Information Sciences 181 (2011) 2422–2435

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Self-generated-certificate public key encryption without pairing and its application ☆

Junzuo Lai [a,*], Weidong Kou [b], Kefei Chen [a]

[a] Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
[b] School of Computer Science and Technology, Xi Dian University, China

ARTICLE INFO

Article history:
Received 28 March 2007
Received in revised form 2 April 2009
Accepted 28 January 2011
Available online 4 February 2011

Keywords:
Certificateless public key cryptography
Self-generated-certificate public key cryptography
Ad hoc wireless networks

ABSTRACT

Recently, Liu et al. [26] discovered that Certificateless Public Key Encryption (CL-PKE) suffers the Denial-of-Decryption (DoD) attack. Based on CL-PKC, the authors introduced a new paradigm called Self-Generated-Certificate Public Key Cryptography (SGC-PKC) that captured the DoD attack and proposed the first scheme derived from a novel application of Water's Identity-Based Encryption scheme [43]. In this paper, we propose a new SGC-PKE scheme that does not depend on the bilinear pairings and hence, is more efficient and requires shorter public keys than Liu et al.'s scheme. More importantly, our scheme reaches Girault's trust level 3 [16] (cf. Girault's trust level 2 of Liu and Au's scheme), the same trust level achieved by a traditional PKI. In addition, we also discuss how our scheme can lead to a secure and self-organized key management and authentication system for ad hoc wireless networks with a function of user-controlled key renewal.

1. Introduction

Public key cryptography enables users who do not share any secret key to securely communicate over a public channel. This idea was first introduced in 1976 by Diffie and Hellman in their seminal paper [13]. Later on, Rivest, Shamir and Adlemann, proposed the well-known RSA cryptosystem [36], which realized such an idea. Since then many new cryptosystems have been proposed (see [31,42] for some relevant examples). Recently, the public-key cryptosystems based on chaotic dynamics have been widely proposed [4,12,24,35,45]. This class of cryptosystems makes use of chaotic mappings or signals. Due to the sensitivity to initial conditions, chaotic systems often have random-like behaviors, which make them popular candidates as building blocks of cryptosystems.

In traditional public key infrastructure (PKI), each user selects his own private/public key pair and publishes his public key. In order to provide an assurance about the relationship between a public key and the corresponding user, a certificate, which essentially is a signature on the user's identity and public key, is issued by a certification authority (CA). However, a PKI faces with many challenges in the practice, such as revocation, storage and distribution of certificates.

Identity-Based Public Key Cryptography (ID-PKC), first proposed by Shamir [39], alleviates the existing problems in PKI by getting rid of certificates. In ID-PKC, a user's public key is derived directly from certain aspects of its identity, such as an e-mail address. Private keys are generated for entities by a trusted third party called a Private Key Generator (PKG). In this way, the certificate is provided implicitly due to the fact that the user will not have the ability of performing any cryptographic operations, if he hasn't obtained a correct private key associated with the published identity. However, the key escrow problem exists in ID-PKC. The PKG can impersonate any user, or decrypt any ciphertext.

☆ An extended abstract of this paper was presented at Public Key Cryptography (PKC) 2007.

* Corresponding author.
E-mail addresses: laijunzuo@sjtu.edu.cn (J. Lai), kou_weidong@yahoo.com.cn (W. Kou), kfchen@sjtu.edu.cn (K. Chen).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.01.037

In order to eliminate the inherent key escrow problem of Identity-Based Cryptography (IBC) and preserve the attractive advantage of IBC, Certificateless Public Key Cryptography (CL-PKC) was introduced by Al-Riyami and Paterson [1,2]. Different from IBC, the user's public key is no longer an arbitrary string. Rather, it is similar to the public key used in the traditional PKC generated by the user. A crucial difference between them is that the public key in CL-PKC does not need to be explicitly certified as it has been generated using some partial private key obtained from the trusted authority called Key Generation Center (KGC). Note here that the KGC does not know the user's private keys since they contain secret information generated by the users themselves, thereby removing the escrow problem in IBC.

It seems that CL-PKC can solve the problem of explicit certification. Nevertheless it suffers Denial-of-Decryption (DoD) attack called by Liu et al. [26]. Suppose Alice wants to send an encrypted message to Bob. She takes Bob's public key and his identity as input to the encryption function. However, Carol, the adversary, has replaced Bob's public key by someone else's public key. Although Carol cannot decrypt the ciphertext, Bob also cannot decrypt the message while Alice is unaware of this. This is similar to Denial of Service (DoS) attack in the way that the attacker cannot gain any secret information but precluding others from getting the normal service.

