_R}$ when $Signcryption_{sim}$ does not return *Reject*.
– $r = H_{sim}(m, y_S, y_R, K)$ with $K = g^{(x_S + r)\cdot s \cdot x_R}$ when $Signcryption_{sim}$ does not return *Reject*.

In the following, we analyze all the above conditions one by one.

1. Adding (r, s, y_S, y_R, τ) to $Glist_2$, (r, s, m, y_S, y_R, r) to $Hlist_2$ does not change the output distribution of G_{sim} and H_{sim}, since $\tau \in \{0,1\}^{l_G}$ which may be used as an output for G_{sim} and $r \in Z_q$ which may be used as an output for H_{sim} are all chosen uniformly at random. Therefore, $Signcryption_{sim}$ does not change the output distribution of G_{sim} and H_{sim}.
2. For the i-th signcryption query, the probability that it returns *Reject* at Step 4 is at most $\frac{j_s + j_u}{q}$, since $(y_S g^r)^s$ is uniformly and randomly distributed in $\langle g \rangle$ and $l_{Glist_2} \leq j_s + j_u$. Similarly, we have for the i-th signcryption query, the probability that it returns *Reject* at Step 5 is also at most $\frac{j_s}{q}$ since $l_{Hlist_2} \leq j_s$. Therefore, for the i-th signcryption query, the probability $Signcryption_{sim}$ does not return *Reject* is at most $\frac{2j_s + j_u}{q}$. Then, the probability that the second condition is not satisfied during some signcryption query is at most $\frac{j_s(2j_s + j_u)}{q}$.

3. For the i-th signcryption query in which $Signcryption_{sim}$ does not return $Reject$, $\tau \neq G_{sim}(y_S, y_R, K)$ if and only if G_{sim} has been run on (y_S, y_R, K) before the i-th signcryption query and the corresponding output does not equal to τ. The probability that G_{sim} has been run on (y_S, y_R, K) before the i-th signcryption query is at most $\frac{j_u+j_g}{q}$, since K is randomly and uniformly distributed in $\langle g \rangle$ and G_{sim} must have been run for at most $j_u + j_g$ times (j_u times called by the unsigncryption oracle, j_g times called directly by the challenger) before the i-th signcryption query. Therefore, the probability that the third condition is not satisfied during some signcryption query is at most $\frac{j_s(j_u+j_g)}{q}$.
4. Following a very similar analysis as for the third condition, we have the probability that the fourth condition is not satisfied during some signcryption query is at most $\frac{j_s(j_u+j_h)}{q}$.

We define a certain event F_1 to be that at least one of the above conditions is not satisfied. From the above analysis, we have

$$Pr[F_1] \leq \frac{j_s(j_g + j_h + 3j_u + 2j_s)}{q}. \tag{4}$$

Now it is clear that the signcryption oracle and $Signcryption_{sim}$ has the same output distribution unless F_1 occurs. Moreover, $Signcryption_{sim}$ does not change the output distribution of G_{sim} and H_{sim}. Thus, the change from Game 2 and Game 3 is a transition based on a failure event F_1. We have

$$|Pr[S_3] - Pr[S_2]| \leq Pr[F_1], \tag{5}$$

Game 4 (Apply $Unsigncryption_{sim}$ to Simulate the Unsigncryption Oracle): In this game $\mathcal{S}_4$ behaves mostly the same as $\mathcal{S}_3$, except that $\mathcal{S}_4$ additionally runs an algorithm $Unsigncryption_{sim}$ to simulate the unsigncryption oracle as follows:

