Int J Inf Secur (2003) 2: 37–44 / Digital Object Identifier (DOI) 10.1007/s10207-003-0023-7

Zheng and Seberry's public key encryption scheme revisited

Joonsang Baek[1], **Yuliang Zheng**[2]

[1] School of Network Computing, Monash University, McMahons Road, Frankston, VIC 3199, Australia
E-mail: joonsang@monash.edu.au
[2] Department of Software and Information Systems, University of North Carolina at Charlotte, Charlotte, NC 28223, USA
E-mail: yzheng@uncc.edu

Published online: 8 October 2003 – © Springer-Verlag 2003

Abstract. In this paper, we prove that a slightly modified version of Zheng and Seberry's public key encryption scheme presented at Crypto '92 is secure against adaptive chosen ciphertext attacks in the random oracle model, assuming the Gap Diffie–Hellman problem is intractable. A further contribution of this paper is to show that Soldera, Seberry, and Qu's recent security analysis of Zheng and Seberry's scheme is in fact flawed.

Keywords: Public key encryption – Chosen ciphertext security – Zheng–Seberry scheme

1 Introduction

1.1 Zheng and Seberry's encryption scheme

Since Diffie and Hellman proposed a concept of public key encryption, designing secure yet efficient public key encryption schemes has been a challenging task for many cryptographers. In addition to security and efficiency, simplicity in designing public key encryption schemes has been regarded as of particular importance due to the fact that a number of cryptographic software packages are actually implemented by application programmers who are not experts in cryptography.

For these reasons, the three earliest, surprisingly simple and efficient schemes proposed by Zheng and Seberry at Crypto '92 [12] are still worth focusing on, although more than 10 years have passed since its proposal. It is interesting to note that a number of recently proposed efficient and provably secure public key encryption schemes, including DHIES [1] and REACT [8], bear a close resemblance to one of the schemes proposed by Zheng and Seberry, which we call the "Zheng–Seberry scheme". This particular scheme is the focus of this paper. Despite its design simplicity and efficiency, a problem remaining with Zheng and Seberry's scheme is that it could not provide reductionist security: Lim and Lee [6] showed that in the Zheng–Seberry scheme the method for providing authentication capability fails, which in effect makes it impossible to prove the security in terms of indistinguishability under a chosen ciphertext attack as shown later by Soldera et al. [10]. Lim and Lee [6], however, proposed a countermeasure for the attack, and this was reflected in the new descriptions of the scheme by Zheng [11]. But no formal security analysis for this was presented in either [6] or [11].

Recently, an attempt to provide provable security to the Zheng–Seberry scheme was made by Soldera et al. [10]. They proposed a slightly different modification of the Zheng–Seberry scheme and claimed it is secure against adaptive chosen ciphertext attack in the random oracle model [3], assuming the hardness of the Decisional Diffie–Hellman (DDH) problem. However, we will demonstrate in a later section that their security proof contains serious flaws.

1.2 Our contribution and organization of the paper

The main contribution of this paper is to prove that the modification of the Zheng–Seberry scheme originally considered by Lim and Lee [6] is actually secure against an adaptive chosen ciphertext attack in the random oracle model, assuming the *Gap Diffie–Hellman* (GDH) problem [7] is intractable.

The remaining part of this paper is organized as follows. In Sect. 2, we recall some basic facts on public key encryption and chosen ciphertext security. In the same section, we review the definition of the GDH problem. In Sect. 3, we present a slight modification of the Zheng–Seberry scheme as considered in [6]. Security analysis of this scheme follows in Sect. 4. Implementation issues regarding the modified Zheng–Seberry scheme are discussed in Sect. 5. Finally, flaws in Soldera et al.'s security proof for their modification of the Zheng–Seberry scheme are discussed in detail in Sect. 6.

2 Preliminaries

2.1 Public key encryption scheme

A public key encryption scheme denoted by $\mathcal{PKE}$ consists of the following algorithms.

– A randomized common parameter generation algorithm $\mathsf{GC}(k)$ that takes a security parameter $k \in \mathbb{N}$ as input and generates a common parameter *cp* containing, say, the security parameter k, descriptions of a mathematical group, and hash functions.
– A randomized key generation algorithm $\mathsf{GK}(cp)$ that takes a common parameter *cp* as input and generates a public key *pk* and a private key *sk*, both of which contain *cp*.
– A randomized encryption algorithm $\mathsf{E}(pk, m)$ that takes the public key *pk* and a plaintext m as input and encrypts m, creating a ciphertext c.
– A deterministic decryption algorithm $\mathsf{D}(sk, c)$ that takes the private key *sk* and a ciphertext c as input and outputs a plaintext m if c is valid or a special symbol "*Reject*", otherwise.

2.2 Chosen ciphertext security for public key encryption scheme

We recall the security notion for a public key encryption scheme against an adaptive chosen ciphertext attack. Note that the notion recalled in this paper is the "indistinguishability under adaptive chosen ciphertext attack", sometimes called "IND-CCA2" [2]. For brevity, we will use the term "IND-CCA" instead.

Let $\mathsf{A}^{\mathsf{CCA}}$ denote an attacker. Note that $\mathsf{A}^{\mathsf{CCA}}$ is assumed to be a probabilistic Turing machine taking a security parameter k as input.

Now, consider the following game, which consists of several stages.