Liu et al. [26] propose a new paradigm called Self-Generated-Certificate Public Key Cryptography (SGC-PKC) to defend the above attack while preserving all advantages of Certificateless Public Key Cryptography. Similar to CL-PKC, every user is given a partial secret key by the KGC and generates his own secret key and corresponding public key. In addition, he also needs to generate a certificate using his own secret key. The purpose of this self-generated certificate is similar to the one in traditional PKC. That is, to bind the identity (or personal information) and the public key together. The main difference is that, it can be verified by using the user's identity and public key only and does not require any trusted party. It is implicitly included in the user's public key. If Carol uses her public key to replace Alice's public key (or certificate), Bob can be aware of this and he may ask Alice to send him again her public key for the encryption.

Wireless ad hoc networks, as a new wireless paradigm of wireless communication, have attracted a lot of attentions recently. It is formed on-the-fly, and employs multi-hop routing to transmit information. The primary advantage of such a network is the underlying self-organizing and infrastructure-less property, which provides an extremely flexible method for establishing communications in situations where geographical or terrestrial constraints demand totally distributed networks, such as battlefields, emergency, and disaster areas. While the great flexibility of wireless ad hoc networks also brings a lot of research challenges, one of the important issues is security. Recent researches have shown that wireless ad hoc networks are highly vulnerable to various security threats due to their inherent characteristics [14,25].

The absence of a centralized control in wireless ad hoc networks makes key management more difficult. Due to their excellent performance, symmetric crypto schemes seem well-suited for wireless ad hoc networks. However, key distribution in such schemes is a major problem. This problem and the need for non-reputable communications in some applications triggered the research on public key infrastructure (PKI) and identity-based cryptographic (IBC) schemes for wireless ad hoc networks. However, these schemes have drawbacks. In Section 5, we discuss it in detail.

Related Work. Al-Riyami and Paterson [1,2] introduced Certificateless Public Key Cryptography and proposed a CL-encryption scheme and a CL-signature scheme. Some concrete efficient implementations of CL-encryption scheme were proposed in [8,40]. In addition, some generic construction were proposed in [5,29,46].

In [7], Baek et al. proposed a CL-encryption scheme without pairing, which was related to the early works on the self-certified keys [16,33]. However, their scheme can't be converted to SGC-PKE directly and only reaches Girault's trust level 2. We modify their scheme to get a new CL-encryption scheme without pairing. Our scheme can be converted to SGC-PKE directly and reaches Girault's trust level 3, which makes our scheme more appealing. Our works are related to the works on Self-Certificate-PKI [28].

On the other hand, CL-signature was first proposed in the same paper as CL-encryption in [1,2]. The security weakness of this signature scheme was pointed out by Huang et al. [20]. Later, some concrete efficient implementations of CL-signature scheme were proposed in [48,9]. A generic construction was proposed by Yum and Lee [47]. Hu et al. [18] showed that the Yum-Lee construction is insecure. Au et al. [3] suggested a malicious-but-passive-KGC attack where KGC may not generate master public/secret key pair honestly to mount the attack and then modified Hu et al.'s model [18] for capturing the attack. Recently, Huang et al. [21] revisited the security models of CL-signature schemes and proposed two concrete CL-signature schemes. They divided three kinds of adversaries against CL-signatures according to their attack power into normal adversary, strong adversary and super adversary (ordered by their attack power). They claimed that their first scheme is a short CL-signature scheme provably secure against a normal type I adversary and a super type II adversary. Unfortunately, Shim [41] showed that their short CL-signature scheme is broken by a type I adversary who can replace users' public keys and access to the signing oracle under the replaced public keys.

Recently, there have been efforts on other certificateless cryptographic primitives. Huang et al. [19] proposed a certificateless signature scheme with designated verifier. Chow et al. [10] proposed a Security-Mediated Certificateless Cryptography with revocation feature using a mediator. Duan [15] proposed a certificateless undeniable signature scheme. Long et al. [27] proposed a certificateless threshold decryption scheme and Wang et al. [44] proposed a certificateless threshold signature scheme.