– When the i-th unsigncryption query is made on (σ, pk_S, pk_R) to the unsigncryption oracle, $Unsigncryption_{sim}$ runs as follows:
  1. Parse pk_S as y_S, pk_R as y_R.
  2. Parse σ as (c, r, s).
  3. Compute $w \leftarrow (y_S g^r)^s \mod p$.
  4. If $Glist_1$ is not empty, then from $\nu = 1$ to $\nu = l_{Glist_1}$ do
     (a) take the value of $(y_{S_\nu}, y_{R_\nu}, K_\nu, \tau_\nu)$ which is the ν-th record on $Glist_1$;
     (b) if $(y_{S_\nu}, y_{R_\nu}) = (y_S, y_R)$ and (g, y_R, w, K_ν) is a DH tuple in $\langle g \rangle$, compute $\hat{\tau} \leftarrow \tau_\nu$ and jump to Step 7;
     (c) $\nu \leftarrow \nu + 1$.
  5. If $Glist_2$ is not empty, then from $\mu = 1$ to $\mu = l_{Glist_2}$ do
     (a) take the value of $(r_\mu, s_\mu, y_{S_\mu}, y_{R_\mu}, \tau_\mu)$ which is the μ-th record on $Glist_2$;
     (b) if $(y_{S_\mu}, y_{R_\mu}) = (y_S, y_R)$ and $(y_S g^{r_\mu})^{s_\mu} = (y_S g^r)^s$, then compute $\hat{\tau} \leftarrow \tau_\mu$ and jump to Step 7;

```
    (c) $\mu \leftarrow \mu + 1$.
 6. Choose $\tau \in \{0,1\}^{l_G}$ uniformly at random, add $(r, s, y_S, y_R, \tau)$ to the end
    of $Glist_2$, and compute $\hat{\tau} \leftarrow \tau$.
 7. Compute $m \leftarrow D_{\hat{\tau}}(c)$.
 8. If $Hlist_1$ is not empty, then from $\nu = 1$ to $\nu = l_{Hlist_1}$ do
    (a) take the value of $(m_\nu, y_{S_\nu}, y_{R_\nu}, K_\nu, r_\nu)$ which is the $\nu$-th record on
        $Hlist_1$;
    (b) if $(m_\nu, y_{S_\nu}, y_{R_\nu}) = (m, y_S, y_R)$ and $(g, y_R, w, K_\nu)$ is a DH tuple in
        $\langle g \rangle$, then compute $\hat{r} \leftarrow r_\nu$ and jump to Step 11;
    (c) $\nu \leftarrow \nu + 1$.
 9. If $Hlist_2$ is not empty, then from $\mu = 1$ to $\mu = l_{Hlist_2}$ do
    (a) take the value of $(r_\mu, s_\mu, m_\mu, y_{S_\mu}, y_{R_\mu}, r'_\mu)$ which is the $\mu$-th record
        on $Hlist_2$;
    (b) if $(m_\mu, y_{S_\mu}, y_{R_\mu}) = (m, y_S, y_R)$ and $(y_S g^{r_\mu})^{s_\mu} = (y_S g^r)^s$, then com-
        pute $\hat{r} \leftarrow r'_\mu$ and jump to Step 11;
    (c) $\mu \leftarrow \mu + 1$;
10. Return $Reject$;
11. Check whether $r = \hat{r}$; if it is, return $m$, otherwise return $Reject$.
```

We define a certain event F_2 to be that for some unsigncryption query, $Unigncryption_{sim}(\sigma, pk_S, pk_R) = Reject$ and $H_{sim}(m, y_S, y_R, K) = r$ where $\sigma = (c, r, s)$, $m = D_\tau(c)$, $\tau = G_{sim}(y_S, y_R, K)$, and $K = w^{x_R}$.

It is easy verify that the unsigncryption oracle and $Unsigncryption_{sim}$ has the output distribution unless F_2 occurs. Therefore, the change from Game 3 to Game 4 is a transition based on a failure event F_2. We have

$$|Pr[S_4] - Pr[S_3]| \leq Pr[F_2]. \tag{6}$$

In this proof, F_2 occurs if and only if $Unsigncryption_{sim}$ returns $Reject$ at Step 10, while $H_{sim}(m, y_S, y_R, K) = r$. According to the description of $Unsigncryption_{sim}$, in this case H_{sim} has never been run on (m, y_S, y_R, K) before this unsigncryption query and there is no record on $Hlist_2$ satisfies the output condition. According to the H_{sim} algorithm, H_{sim} will generate and return a random value at Step 3. For each unsigncryption query, the probability that r equals to that random value is $\frac{1}{q}$. Considering all j_u unsigncryption queries, the probability that $Unsigncryption_{sim}$ returns $Reject$ at Step 10, while $H_{sim}(m, y_S, y_R, K) = r$ in that case is $\frac{j_u}{q}$. Therefore, we have

$$Pr[F_2] = \frac{j_u}{q}. \tag{7}$$

Game 5 (Replace H_{sim} with H' at Stage 3): In this game $\mathcal{S}_5$ behaves mostly the same as $\mathcal{S}_4$, except that $\mathcal{S}_5$ replaces H_{sim} with another algorithm H' at Stage 3. On input $(m^*, y_{S^*}, y_{R^*}, K^*)$, H' chooses $\bar{r}^* \in Z_q$ uniformly at random and outputs $\bar{r}^*$.