Setup: The common parameter generation algorithm on input of the security parameter k is run. On input of the resulting common parameter *cp*, the key generation algorithm is run. The resulting public key *pk* is given to $\mathsf{A}^{\mathsf{CCA}}$. (Note that *pk* contains *cp*.)
Phase 1: $\mathsf{A}^{\mathsf{CCA}}$ interacts with the decryption oracle, submitting ciphertexts and obtaining their decryptions. Note that $\mathsf{A}^{\mathsf{CCA}}$'s interaction with the decryption oracle can be adaptive.
Challenge: $\mathsf{A}^{\mathsf{CCA}}$ chooses a pair of equal-length plaintexts (m_0, m_1) and gives it to the encryption oracle. Then, the encryption oracle chooses $\beta \in \{0, 1\}$ at random and returns a target ciphertext $c^* = \mathsf{E}(pk, m_\beta)$ to $\mathsf{A}^{\mathsf{CCA}}$.
Phase 2: $\mathsf{A}^{\mathsf{CCA}}$ interacts with the decryption oracle, submitting ciphertexts and obtaining decryptions. However, $\mathsf{A}^{\mathsf{CCA}}$ is not allowed to submit the target ciphertext c^* to the decryption oracle. Note that $\mathsf{A}^{\mathsf{CCA}}$'s interaction with the decryption oracle can be adaptive.
Guess: $\mathsf{A}^{\mathsf{CCA}}$ outputs a bit $\beta' \in \{0, 1\}$.
We define the attacker $\mathsf{A}^{\mathsf{CCA}}$'s success probability by

$$\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathcal{PKE},\mathsf{A}^{\mathsf{CCA}}}(k) \stackrel{\mathrm{def}}{=} 2\Pr[\beta' = \beta] - 1$$

We denote by $\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathcal{PKE}}(t, q_D)$ the maximal success probability $\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathcal{PKE},\mathsf{A}^{\mathsf{CCA}}}(k)$ over all attackers $\mathsf{A}^{\mathsf{CCA}}$ having runing time t and making at most q_D queries to the decryption oracle. The running time t and the number of queries q_D are polynomial in the security parameter k. We say that the public key encryption scheme $\mathcal{PKE}$ is secure in the sense of IND-CCA if $\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathcal{PKE}}(t, q_D)$ is negligible in k.

2.3 The Gap Diffie–Hellman problem

We review the GDH problem, which was originally proposed by Okamoto and Pointcheval at PKC 2001 [7].

Let $\mathcal{G}$ be a group of order $q \geq 2^k$ generated by $g \in \mathcal{G}$, where q is a prime and k is a security parameter. Let $\mathsf{A}^{\mathsf{GDH}}$ denote an attacker assumed to be a probabilistic Turing machine taking the security parameter k as input.

Assume that $\mathsf{A}^{\mathsf{GDH}}$ is given a set of parameters $(\mathcal{G}, q, g, g^a, g^b)$, where a and b are uniformly chosen at random from $\mathbb{Z}_q^*$. Suppose that $\mathsf{A}^{\mathsf{GDH}}$ has access to the DDH oracle $\mathcal{O}_g^{DDH}(\cdot,\cdot,\cdot)$, which on input of $\mathsf{A}^{\mathsf{GDH}}$'s query (g^u, g^v, g^w) for $u, v, w \in \mathbb{Z}_q^*$ works as follows:

– $\mathcal{O}_g^{DDH}(g^u, g^v, g^w) = 1$ if $w = uv$. (That is, the oracle outputs 1 if (g, g^u, g^v, g^w) is a Diffie–Hellman tuple.)
– $\mathcal{O}_g^{DDH}(g^u, g^v, g^w) = 0$ otherwise.

The aim of the attacker $\mathsf{A}^{\mathsf{GDH}}$ is to compute the Diffie–Hellman key g^{ab} of g^a and g^b with the help of the above DDH oracle.

We define the attacker $\mathsf{A}^{\mathsf{GDH}}$'s success probability by

$$\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{GDH}}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{A}^{\mathsf{GDH}}(\mathcal{G}, q, g, g^a, g^b) = g^{ab}]$$

We denote by $\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G}}(t, q_{DDH})$ the maximal success probability $\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{GDH}}}(k)$ over all attackers $\mathsf{A}^{\mathsf{GDH}}$ having running time t and making to the DDH oracle at most q_{DDH} queries. The run time t and the number of queries q_{DDH} are polynomial in the security parameter k. We say that the GDH problem is intractable in the group $\mathcal{G}$ if $\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G}}(t, q_{DDH})$ is negligible in k.

3 Description of the modified Zheng–Seberry scheme

Recall that in the Zheng–Seberry scheme [12], a plaintext message m is encrypted by creating a ciphertext

$$c = (g^r, G(y^r) \oplus (m||H(m))$$

where G, H are hash functions and y is a public key such that $y = g^x$ for a private key $x \in \mathbb{Z}_q^*$. Throughout this paper, "$||$" denotes a concatenation.

The structure of the modified Zheng–Seberry scheme described in [6] is basically the same as the original one described above. The only difference is that in the modified scheme, the Diffie–Hellman key y^r as well as the message m is provided as input to the hash function H. Intuitively, this seems to prevent the known plaintext attack presented by Lim and Lee. Indeed, we show in a later section that this intuition is correct.

Now we describe the modified Zheng–Seberry scheme, which we denote by $\mathcal{MZS}$. Below, we assume that a plaintext message m is drawn from the space $\{0,1\}^{k_0}$ for $k_0 \in \mathbb{N}$.