Liu et al. proposed the first SGC-PKE scheme in [26], which defends the DoD attack that exists in CL-PKE. However, their scheme is based on a CL-encryption scheme and a CL-signature scheme that are using the same set of public parameters and user key generation algorithm. In addition, their scheme has long public keys and only reaches Girault's trust level 2.

Contribution. In this paper, we propose a SGC-PKE scheme without pairing and prove that it is secure in a fully adaptive adversarial model, provided that the standard Computational Diffie–Hellman (CDH) problem is hard. Compared with the first scheme, our scheme is more efficient, has shorter public keys and reaches Girault's trust level 3, which makes our scheme more practical. In addition, based on our SGC-PKE scheme, we propose a secure and self-organized key management and authentication system for ad hoc wireless networks with a function of user-controlled key renewal.

Organization. The rest of the paper is organized as follow. We give some preliminaries in Section 2. We propose a CL-encryption scheme in Section 3. The proposed SGC-PKE scheme is presented in Section 4. Based on our SGC-PKE scheme, in Section 5, our proposed key management and authentication system for ad hoc wireless networks is described in detail. Finally, concluding remarks are given in Section 6.

2. Preliminaries

In this section we first introduce our model of CL-PKE and its security definition. Next, we recall the security definition of SGC-PKE defined by Liu et al. [26]. Then we review some computational problems and the Schnorr's signature scheme. Finally, Baek, Safavi–Naini and Susilo's CL-PKE scheme [7] is reviewed.

2.1. Certificateless public key encryption

Our model is similar to that of [7] but with a crucial difference which makes our scheme reach Girault's trust level 3 and easy to be converted to a SGC-PKE scheme. Next, we formally describe our model of CL-PKE.

Definition 1 (*Certificateless public key encryption).* A generic certificateless public key encryption scheme, denoted by Π, consists of the following algorithms:

- **Setup:** is a probabilistic polynomial time (PPT) algorithm, run by a Key Generation Center (KGC), given a security parameter k as input, outputs a randomly chosen master secret **mk** and a list of public parameter **param**. We write (**mk**, **param**) = **Setup** (k).
- **UserKeyGeneration:** is a PPT algorithm, run by the user, given a list of public parameters **param** as inputs, outputs a secret key **sk** and a public key **pk**. We write (**sk**, **pk**) = **UserKeyGeneration** (**param**).
- **PartialKeyExtract:** Taking **param**, **mk**, a user's identity $\mathtt{ID}$ and **pk** received from the user, the KGC runs this PPT algorithm to generate a partial private key $D_{\mathtt{ID}}$ and a partial public key $P_{\mathtt{ID}}$. We write ($P_{\mathtt{ID}}$, $D_{\mathtt{ID}}$) = **PartialKeyExtract** (**param**, **mk**, $\mathtt{ID}$, **pk**).
- **SetPrivateKey:** Taking **param**, $D_{\mathtt{ID}}$ and **sk** as input, the user runs this PPT algorithm to generate a private key $SK_{\mathtt{ID}}$. We write $SK_{\mathtt{ID}}$ = **SetPrivateKey** (**param**, $D_{\mathtt{ID}}$, **sk**).
- **SetPublicKey:** Taking **param**, $P_{\mathtt{ID}}$ and **pk** as input, the user runs this PPT algorithm to generate a public key $PK_{\mathtt{ID}}$. We write $PK_{\mathtt{ID}}$ = **SetPublicKey** (**param**, $P_{\mathtt{ID}}$, **pk**).
- **Encrypt:** Taking a plaintext M, list of parameters **param**, a receiver's identity $\mathtt{ID}$ and $PK_{\mathtt{ID}}$ as inputs, a sender runs this PPT algorithm to create a ciphertext C. We write C = **Encrypt** (**param**, $\mathtt{ID}$, $PK_{\mathtt{ID}}$, M).
- **Decrypt:** Taking **param**, $SK_{\mathtt{ID}}$, the ciphertext C as inputs, the user as a recipient runs this deterministic algorithm to get a decryption δ, which is either a plaintext message or a "Reject" message. We write δ=**Decrypt** (**param**, $SK_{\mathtt{ID}}$, C).

For correctness, as usual we require that **Decrypt** (**param**, $SK_{\mathtt{ID}}$, C) = M whenever C = **Encrypt** (**param**, $\mathtt{ID}$, $PK_{\mathtt{ID}}$, M).