Since $\bar{r}^*$ is chosen uniformly at random from Z_q, the probability that $r^* = \bar{r}^*$ is $\frac{1}{q}$. Therefore, we have

$$Pr[S_5] = \frac{1}{q}. \tag{8}$$

We define a certain event F_3 to be that at Stage 2, H_{sim} is run on input $(m^*, y_{S^*}, y_{R^*}, K^*)$ where $K^* = {y_{R^*}}^{s^*(r^*+x_{S^*})}$.

If F_3 does not occur, then from $\mathcal{A}$'s point of view, Game 5 and Game 4 proceeds identically. Therefore, the change from Game 4 to Game 5 is a transition based on a failure event F_3. Then, we have

$$|Pr[S_5] - Pr[S_4]| \leq Pr[F_3] \tag{9}$$

Game 6 (Change the Way to Generate an Input to $\mathcal{A}$): In this game $\mathcal{S}_6$ behaves mostly the same as $\mathcal{S}_5$, except that $\mathcal{S}_6$ runs in a different way at Stage 1, and at Stage 2 it calls for a restricted DDH oracle to check whether a quadruple is a DH quadruple and makes use of a random beacon R to replace the R' algorithm, where both the restricted DDH oracle and the random beacon R are provided by the rbGDL problem. In this game $\mathcal{S}_6$ (which can also be regarded as an adversary $\mathcal{A}_{rbgdl}$) prepares to take up the challenge of attacking the rbGDL problem in a group $\langle g \rangle$ with an input (g, g^a). Particularly, at Stage 1, $\mathcal{S}_6$ runs as follows:

1. Set (p, q, g) as the same as in the rbGDL problem.
2. Set $G, H, \mathcal{SKE}$ according to the Setup algorithm.
3. Set $y_A \leftarrow g^a$.
4. Give (cp, pk_A) to $\mathcal{A}$, where $cp = (p, q, g, G, H, \mathcal{SKE})$, $pk_A = y_A$.

It is obvious that the changes are only conceptual. In other words, from $\mathcal{A}$'s point of view, Game 6 and Game 5 proceeds identically. Therefore, F_3 in Game 6 and Game 5 occurs with the same probability.

Now we analyze the probability that F_3 occurs. In this proof, records on $Hlist_2$ are only be generated by $Signcryption_{sim}$ and according to the rule of the game, $\mathcal{A}$ is not allowed to make a signcryption query on $(m^*, pk_{S^*}, pk_{R^*})$ which implies there will be no such a record $(m_\mu, r_\mu, s_\mu, y_{S_\mu}, y_{R_\mu}, r'_\mu)$ on $Hlist_2$ satisfying $(m_\mu, y_{S_\mu}, y_{R_\mu}) = (m^*, y_{S^*}, y_{R^*})$. Therefore, when H_{sim} is queried on $(m^*, y_{S^*}, y_{R^*}, K^*)$, the output value will never be returned at Step 2. That is, the output value of $H_{sim}(m^*, y_{S^*}, y_{R^*}, K^*)$ is generated at Step 3 when it is first queried. In this case, according to the H_{sim} algorithm, $r^* = R(y_{R^*}, K^*)$. Therefore, $\mathcal{S}_6$ can solve the rbGDL problem by outputting (r^*, s^*, i^*) where i^* denotes R runs on input (y_{R^*}, K^*) at the i-th time. From the above analysis, we have

$$Pr[F_3] \leq Adv^{rbgdl}_{\mathcal{A}_{rbgdl}}(t_{rbgdl}, j_{rbgdl}, j_r) \tag{10}$$

where $Adv^{egdh}_{\mathcal{EGDH},\mathcal{A}_{egdh}}(t_{egdh}, j_{egdh})$ is the advantage of $\mathcal{A}_{rbgdl}$ running in time t_{rbgdl} and making at most j_{rbgdl} restricted DDH queries, and at most j_r random beacon queries. According to the execution of $\mathcal{S}_6$ in Game 6, we can compute that $t_{rbgdl} = t + t'_c$ where $t'_c = O((j_s + j_u)^2 + {j_h}^2 + {j_g}^2)$ is the simulation time of $\mathcal{S}_6$, $j_{rbgdl} = O((j_g + j_h)(j_s + j_u))$ and $j_r \leq j_h$. Therefore, t_{rbgdl}, j_{rbgdl}, and j_r are all polynomials in k.