- A randomized common parameter generation algorithm $\mathsf{GC}(k)$
  - Choose a group $\mathcal{G}$ of prime order $q \geq 2^k$.
  - Choose a generator g of $\mathcal{G}$.
  - Find a real constant $\lambda \geq 1$ such that λk is an element of $\mathbb{N}$.
    - Let $k_1 = \lambda k$.
  - Pick a hash function $H : \{0,1\}^{k_0} \times \mathcal{G} \to \{0,1\}^{k_1}$ modeled as a random oracle.
  - Pick a hash function $G : \mathcal{G} \to \{0,1\}^{k_0+k_1}$ modelled as a random oracle.
  - Output a common parameter $cp = (\mathcal{G}, g, q, G, H, k, k_0, k_1)$.
- A randomized key generation algorithm $\mathsf{GK}(cp)$
  - Pick x uniformly at random from $\mathbb{Z}_q^*$.
  - Compute $y = g^x$.
  - Output a public key $pk = (cp, y)$ and a private key $sk = (cp, x)$.
- A randomized encryption algorithm $\mathsf{E}(pk, m)$
  - Pick r uniformly at random from $\mathbb{Z}_q^*$.
  - Compute $u = g^r$ and $\kappa = y^r$.
  - Compute $\mu = G(\kappa)$ and $\sigma = H(m||\kappa)$.
  - Compute $v = \mu \oplus (m||\sigma)$.
  - Output a ciphertext $c = (u, v)$.
- A deterministic decryption algorithm $\mathsf{D}(sk, c)$
  - Parse c as (u, v).
  - Compute $\kappa = u^x$ and $\mu = G(\kappa)$.
  - Compute $t = v \oplus \mu$.
    - Let $[t]^{k_0}$ be the first k_0 bits of t, starting with the first bit.
    - Let $[t]_{k_1}$ be the remaining k_1 bits of t, starting with the $(k_0 + 1)$-th bit.
  - Compute $\sigma = H([t]^{k_0}||\kappa)$.
  - If $[t]_{k_1} = \sigma$, do the following:
    - Define m as $[t]^{k_0}$ and output m.
  - Else output "*Reject*".

4 Security analysis

In this section, we analyze the security of the $\mathcal{MZS}$ scheme. For careful analysis, we use the proof methodology introduced by Shoup [9].

In the proof, we start with the real attack game where the attacker $\mathsf{A}^{\mathsf{CCA}}$ tries to defeat the security of the Zheng–Seberry scheme in the sense of IND-CCA defined in Sect. 2.2. Then, we modify this game by changing its rules and obtain a new game. Note here that the rules of each game are to describe how variables in the view of $\mathsf{A}^{\mathsf{CCA}}$ are computed. We repeat the modification until we simulate the view of $\mathsf{A}^{\mathsf{CCA}}$ completely and obtain a game related to the ability of the attacker $\mathsf{A}^{\mathsf{GDH}}$ to solve the GDH problem defined in Sect. 2.3.

When a new game is derived from a previous one, a difference of the views of the attacker in each game might occur. This difference is measured by the technique presented in the following lemma.

Lemma 1. *Let* A_1, A_2, B_1 *and* B_2 *be events defined over some probability space.*

If $\Pr[\mathsf{A}_1 \wedge \neg\mathsf{B}_1] = \Pr[\mathsf{A}_2 \wedge \neg\mathsf{B}_2]$ *and* $\Pr[\mathsf{B}_1] = \Pr[\mathsf{B}_2] = \varepsilon$*, then we have* $|\Pr[\mathsf{A}_1] - \Pr[\mathsf{A}_2]| \leq \varepsilon$.

The proof is a straightforward calculation and can be found in [9]. Now we state and prove our main theorem.

Theorem 1. *The modified Zheng–Seberry scheme $\mathcal{MZS}$ is secure in the sense of IND-CCA in the random oracle model, assuming the GDH problem in group $\mathcal{G}$ is intractable. More precisely, we have*

$$\frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathcal{MZS}}(t, q_G, q_H, q_D) \leq \mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G}}(t', q_{DDH}) + \frac{q_D}{2^{k_1-1}}$$

where $t' = t + q_G + q_H + q_D(q_G + q_H)(T_{DDH} + O(1))$ *and* $q_{DDH} \leq q_G + q_H + q_D(q_G + q_H)$.

Here q_G, q_H, and q_D denote the number of queries made by IND-CCA attackers having running time t to the random oracles G and H and the decryption oracle, respectively. Also, q_{DDH} denotes the number of queries made by GDH attackers having running time t' to the DDH oracle $\mathcal{O}_g^{DDH}(\cdot,\cdot,\cdot)$ whose run time is denoted by T_{DDH}.

Proof. Let $\mathsf{A}^{\mathsf{CCA}}$ be an IND-CCA attacker whose running time is polynomial in a security parameter k. Also, let $\mathsf{A}^{\mathsf{GDH}}$ be an attacker trying to solve the GDH problem, given $(\mathcal{G}, q, g, g^a, g^b)$.

As mentioned earlier, we start with the following game, which is equivalent to the real attack game.