The function of **UserKeyGeneration** algorithm is the same as the **SetSecretValue** algorithm in Baek et al.'s definition. However, note that the **UserKeyGeneration** algorithm in our definition must be run prior to the **PartialKeyExtract** algorithm, compared with the **PartialKeyExtract** algorithm can be run prior to **SetSecretValue** algorithm in Baek et al.'s definition.

Security Model. According to the original scheme in [1,2], there are two types of adversaries. Type I adversary does not have the KGC's mater secret key but it can replace public keys of arbitrary identities with other public keys of its own choices. It can also obtain partial and full secret keys of arbitrary identities.

Type II adversary knows the master secret key (hence it can compute partial secret key by itself). It is still allowed to obtain full secret key for arbitrary identities but is not allowed to replace public keys at any time.

Definition 2 (*IND-CCA security).* A Certificateless Public Key Encryption scheme Π is IND-CCA secure if no PPT adversary $\mathcal{A}$ of Type I or Type II has a non-negligible advantage in the following game played against the challenger:

(1) The challenger takes a security parameter k and runs the **Setup** algorithm. It gives $\mathcal{A}$ the resulting system parameters **param**. If $\mathcal{A}$ is of Type I, the challenger keeps the master secret key **mk** to itself, otherwise, it gives **mk** to $\mathcal{A}$.
(2) $\mathcal{A}$ is given access to the following oracles:

- **Public-Key-Request-Oracle:** with a user's identity $\mathtt{ID}$, it computes (**sk**, **pk**) = **UserKeyGeneration** (**param**) and ($P_{\mathtt{ID}}$, $D_{\mathtt{ID}}$) = **PartialKeyExtract** (**param**, **mk**, $\mathtt{ID}$, **pk**). It then computes $PK_{\mathtt{ID}}$ = **SetPublicKey** (**param**, $P_{\mathtt{ID}}$, **pk**) and returns it to $\mathcal{A}$.
- **Partial-Key-Extract-Oracle:** with a user's identity $\mathtt{ID}$ and **pk**, it computes ($P_{\mathtt{ID}}$, $D_{\mathtt{ID}}$) = **PartialKeyExtract** (**param**, **mk**, $\mathtt{ID}$, **pk**) and returns it to $\mathcal{A}$. (Note that it is only useful to Type I adversary.)
- **Private-Key-Request-Oracle:** with a user's identity $\mathtt{ID}$, it computes (**sk**, **pk**) = **UserKeyGeneration** (**param**) and ($P_{\mathtt{ID}}$, $D_{\mathtt{ID}}$) = **PartialKeyExtract**(**param**, **mk**, $\mathtt{ID}$, **pk**). It then computes $SK_{\mathtt{ID}}$ = **SetPrivateKey** (**param**, $D_{\mathtt{ID}}$, **sk**) and returns it to $\mathcal{A}$. it outputs bot (denotes failure) if the user's public key has been replaced (in the case of Type I adversary.)
- **Public-Key-Replace-Oracle:** (For Type I adversary only) with identity and a valid public key, it replaces the associated user's public key with the new one.
- **Decryption-Oracle:** on input a ciphertext and an identity, returns the decrypted plaintext using the private key corresponding to the current value of the public key associated with the identity of the user.

(3) After making oracle queries a polynomial times, $\mathcal{A}$ outputs and submits two message (M_0, M_1), together with an identity $\mathtt{ID}^*$ of uncorrupted secret key to the challenger. The challenger picks a random bit $\beta \in \{0,1\}$ and computers C^*, the encryption of M_β under the current public key $PK_{\mathtt{ID}^*}$ for $\mathtt{ID}^*$. If the output of the encryption is bot, then $\mathcal{A}$ immediately losses the game. Otherwise C^* is delivered to $\mathcal{A}$.

(4) $\mathcal{A}$ makes a new sequence of queries.

(5) $\mathcal{A}$ outputs a bit β'. It wins if $\beta' = \beta$ and fulfills the following conditions:
- At any time, $\mathtt{ID}^*$ has not been submitted to **Private-Key-Request-Oracle**.
- In Step (4), C^* has not been submitted to **Decryption-Oracle** for the combination $(\mathtt{ID}^*, PK_{\mathtt{ID}^*})$ under which M_β was encrypted.
- If it is Type I, $\mathtt{ID}^*$ has not been submitted to both **Public-Key-Replace-Oracle** before Step (3) and **Partial-Key-Extract-Oracle** at some step.