By Lemma 3, we can construct an algorithm $\mathcal{A}_{gdl}$ to attack the GDL assumption that runs in time $t_{gdl} = 2t_{rbgdl} + O(q^2)$ and makes $j_{gdl} = 2j_{rbgdl}$ restricted DDH queries, with a success probability

$$Adv^{gdl}_{\mathcal{GDL},\mathcal{A}_{gdl}}(t_{gdl}, j_{gdl}) \geq \frac{1}{j_r}(\frac{Adv^{rbgdl}_{\mathcal{RBGDL},\mathcal{A}_{rbgdl}}(t_{rbgdl}, j_{rbgdl}, j_r)}{2} - \frac{j_r}{q})^2. \quad (11)$$

Here t_{gdl} and j_{gdl} are also polynomials in k, since t_{rbgdl} and j_{rbgdl} are polynomials in k. Recall that in Lemma 3, we have

$$Adv^{rbgdl}_{\mathcal{RBGDL},\mathcal{A}_{rbgdl}}(t_{rbgdl}, j_{rbgdl}, j_r) \geq \frac{2j_r}{q},$$

as a result, (11) can be expressed as

$$Adv^{rbgdl}_{\mathcal{RBGDL},\mathcal{A}_{rbgdl}}(t_{rbgdl}, j_{rbgdl}, j_r) \leq 2(\sqrt{j_r \cdot Adv^{gdl}_{\mathcal{GDL},\mathcal{A}_{gdl}}(t_{gdl}, j_{gdl})} + \frac{j_r}{q}). (12)$$

Combining (10) and (11), with $j_r \leq j_h$, the probability for F_3 to occur is

$$Pr[F_3] \leq \frac{2j_h}{q} + 2 \cdot \sqrt{j_h \cdot Adv^{gdl}_{\mathcal{GDL},\mathcal{A}_{gdl}}(t_{gdl}, j_{gdl})}. \quad (13)$$

Arrive at our conclusion: Combining the formulas from (1) to (9), and formula (13), we have

$$Adv^{euf-cspa}_{\mathcal{SC},\mathcal{A}}(t, j_s, j_u) \quad (14)$$
$$\leq \frac{j_s(j_g + j_h + 3j_u + 2j_s) + 2j_h + j_u + 1}{q} + 2\sqrt{j_h \cdot Adv^{gdl}_{\mathcal{GDL}\mathcal{A}_{gdl}}(t_{gdl}, j_{gdl})}. \quad (15)$$

Let $\epsilon^{gdl}_{t_{gdl},j_{gdl}}$ be the maximum of $Adv^{gdl}_{\mathcal{GDL},\mathcal{A}_{gdl}}(t_{gdl}, j_{gdl})$ over all algorithms attacking the GDL problem that runs in time t_{gdl} and makes at most j_{gdl} restricted DDH queries to a DDH oracle. From the analysis of Game 6, we get that t_{gdl} and j_{gdl} are polynomials in k. Therefore, under the GDL assumption, $\epsilon^{gdl}_{t_{gdl},j_{gdl}}$ must be negligible in k.

Taking a maximum over all EUF-CSPA adversaries with appropriate resource parameters, we get our conclusion that

$$\epsilon^{euf-cspa}_{t,j_s,j_u} \leq \frac{j_s(j_g + j_h + 3j_u + 2j_s) + 2j_h + j_u + 1}{q} + 2 \cdot \sqrt{j_h \cdot \epsilon^{gdl}_{t_{gdl},j_{gdl}}}. \quad (16)$$

Finally, we remark that the minor tweak we made is useful in Step 4(b) of $Signcryption_{sim}$, and Step 5(b) of $Unsigncryption_{sim}$. This tweak takes y_S and y_R as part of the input to the G hash function. Therefore, in these two cases we are sure that $(y_{S_\mu}, y_{R_\mu}) = (y_S, y_R)$.