- Game G_0: This game is actually the same as the real attack game described in Sect. 2.2.
  First, we run the common parameter generation algorithm of the scheme $\mathcal{MZS}$, taking the security parameter k as input, and obtain a common parameter cp. Then, we run the key generation algorithm on input

cp and obtain a private key (cp, x) and a public key (cp, y), where $y = g^x$. The public key $pk \overset{\text{def}}{=} (cp, y)$ is given to $\mathsf{A}^{\mathsf{CCA}}$. After $\mathsf{A}^{\mathsf{CCA}}$ submits a pair of plaintexts (m_0, m_1), we create a target ciphertext $c^* = (u^*, v^*)$ as follows:

$$u^* = g^{r^*}; v^* = G(\kappa^*) \oplus (m_\beta || \sigma^*)$$

where

$$\kappa^* = y^{r^*}; \sigma^* = H(m_\beta || \kappa^*)$$

for r^* and β picked uniformly at random from $\mathbb{Z}_q^*$ and $\{0,1\}$, respectively. On input c^*, $\mathsf{A}^{\mathsf{CCA}}$ outputs $\beta' \in \{0,1\}$. We denote by S_0 the event $\beta' = \beta$ and use a similar notation S_i for all G_i.
Since this game is the same as the real attack game, we have

$$\Pr[S_0] = \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\text{IND-CCA}}_{\mathcal{MZS},\mathsf{A}^{\mathsf{CCA}}}(k)$$

– Game G_1: In this game, we modify the encryption oracle (creation of a target ciphertext) presented in the previous game. Our modification obeys the following rules:

  R1-1 First, we choose $\kappa^+ \in \mathcal{G}$, $u^+ \in \mathcal{G}$, $\mu^+ \in \{0,1\}^{k_0+k_1}$, and $\sigma^+ \in \{0,1\}^{k_1}$ uniformly at random. Then, we replace u^*, κ^*, $G(\kappa^*)$, and $H(m_\beta||\kappa^*)$ in the target ciphertext c^* by u^+, κ^+, μ^+, and σ^+, respectively. Accordingly, we replace v^* in c^* by $v^+ = \mu^+ \oplus (m_\beta||\sigma^+)$. A new target ciphertext is (u^+, v^+) and is denoted by c^*_+.
  R1-2 Whenever the random oracle G is queried at κ^+, we respond to it with μ^+.
  R1-3 Whenever the random oracle H is queried at $(m||\kappa^+)$ for some $m \in \{0,1\}^{k_0}$, we respond to it with σ^+.

The attacker $\mathsf{A}^{\mathsf{CCA}}$'s view has the same distribution in both Game G_0 and Game G_1 since we have replaced one set of random variables by another set of random variables that is different yet has the same distribution.
In this game, we assume that the decryption oracle is perfect. That is, receiving $\mathsf{A}^{\mathsf{CCA}}$'s decryption query, $c = (u, v)$, which is different from the target ciphertext, we decrypt it in the same way as we do in the real attack game. We refer to this rule as "**R1-4**".
Accordingly, we have

$$\Pr[S_1] = \Pr[S_0]$$

– Game G_2: In this game, we retain the rules **R1-1** and **R1-4**, renaming them as "**R2-1**" and "**R2-4**", respectively. However, we drop the rules **R1-2** and **R1-3**. That is, μ^+ and σ^+ are used only in the encryption oracle for producing the target ciphertext c^*_+, while in other cases when the decryption oracle queries to the random oracles G and H or $\mathsf{A}^{\mathsf{CCA}}$ directly queries to them, answers from G or H are taken. We refer to these rules regarding the random oracles G and H as "**R2-2**" and "**R2-3**", respectively.
Since we have dropped the rule **R1-2**, the input to $\mathsf{A}^{\mathsf{CCA}}$ follows a distribution that does not depend on β. Hence we get $\Pr[S_2] = 1/2$.
Also, note that Game G_1 and Game G_2 may differ if G is queried at κ^+ or H is queried at $(m||\kappa^+)$ for some $m \in \{0,1\}^{k_0}$. Let $\mathsf{AskG}_2 \vee \mathsf{AskH}_2$ denote an event in which, in Game G_2, G is queried at κ^+ or H is queried at $(m||\kappa^+)$. For notational convenience, let $\mathsf{AskKey}_2 = \mathsf{AskG}_2 \vee \mathsf{AskH}_2$. We will use an identical notation AskKey_i for all the remaining games.
Now we have

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\mathsf{AskKey}_2]$$

– Game G_3: In this game, we again modify the target ciphertext $c^*_+ = (u^+, v^+)$, where $v^+ = \mu^+ \oplus (m_\beta||\sigma^+)$ and $\mu^+ = G(\kappa^+)$, produced by the rule **R2-1** in Game G_2. We replace this rule by the following new rule **R3-1**.

  R3-1 First, we replace y the public key element given to $\mathsf{A}^{\mathsf{CCA}}$ by g^b and u^+ by g^a, where (g^a, g^b) are the parameters given to the attacker $\mathsf{A}^{\mathsf{GDH}}$. Now we define κ^+ as y^a, μ^+ as $G(y^a)$, and σ^+ as $H(m_\beta||y^a)$. That is, we use the same v^+ in c^*_+ for the second element of a new target ciphertext c^*_{DH} but we rename the values in it. Therefore, the resulting target ciphertext denoted by c^*_{DH} is (g^a, v^+), where $v^+ \overset{\text{def}}{=} G(y^a) \oplus (m_\beta||H(m_\beta||y^a))$.

However, we retain the rules **R2-2**, **R2-3**, and **R2-4** in Game G_2, renaming them as "**R3-2**", "**R3-3**", and "**R3-4**", respectively.
Thanks to the randomness of the oracle G and the uniformity of g^a in the group $\mathcal{G}$, $\mathsf{A}^{\mathsf{CCA}}$'s view has the same distribution in both Game G_2 and Game G_3. Hence we have

$$\Pr[\mathsf{AskKey}_3] = \Pr[\mathsf{AskKey}_2]$$

From now on, we deal with the decryption oracle that has been regarded as perfect up to this game.
– Game G_4: We retain all the rules **R3-1**, **R3-2**, and **R3-3**, renaming them as "**R4-1**", "**R4-2**", and "**R4-3**", respectively. But we modify the rule **R3-4** and obtain a new rule "**R4-4**" described in the following.
We make the decryption oracle reject all ciphertexts $c = (u, v)$ such that the corresponding value $(m||\kappa)$ has not been queried to the random oracle H by $\mathsf{A}^{\mathsf{CCA}}$. If c is a valid ciphertext and $G(\kappa)$ has been queried, then the rule of this game causes a difference: if c is valid, then we have $[t]_{k_1} = [v \oplus G(\kappa)]_{k_1} = H(m||\kappa)$. But we have assumed that $H(m||\kappa)$ had not been queried and hence the event $[v \oplus G(\kappa)]_{k_1} = H(m||\kappa)$ happens with probability $1/2^{k_1}$ since the output of the random oracle H is uniformly distributed in $\{0,1\}^{k_1}$.