Define the guessing advantage of $\mathcal{A}$ as $Adv_{\mathrm{CLE}}^{\mathrm{IND-CCA}}(\mathcal{A}) = \left|\Pr[\beta' = \beta] - \frac{1}{2}\right|$. A Type I adversary $\mathcal{A}_I$ breaks a IND-CCA secure CL-PKE scheme Π with $(t, q_{par}, q_{pub}, q_{prv}, q_D, \epsilon)$ if and only if the guessing advantage of $\mathcal{A}_I$ that accesses q_{par} times **Partial-Key-Extract-Oracle**, q_{pub} times **Public-Key-Request-Oracle**, q_{prv} times **Private-Key-Request-Oracle** and q_D times **Decryption-Oracle** is greater than ϵ within running time t. The scheme Π is said to be $(t, q_{par}, q_{pub}, q_{prv}, q_D, \epsilon)$-IND-CCA secure against Type I adversary if there is no attacker $\mathcal{A}_I$ that breaks IND-CCA secure scheme Π with $(t, q_{par}, q_{pub}, q_{prv}, q_D, \epsilon)$. There is the similar definition about Type II adversary.

2.2. Self-generated-certificate public key encryption

The definition of SGC Encryption is the same as the definition of CL-encryption given in Definition 1, except for **SetPublicKey** in which the user generates a certificate using his own secret key.

For security, in addition to IND-CCA, we require the scheme to be DoD-Free, which is formally defined as follow as a game played between the challenger and a PPT adversary (DoD adversary), which has the same power of a Type I adversary defined in CL-encryption.

Definition 3 (*DoD-Free security*). A SGC Encryption scheme is DoD-Free secure if no PPT adversary $\mathcal{A}$ has a non-negligible advantage in the following game played against the challenger:

(1) The challenger takes a security parameter k and runs the **Setup** algorithm. It gives $\mathcal{A}$ the resulting systems parameters **param**. The challenger keeps the master secret key **mk** to itself.

(2) $\mathcal{A}$ is given access to **Public-Key-Request-Oracle**, **Partial-Key-Extract-Oracle**, **Private-Key-Request-Oracle** and **Public-Key-Replace-Oracle**.

(3) After making oracle queries a polynomial times, $\mathcal{A}$ outputs a message M^*, together with an identity $\mathtt{ID}^*$ to the challenger. The challenger computes C^*, the encryption of M^* under the current public key $PK_{\mathtt{ID}^*}$ for $\mathtt{ID}^*$. If the output of the encryption is bot, then $\mathcal{A}$ immediately losses the game. Otherwise it outputs C^*.

(4) $\mathcal{A}$ wins if the following conditions are fulfilled:
- The output of the encryption in Step (3) is not bot.
- **Decrypt** (**param**, $SK_{\mathtt{ID}^*}$, C^*) = M^*.
- At any time, $\mathtt{ID}^*$ has not been submitted to **Partial-Key-Extract-Oracle**.

Define the advantage of $\mathcal{A}$ as $Adv_{\mathrm{SGCE}}^{\mathrm{DoD-Free}}(\mathcal{A}) = \Pr[\mathcal{A}wins]$.

2.3. Computational problem

We now review the standard Discrete Logarithm (DL) problem and Computational Diffie–Hellman (CDH) problem used in a large number of cryptographic schemes.

Definition 4 (*DL*). Let q be prime, Let g be a generator of $\mathbb{Z}_q$.$\mathcal{A}$ be an attacker. Define $y = g^x$ for uniformly chosen $x \in \mathbb{Z}_q$. Being given (g, y), $\mathcal{A}$ tries to find the value of x.

Definition 5 (*CDH*). Let p and q be primes such that $q|p-1$, Let g be a generator of $\mathbb{Z}_p^*$. Let $\mathcal{A}$ be an attacker. $\mathcal{A}$ tries to solve the following problem: Given (g,g^a,g^b) for uniformly chosen $a, b \in \mathbb{Z}_q^*$, compute $k = g^{ab}$.