5.2 Proof of Confidentiality

Theorem 2. *Let H and G be two hash functions modeled as random oracles. Then under the GDH assumption in $\langle g \rangle$ which is a subgroup of Z_p^* generated by g, and the assumption that the $\mathcal{SKE}$ is PI-SKE secure, the Zheng signcryption scheme is IND-CSPA secure.*

Specifically, let k be a security parameter of the Zheng signcryption, $\mathcal{A}$ be an IND-CSPA adversary that runs in time t, and makes at most j_s signcryption queries, j_u unsigncryption queries, j_g hash queries to G and j_h hash queries to H, where t, j_s, j_u, j_g, j_h are all polynomials in k. Then the maximum advantage $\epsilon_{t,j_s,j_u}^{ind-cspa}$ of the adversary satisfies the following condition:

$$\epsilon_{t,j_s,j_u}^{ind-cspa} \leq \epsilon_{t_{egdh},j_{egdh}}^{egdh} + \epsilon_{t_{ske}}^{pi-ske} + \frac{j_s(j_g + j_h + 6j_u + 3j_s + 2)}{q}$$

where $\epsilon_{t_{egdh},j_{egdh}}^{egdh}$, $\epsilon_{t_{ske}}^{pi-ske}$ are negligible in k for all sufficiently large k.

The proof for this theorem follows a similar path to that for Theorem 1. We leave details of the proof to a full version of this paper.

6 Relationships with Proofs by Baek, Steinfeld and Zheng

The proof of confidentiality by Baek, Steinfeld and Zheng can be naturally extended to the single key pair setting, with the exception that in the new model, more cases need to be considered. As a result, all the games in the proof should be properly described and probabilities for all the events need to be carefully analyzed by taking into account all the added cases throughout the whole proof.

The proof of unforgeability in the new model can not be naturally derived from the proof by Baek, Steinfeld and Zheng. For example, in Game 3 of the proof by Baek, Steinfeld and Zheng, when $(m^*, {y_S}^*, {y_R}^*, K^*)$ is presented to $HSim$, it is the same as that R (the random beacon) has been run on $({y_R}^*, K^*)$ which implies the rbGDL problem has been resolved. When it is extended to the single key pair setting, there should be unsigncryption queries which (according to their proof for confidentiality) may add records to $Hlist_2$. In this case $(m^*, {y_S}^*, {y_R}^*, K^*)$ is presented to $HSim$ which can be different from R being run on $({y_R}^*, K^*)$, since the result of $HSim(m^*, {y_S}^*, {y_R}^*, K^*)$ may come from $Hlist_2$. To ensure that unforgeability can be reduced to the GDL assumption under the new model, we had to resolve a number of technical issues, including the use of a random beacon, the way to add records to $Hlist_2$, and the way to simulate the unsigncryption oracle among many other minor technical issues.

Acknowledgment. We thank Joonsang Baek and Ron Steinfeld for thoroughly reading the early version of this paper and providing helpful comments. We also thank the anonymous reviewers of ACISP 2011 for their valuable comments.

References

1. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
2. Baek, J., Steinfeld, R., Zheng, Y.: Formal proofs for the security of signcryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 80–98. Springer, Heidelberg (2002)
3. Baek, J., Steinfeld, R., Zheng, Y.: Formal proofs for the security of signcryption. J. Cryptology 20(2), 203–235 (2007)
4. Bao, F., Deng, R.H., Zhu, H.: Variations of Diffie-Hellman problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (2003)
5. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the First ACM Conference on Computer and Communications Security, New York, pp. 62–73. The Association for Computing Machinery (November 1993)
6. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33, 167–226 (2003)
7. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
8. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs (2004), `http://eprint.iacr.org/2004/332`
9. Zheng, Y.: Digital signcryption or how to achieve cost (Signature & encryption) $<<$ cost(Signature) + cost(Encryption). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)
10. Zheng, Y., Imai, H.: Efficient signcryption schemes on elliptic curves. In: IFIP/SEC 1998: Proceedings of the IFIP 14th International Information Security Conference, New York, pp. 75–84. Chapman and Hall, Boca Raton (1998)