Summing up all the decryption queries, we have

$$|\Pr[\mathsf{AskKey}_4] - \Pr[\mathsf{AskKey}_3]| \le \frac{q_D}{2^{k_1}}$$

– Game G_5: We retain all the rules **R4-1**, **R4-2**, and **R4-3**, renaming them as "**R5-1**", "**R5-2**", and "**R5-3**", respectively. But we add the following rule to **R4-4** and obtain a new rule "**R5-4**".
We make the decryption oracle reject all ciphertexts $c = (u, v)$ such that the corresponding value κ has not been queried to the random oracle G by $\mathsf{A}^{\mathsf{CCA}}$. If c is a valid ciphertext and $H(m||\kappa)$ has been queried, then the rule of this game causes a difference: if c is valid, we have $[t]_{k_1} = [v \oplus G(\kappa)]_{k_1} = H(m||\kappa)$. But we have assumed that $G(\kappa)$ had not been queried and hence the event $[v \oplus G(\kappa)]_{k_1} = H(m||\kappa)$ happens with probability $1/2^{k_1}$, assuming that $\mathsf{A}^{\mathsf{CCA}}$ correctly guesses the last k_1 bits of the output of the random oracle G.
Summing up all decryption queries, we have

$$|\Pr[\mathsf{AskKey}_5] - \Pr[\mathsf{AskKey}_4]| \le \frac{q_D}{2^{k_1}}$$

– Game G_6: Note that the cases when $H(m||\kappa)$ or $G(\kappa)$ has not been queried are excluded in this game since these cases were already dealt with in the previous game. That is, we assume that $H(m||\kappa)$ and $G(\kappa)$ have been queried at some point.
We retain all the rules **R5-1**, **R5-2**, and **R5-3**, renaming them as "**R6-1**", "**R6-2**", and "**R6-3**", respectively. But we add the following rule to **R5-4** and obtain a new rule "**R6-4**".
We replace the decryption oracle by a decryption oracle simulator, which can decrypt a submitted decryption query $c = (u, v)$ without knowing the private key. Before presenting the simulator, we define some conventions. We denote by $\mathsf{GList1}$ a list that consists of simple "query-answer" entries for the random oracle G of the form $\langle \kappa, \mu \rangle$, where $\mu = G(\kappa)$. We also denote by $\mathsf{GList2}$ a list that consists of the special "query-answer" entries for the random oracle G, which are of the form $c||(?, \mu)$. The symbol μ implicitly represents the query-answer relation $\mu = \mathrm{G}(\kappa)$, although the input κ is not explicitly stored and hence is denoted by "?". Note that new entries in the list $\mathsf{GList2}$ are added by the decryption oracle simulator. Similarly, we denote a list of all "query-answer" pairs for the random oracle H by HList. More specifically, HList consists of the pairs $\langle (m||\kappa), \sigma) \rangle$, where $\sigma = H(m||\kappa)$. Notice that all these lists are growing as $\mathsf{A}^{\mathsf{CCA}}$'s attack proceeds.
Now we describe a complete specification of the decryption oracle simulator. Note in the following that the decryption oracle simulator can be constructed using $\mathsf{A}^{\mathsf{GDH}}$'s DDH oracle $\mathcal{O}_g^{DDH}(\cdot,\cdot,\cdot)$ to check whether (u, y, κ) is a Diffie–Hellman tuple, where g is a generator of group $\mathcal{G}$, u is from the submitted ciphertext $c = (u, v)$, and y is the public key element replaced by g^b in Game G_3.

  – If there exists $\langle \kappa, \mu \rangle \in \mathsf{GList1}$ such that $\mathcal{O}_g^{DDH}(u, y, \kappa) = 1$
    - Compute $t = v \oplus \mu$.
    - If there exists $\langle (m||\kappa), \sigma \rangle \in \mathsf{HList}$ such that $m = [t]^{k_0}$ and $\sigma = [t]_{k_1}$, then output m, otherwise reject c.
  – Else if there exists $c||(?, \mu) \in \mathsf{GList2}$
    - Compute $t = v \oplus \mu$.
    - If there exists $\langle (m||\kappa), \sigma \rangle \in \mathsf{HList}$ such that $m = [t]^{k_0}$ and $\sigma = [t]_{k_1}$, then output m, otherwise reject c.
  – Else generate μ uniformly at random from $\{0, 1\}^{k_0+k_1}$
    - Compute $t = v \oplus \mu$.
    - Put $c||(?, \mu)$ into $\mathsf{GList2}$.
    - If there exists $\langle (m||\kappa), \sigma \rangle \in \mathsf{HList}$ such that $\mathcal{O}_g^{DDH}(u, y, \kappa) = 1$ and $m = [t]^{k_0}$ and $\sigma = [t]_{k_1}$, then output m, otherwise reject c.