Formally, we define $\mathcal{A}$'s advantage $Adv_{\mathbb{Z}_p^*}^{\mathrm{CDH}}(\mathcal{A})$ by $\Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}]$. $\mathcal{A}$ solves the CDH problem with if and only if the advantage of $\mathcal{A}$ is greater than ϵ within running time t. The CDH problem is said to be (t,ϵ)-intractable if there is no attacker $\mathcal{A}$ that solves the CDH problem with (t,ϵ).

2.4. The Schnorr's signature scheme

The Schnorr's signature scheme [37] is a signature scheme based on the discrete logarithm problem, which can be proved secure against the adaptively chosen message attack in the random oracle model. It consists of the following three algorithms: **Setup**, **Sign**, **Verify**.

Setup: It takes as input a security parameter k and outputs a public key $(p,q,g,H(\cdot),y)$ and a secret key x, where p and q are two large primes, $q|p-1$, g is a generator of order q in $\mathbb{Z}_p^*$, $H(\cdot)$ is a cryptographic hash function: $\{0,1\}^* \to \mathbb{Z}_q^*$, and $y = g^x \bmod p$.

Sign: To sign a message m, the user does the following performances. (1) choose a random $r \in \mathbb{Z}_q^*$, (2) compute $R = g^r \bmod p$, and (3) set the signature to be (R,σ), where $\sigma = r + xH(m\|R) \bmod q$.

Verify: To verify a signature (R,σ) for message m, the verifier does: check $g^\sigma = Ry^{H(m\|R)} \bmod p$. If it holds, the signature is valid; otherwise, the signature is invalid.

Security (Theorem 10 in) [34]: *If an existential forgery of the Schnorr signature scheme, under an adaptively chosen message attack, has non-negligible probability of success, then the discrete logarithm in subgroups can be solved in polynomial time.*

2.5. BSS's CL-PKE scheme

Because our CL-PKE scheme is based on Baek, Safavi–Naini and Susilo's CL-PKE scheme [7], we describe their scheme as follow:

- Setup (): Generate two large primes p and q such that $q|p-1$. Pick a generator g of $\mathbb{Z}_p^*$. Pick $x \in \mathbb{Z}_q^*$ uniformly at random and compute $y = g^x$. Choose hash functions $H_1 : \{0,1\}^* \times \mathbb{Z}_p^* \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \to \mathbb{Z}_q^*$ and $H_3 : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \{0,1\}^l$, where $l = l_0 + l_1 \in \mathbb{N}$. Return params = (p,q,g,y,H_1,H_2,H_3) and masterkey = (p,q,g,x,H_1,H_2,H_3).
- PartialKeyExtract (params, masterKey, ID): Pick $s \in \mathbb{Z}_q^*$ at random and compute $w = g^s$ and $t = s + xH_1(\mathrm{ID}, w)$. Return $(P_{\mathrm{ID}}, D_{\mathrm{ID}}) = (w,t)$.
- SetSecretValue (params, ID): Pick $z \in \mathbb{Z}_q^*$ at random. Return $s_{\mathrm{ID}} = z$.
- SetPrivateKey (params, $D_{\mathrm{ID}}, s_{\mathrm{ID}}$): Set $SK_{\mathrm{ID}} = (s_{\mathrm{ID}}, D_{\mathrm{ID}}) = (z,t)$. Return SK_{ID}.
- SetPublicKey (params, $P_{\mathrm{ID}}, s_{\mathrm{ID}}$, ID): Let $P_{\mathrm{ID}} = w$ and $s_{\mathrm{ID}} = z$. Compute $\mu = g^z$ and set $PK_{\mathrm{ID}} = (w,\mu)$. Return PK_{ID}.
- Encrypt (params, ID, PK_{ID}, M) where the bit-length of M is l_0: Parse PK_{ID} as (w,μ) and compute $\gamma_{\mathrm{ID}} = wy^{H_1(\mathrm{ID},w)}$. Pick $\sigma \in \{0,1\}^{l_1}$ at random, and compute $r = H_2(M,\sigma)$. Compute $C = (c_1, c_2)$ such that

$$c_1 = g^r; \quad c_2 = H_3(k_1, k_2) \oplus (M\|\sigma).$$

where $k_1 = \mu^r$ and $k_2 = \gamma_{\mathrm{ID}}^r$. Return C.
- Decrypt (params, SK_{ID}, C): Parse C as (c_1, c_2) and SK_{ID} as (z,t). Compute $M\|\sigma = H_3(c_1^z, c_1^t) \oplus c_2$. If $g^{H_2(M,\sigma)} = c_1$, return M. Else return "Reject".