Note that the above decryption oracle simulator perfectly simulates the real decryption oracle since $H(m||\kappa)$ and $G(\kappa)$ had been previously queried before the current game started. Thus, we get

$$\Pr[\mathsf{AskKey}_6] = \Pr[\mathsf{AskKey}_5]$$

As defined in Game G_3, AskKey_6 denotes the event that the Diffie–Hellman key $y^a (= g^{ab})$ has been queried to the random oracle G or H. At this stage, we can check which one of the queries to the random oracles G and H is a Diffie–Hellman key of g^{ab} using $\mathsf{A}^{\mathsf{GDH}}$'s DDH oracle $\mathcal{O}_g^{DDH}(\cdot,\cdot,\cdot)$. Also, note that we have used the DDH oracle to simulate the decryption oracle. That is, we have now reached the stage where we can solve the GDH problem and hence we have

$$\Pr[\mathsf{AskKey}_6] \le \mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{GDH}}}(k)$$

Now, putting all the bounds we have obtained in each game together, we have

$$\begin{aligned}\frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathcal{MZS},\mathsf{A}^{\mathsf{CCA}}}(k) &= |\Pr[S_0] - \Pr[S_2]| \\ &\le \frac{q_D}{2^{k_1}} + \frac{q_D}{2^{k_1}} + \Pr[\mathsf{AskKey}_6] \\ &\le \frac{q_D}{2^{k_1-1}} + \mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{GDH}}}(k)\end{aligned}$$

Note that since $k_1 = \lambda k \in \mathbb{N}$ for some real constant $\lambda \ge 1$ as defined in the description of $\mathcal{MZS}$, the term $\frac{q_D}{2^{k_1-1}}$ is negligible in k.

Finally, we work out the number of the calls to the DDH oracle $\mathcal{O}_g^{DDH}(\cdot,\cdot,\cdot)$ and $\mathsf{A}^{\mathsf{CCA}}$'s running time. In the worst case, all of the queries to the random oracles G and H should be checked using the oracle $\mathcal{O}_g^{DDH}(\cdot,\cdot,\cdot)$ to find the correct Diffie–Hellman key of g^a and g^b. The number of these queries are bounded by $q_G + q_H$. Also, up to $q_G + q_H$ queries are made to the oracle $\mathcal{O}_g^{DDH}(\cdot,\cdot,\cdot)$ per

Game	Target Ciphertext	Rules
Game G_0	$(u^*, G(\kappa^*) \oplus (m_\beta\|\|H(m_\beta\|\|\kappa^*)))$, where $u^* = g^{r^*}$ and $\kappa^* = y^{r^*}$ for $r^* \in_R \mathbb{Z}_q^*$ Public key: $(\mathcal{G}, g, q, y = g^x$ for $x \in_R \mathbb{Z}_q^*)$	Same as the real attack
Game G_1	$(u^+, \mu^+ \oplus (m_\beta\|\|\sigma^+))$, where $u^+ \in_R \mathcal{G}$, $\mu^+ \in_R \{0,1\}^{k_0+k_1}$ and $\sigma^+ \in_R \{0,1\}^{k_1}$ Public key: $(\mathcal{G}, g, q, y = g^x$ for $x \in_R \mathbb{Z}_q^*)$	**[R1-1]** Change u^* to u^+; κ^* to κ^+; $G(\kappa^*)$ to μ^+; $H(m_\beta\|\|\kappa^*)$ to σ^+ **[R1-2]** G is queried at κ^+: answer is μ^+ **[R1-3]** H is queried at $(m\|\|\kappa^+)$: answer is σ^+ **[R1-4]** Decryption Oracle $\rightarrow$ perfect
Game G_2	Same as above	**[R2-1]** Same as **R1-1** **[R2-2]** Answers from G are taken **[R2-3]** Answers from H are taken **[R2-4]** Decryption Oracle $\rightarrow$ perfect
Game G_3	$(g^a, \mu^+ \oplus (m_\beta\|\|\sigma^+))$, where $u^+ \in_R \mathcal{G}$, $\mu^+ \in_R \{0,1\}^{k_0+k_1}$ and $\sigma^+ \in_R \{0,1\}^{k_1}$ Public key: $(\mathcal{G}, g, q, y = g^b)$	**[R3-1]** Change u^+ to g^a; y to g^b Define κ^+ as g^{ab} **[R3-2]** Same as **R2-2** **[R3-3]** Same as **R2-3** **[R3-4]** Decryption Oracle $\rightarrow$ perfect
Game G_4	Same as above	**[R4-1]** Same as **R3-1** **[R4-2]** Same as **R3-2** **[R4-3]** Same as **R3-3** **[R4-4]** For $c = (u, v)$, $(m\|\|\kappa)$ has not been queried to H $\rightarrow$ reject c
Game G_5	Same as above	**[R5-1]** Same as **R4-1** **[R5-2]** Same as **R4-2** **[R5-3]** Same as **R4-3** **[R5-4][R4-4]** && For $c = (u, v)$, κ has not been queried to G $\rightarrow$ reject c
Game G_6	Same as above	**[R6-1]** Same as **R5-1** **[R6-2]** Same as **R5-2** **[R6-3]** Same as **R5-3** **[R6-4][R5-4]** && Decryption Oracle Simulator

Fig. 1. Summary of the proof of Theorem 1

each decryption query in the decryption oracle simulator. Therefore, the total number of calls to the DDH oracle $\mathcal{O}_g^{DDH}(\cdot,\cdot,\cdot)$ is bounded by

$$q_{DDH} \leq q_G + q_H + q_D(q_G + q_H)$$

The total running time of $\mathsf{A}^{\mathsf{GDH}}$ is bounded by

$$t' = t + q_G + q_H + q_D(q_G + q_H)(T_{DDH} + O(1))$$

where t and T_{DDH} denote the run time of $\mathsf{A}^{\mathsf{CCA}}$ and the DDH oracle, respectively.

Readers are referred to Fig. 1 for the outline of the proof of Theorem 1. Note that in the table, "&&" denotes the conditional operator "and".

5 Implementation issues

One important aspect of implementing the $\mathcal{MZS}$ scheme is how to choose a suitable group $\mathcal{G}$. The choice is quite flexible. A prime-order subgroup of the multiplicative group $\mathbb{Z}_p^*$ where p is prime would be a possible one. Also, one can choose a group of points on suitable elliptic curves instead. As is widely known, properly chosen elliptic curves can provide a highly secure public key cryptosystem with relatively small block size.

The other important issue is the implementation of the hash functions G and H, which are assumed to be random oracles [3].

Recall that the inputs of the hash function H are of the form $(m||\kappa)$, where m is a bit string of length k_0 and κ is an element of the group $\mathcal{G}$, represented as an integer. Since applications usually work with data represented as octet (byte) strings rather than bit strings, we need to convert any bit strings or integers used as variables in the scheme into octet strings. According to the IEEE P1363 standard [5], we can use the *BS2OSP* function to convert a bit string to an octet string. Also, using the *I2OSP* function, we can convert an integer to an octet string. Therefore, $(m||\kappa)$ in the scheme is actually implemented as $(BS2OSP(m)||I2OSP(\kappa))$.

Construction of the hash function H as a random oracle can be quite flexible, but we recommend the Key Derivation Function 1 (KDF1) defined in the IEEE P1363 standard.

Using the KDF1, H can be implemented as follows. $H(M) = hash(M||I2OSP(0))||\cdots||hash(M||I2OSP(l-1))$, where $hash$ is a conventional hash function such as

SHA-1 [4] with output length $hLen$, $M \overset{\text{def}}{=} (BS2OSP(m)|| I2OSP(\kappa))$ and $l = \lceil k_1/hLen \rceil$.

The hash function G can be implemented in a similar manner except for using a different conventional hash algorithm for $hash$, such as MD5.

6 Discussions on security analysis given in [10]

As mentioned in a previous section, Soldera et al. [10] proposed the following slightly different modification of the Zheng–Seberry scheme called "Secure ElGamal (SEG)".

Let $pk = (\mathcal{G}, g, q, y)$, where $y = g^x$ and $sk = (\mathcal{G}, g, q, x)$. Now a plaintext message m is encrypted by creating a ciphertext

$$c = (u, v) = (g^r, y^r \cdot (m||H(m||y^r))^2)$$

where r is randomly chosen from $\mathbb{Z}_q^*$ and H is a hash function modeled as a random oracle. Decryption of c is similar to the $\mathcal{MZS}$ scheme, so we omit it here.

First, we point out that the description of the SEG scheme given in [10] contains an error. As readers are aware, "$(m||H(m||y^r))$" in the above scheme is not an element of the group $\mathcal{G}$, hence how m is mapped to an element of $\mathcal{G}$ should have been described. In practice, this could be done using the $BS2IP$ or $OS2IP$ function recommended in the IEEE P1363 standard [5], which converts a bit string (or an octet string, respectively) to an integer.

We now look into the proof of the SEG scheme. According to [10], the SEG scheme described above is provably secure in the sense of IND-CCA in the random oracle model assuming the DDH problem is intractable. However, we show that the proof is incorrect.

Recall that in the DDH problem, given a tuple (g^a, g^b, g^c) and a generator g, the attacker is to decide whether $c = ab$. In the security proof of the SEG scheme, a variant of the DDH is used. In this variation, the attacker is to distinguish the distribution of random quadruples $D = (g_1, g_2, u_1, u_2) \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}$ from the distribution of quadruples $R = (g_1, g_2, u_1, u_2) \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}$, where g_1 and g_2 are random and $u_1 = g_1^r$ and $u_2 = g_2^r$ for random $r \in \mathbb{Z}_q^*$. If we let $g_1 = g$ and $g_2 = g^s$, then the distribution D becomes (g, g^s, g^r, g^{sr}), and hence the above "distinguishing D from R" problem becomes the standard DDH problem.

To show that the SEG scheme is secure in the sense of IND-CCA assuming that the DDH problem is hard, we should simulate the view of the IND-CCA attacker up to the point where we get the ability of the attacker solving the DDH problem to achieve its goal.

In [10], the encryption oracle of the SEG scheme is simulated as follows. We choose a private key x_R randomly from $\mathbb{Z}_q^*$ and compute $y_{sim} = g^{x_R}$. Then we give $(\mathcal{G}, g, q, y_{sim})$ as a public key to the IND-CCA attacker. On receiving a plaintext pair (m_0, m_1) from the IND-CCA attacker, we select $\beta \in \{0, 1\}$ at random and output a target ciphertext (c_1, c_2, c_3), where $c_1 = u_1$, $c_2 = u_2$, and $c_3 = c_1^{x_R} \cdot c_2 \cdot (m_\beta||H(m_\beta||c_1^{x_R}))^2$. Notice that u_1 and u_2 are from the distribution D or R.