3. Our CL-PKE scheme without pairing

Our scheme modifies from the first CL-PKE scheme without pairing [7]. The main difference is that, the **UserKeyGeneration** algorithm in our scheme must be run prior to the **PartialKeyExtract** algorithm. The scheme in [7] does not have this limitation. It is the crucial difference that enables us to construct SGC-PKE scheme. We also modify the **Encrypt** algorithm in [7].

3.1. Construction

Setup(k): Generate a k bit large prime q (see Chapter 4 in [31] for appropriate method) and set $p = 2q + 1$ such that $q|p-1$. We denote $\mathbb{Z}_q^*$ the multiplicative group of integers modulo q. We also denote $\mathbb{Z}_p^*$ alike. Pick a generator g of $\mathbb{Z}_p^*$ (see Chapter 4 in [31] for material algorithm). Pick $x \in \mathbb{Z}_q^*$ uniformly at random and compute $y = g^x \bmod p$. Choose hash functions $H_1 : \{0,1\}^* \times \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \to \mathbb{Z}_q^*$ and $H_3 : \mathbb{Z}_p^* \to \{0,1\}^l$, where $l = l_0 + l_1 \in \mathbb{N}$ ($\mathbb{N}$ denotes the positive integers). These hash functions can be achieved easily by collision-resistant hash function, such as SHA-1. Return **param** $= (p,q,g,y,H_1,H_2,H_3)$ and **mk** $= (p,q,g,x,H_1,H_2,H_3)$.

UserKeyGeneration (**param**): Pick $z \in \mathbb{Z}_q^*$ at random and compute $\mu = g^z \bmod p$. Return $(\mathbf{sk}, \mathbf{pk}) = (z,\mu)$.

PartialKeyExtract (**param**, **mk**, ID, **pk**): Pick $s \in \mathbb{Z}_q^*$ at random and compute $w = g^s \bmod p$ and $t = (s + xH_1(\mathrm{ID}, w, \mathbf{pk}) \bmod q) = (s + xH_1(\mathrm{ID}, w, \mu) \bmod q)$, Return $(P_{\mathrm{ID}}, D_{\mathrm{ID}}) = (w, t)$.

SetPrivateKey (**param**, D_{ID}, **sk**): Set $SK_{\mathrm{ID}} = (\mathbf{sk}, D_{\mathrm{ID}}) = (z, t)$. Return SK_{ID}.

SetPublicKey (**param**, P_{ID}, **pk**): Set $PK_{\mathrm{ID}} = (\mathbf{pk}, P_{\mathrm{ID}}) = (\mu, w)$. Return PK_{ID}.

Encrypt (**param**, ID, PK_{ID}, M) where the bit-length of M is l_0: Parse PK_{ID} as (μ, w), Pick $\sigma \in \{0,1\}^{l_1}$ at random, and compute $r = H_2(M, \sigma)$. Compute $C = (c_1, c_2)$ such that

$$c_1 = g^r \bmod p; \quad c_2 = H_3((\mu w y^{H_1(\mathrm{ID}, w, \mu)})^r \bmod p) \oplus (M \| \sigma).$$

Return C. (Note that "||" denotes "concatenation". Note also that the bit-length of (M, σ) equals to $l = l_0 + l_1$.)

Decrypt (**param**, SK_{ID}, C): Parse C as (c_1, c_2) and SK_{ID} as (z, t). Compute $M \| \sigma = H_3((c_1)^{z+t} \bmod p) \oplus c_2$. If $(g^{H_2(M,\sigma)} \bmod p) = c_1$, return M. Else return "Reject".

Due to $g^{z+t} = g^z \cdot g^t = \mu g^{s + xH_1(\mathrm{ID}, w, \mu)} = \mu w y^{H_1(\mathrm{ID}, w, \mu)} \bmod p$, it can be easily seen that the above decryption algorithm is consistent.

Note that in **PartialKeyExtract** algorithm, it includes **pk** generated by the user as input. It is the same binding technique used by the original certificateless encryption scheme [1,2] which raises our scheme to trust level 3 in the trust hierarchy of [16]. Now, with the binding technique in place, a KGC who replaces an entity's public key will be implicated in the event of a dispute: the existence of two working public keys for an identity can only result from the existence of two partial private keys binding that identity to two different public keys; only the KGC could have created these two partial private keys. Thus this binding technique makes the KGC's replacement of a public key apparent and equivalent to a CA forging a certificate in a traditional PKI.