Obviously, there is a serious problem in the above simulation of the encryption oracle. Note in the above simulation that the simulated target ciphertext has an extra component c_2. Since c_2 is an element of the group $\mathcal{G}$, the length of the simulated target ciphertext (c_1, c_2, c_3) is always 1.5 times longer than that of the target ciphertext in the real attack. Hence, with nonnegligible probability, the IND-CCA attacker can distinguish the target ciphertext in the simulation from that in the real attack by telling which one is longer.

To make the above simulation correct, we should convert (c_1, c_2, c_3) to one that has the same length of the target ciphertext in the real attack. But how this is done is not precisely mentioned anywhere in the proof except the very vague statement that "the transformation back is obvious": actually, throughout the entire proof they use the lengthened target ciphertext (c_1, c_2, c_3) as if it were a correctly simulated one.

There is another problem. It is claimed in the proof that "$c_1^{x_R} c_2$ is equivalent to the output of the actual encryption oracle". However, we show in the following that this is not the case.

In the real attack, the target ciphertext is of the form $(u, v) = (g^r, y^r \cdot (m||H(m||y^r))^2)$, where $y = g^x$; hence the component y^r yields a correct Diffie–Hellman key of $u = g^r$ and $y = g^x$ since $y^r = g^{xr}$. On the other hand, in the simulation, even if u_1 and u_2 are from the distribution D, $c_1^{x_R} c_2$ does not yield a Diffie–Hellman key of any combination of y_{sim}, c_1 and c_2: suppose that $g_1 = g$ and $g_2 = g^s$ for random $s \in \mathbb{Z}_q^*$. Since $u_1 = g_1^r$ and $u_2 = g_2^r$, we have $u_1 = g^r$ and $u_2 = g^{sr}$. Now we obtain

$$c_1^{x_R} c_2 = u_1^{x_R} u_2 = g^{r x_R} g^{sr} = g^{r x_R + sr}$$

Thus the distribution $(y_{sim}, c_1, c_2, c_1^{x_R} c_2)$ is equivalent to $(g^{x_R}, g^r, g^{sr}, g^{r x_R + sr})$. However, any three components of this distribution do not yield a Diffie–Hellman tuple.

The implication of the above problem is even more serious. Since the combination $(g^r, g^{sr}, g^{r x_R + sr})$ is not a Diffie–Hellman tuple, we cannot hope to solve the problem of distinguishing D from R, that is, the DDH problem, using the ability of the IND-CCA attacker.

7 Conclusion

In this paper, we have shown that the hardness of the GDH problem implies the chosen ciphertext security of the slightly modified version of Zheng and Seberry's public key encryption scheme. We have also discussed the current security analysis of the modified Zheng–Seberry scheme and concluded that the security proof given in [10] is incorrect.

Acknowledgements. We would like to thank the anonymous referees of IJIS for their valuable comments and suggestions.

References

1. Abdala M, Bellare M, Rogaway P (2001) The oracle Diffie–Hellman assumptions and an analysis of DHIES. In: Naccache D (ed) Progress in cryptology – CT-RSA 2001, San Francisco. Lecture notes in computer science, vol 2020, Springer, Berlin Heidelberg New York, pp 143–158
2. Bellare M, Desai A, Pointcheval D, Rogaway P (1998) Relations among notions of security for public-key encryption schemes. In: Krawczyk H (ed) Advances in cryptology – Crypto '98, Santa Barbara. Lecture notes in computer science, vol 1462, Springer, Berlin Heidelberg New York, pp 26–45
3. Bellare M, Rogaway P (1993) Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on computer and communications security, Fairfax, VA, November 1993. ACM Press, New York, pp 62–73
4. FIPS Publication 180-1 (1995) Secure hash standard
5. IEEE P1363 (2000) Standard specifications for public key cryptography
6. Lim C, Lee P (1993) Another method for attaining security against adaptively chosen ciphertext attack. In: Stinson D (ed) Advances in cryptology – Crypto '93, Santa Barbara, CA. Lecture notes in computer science, vol 773, Springer, Berlin Heidelberg New York, pp 410–434
7. Okamoto T, Pointcheval D (2001) The Gap-problems: a new class of problems for the security of cryptographic schemes. In: Kim K (ed) Public key cryptography – PKC 2001, Cheju Island, South Korea. Lecture notes in computer science, vol 1992, Springer, Berlin Heidelberg New York, pp 104–118
8. Okamoto T, Pointcheval D (2001) REACT: Rapid enhanced-security asymmetric cryptosystem transform. In: Naccache D (ed) Progress in cryptology – CT-RSA 2001, San Francisco. Lecture notes in computer science, vol 2020, Springer, Berlin Heidelberg New York, pp 159–174
9. Shoup V (2001) OAEP reconsidered. In: Kilian J (ed) Advances in cryptology – Crypto 2001, Santa Barbara, CA. Lecture notes in computer science, vol 2139, Springer, Berlin Heidelberg New York, pp 239–259
10. Soldera D, Seberry J, Qu C (2002) The analysis of Zheng–Seberry scheme. In: Batten L, Seberry J (eds) Proceedings of the Australasian conference on information security and privacy – ACISP 2002, Melbourne, Australia, July 2002. Lecture notes in computer science, vol 2384, Springer, Berlin Heidelberg New York, pp 159–168
11. Zheng Y (1994) Improved public key cryptosystems secure against chosen ciphertext attacks. Technical Note, The Centre for Computer Security Research, University of Wollongong, Sydney, Australia
12. Zheng Y, Seberry J (1992) Practical approaches to attaining security against adaptively chosen ciphertext attacks. In: Brickell E (ed) Advances in cryptology – Crypto '92, Santa Barbara, CA. Lecture notes in computer science, vol 742, Springer, Berlin Heidelberg New York, pp 292–304