3.2. Security analysis

The security proofs of our scheme is similar to the first CL-PKE scheme without pairing [7]. Basically, the main idea of the security proofs given in this section is to have the CDH attacker $\mathcal{B}$ simulate the "environment" of the Type I and Type II attackers $\mathcal{A}_I$ and $\mathcal{A}_{II}$ respectively until it can compute a Diffie-Hellman key g^{ab} of g^a and g^b using the ability of $\mathcal{A}_I$ and $\mathcal{A}_{II}$.

For the attacker $\mathcal{A}_I$, $\mathcal{B}$ sets g^a as a part of the challenge ciphertext and g^b as a KGC's public key. On the other hand, for the attacker $\mathcal{A}_{II}$, $\mathcal{B}$ set g^a as a part of the challenge ciphertext but uses g^b to generate a public key associated with the challenge identity.

The following two theorems show that our scheme is IND-CCA secure in the random oracle model, assuming that the CDH problem is intractable. We will give the proofs of Theorem 7 and omit the certification process of Theorem 6 due to the similarity of Theorem 7.

Theorem 6. *The CL-PKE scheme is* $(t, q_{H_1}, q_{H_2}, q_{H_3}, q_{par}, q_{pub}, q_{prv}, q_D, \epsilon)$*-IND-CCA secure against the Type I attacker* $\mathcal{A}_I$ *in the random oracle assuming the CDH problem is* (t', ε')*-intractable, where* $\epsilon' > \frac{1}{q_{H_2}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{l_1}} - \frac{q_D q_{H_2}}{2^{l_1}} - \frac{q_D}{q}\right)$ *and* $t' > t + 2(q_{par} + q_{pub} + q_{prv})t_{ex} + 2q_D q_{H_2} q_{H_3} t_{ex} + 3t_{ex}$ *where* t_{ex} *denotes the time for computing exponentiation in* $\mathbb{Z}_p^*$.

Theorem 7. *The CL-PKE scheme is* $(t, q_{H_1}, q_{H_2}, q_{H_3}, q_{pub}, q_{prv}, q_D, \epsilon)$*-IND- CCA secure against the Type II attacker* $\mathcal{A}_{II}$ *in the random oracle assuming the CDH problem is* (t', ε')*-intractable, where* $\epsilon' > \frac{1}{q_{H_2}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{l_1}} - \frac{q_D q_{H_2}}{2^{l_1}} - \frac{q_D}{q}\right)$ *and* $t' > t + 2(q_{pub} + q_{prv})$ $t_{ex} + 2q_D q_{H_2} q_{H_3} t_{ex} + 3t_{ex}$ *where* t_{ex} *denotes the time for computing exponentiation in* $\mathbb{Z}_p^*$.

Proof 1. Assume there isa TypeII adversary $\mathcal{A}_{II}$ exists. We are going to construct another PPT $\mathcal{B}$ that make uses of $\mathcal{A}_{II}$ to solve the CDH problem with probability at least ϵ' and in the time at most t'.

$\mathcal{B}$ is given (p, q, g, g^a, g^b) as an instance of the CDH problem. In order to use $\mathcal{A}_{II}$ to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and all oracles for $\mathcal{A}_{II}$.$\mathcal{B}$ does it in the following way. □

Setup. $\mathcal{B}$ picks $x \in \mathbb{Z}_q^*$ uniformly at random and computes $y = g^x$, then sets **param** $= (p, q, g, y, H_1, H_2, H_3)$ and **mk** $= (p, q, g, x, H_1, H_2, H_3)$. Finally gives $\mathcal{A}_{II}$ **param** and **mk**.

We suppose that H_1, H_2, H_3 are random oracles [6]. Adversary $\mathcal{A}_{II}$ may make queries of all random oracles at any time during its attack. $\mathcal{B}$ handles as follows:

H_1 **queries:** On receiving a query (ID, w, μ) to H_1:
(1) If $\langle(\mathrm{ID}, w, \mu), e\rangle$ exists in **H_1List**, return e as answer.
(2) Otherwise, pick $e \in \mathbb{Z}_q^*$ at random, add $\langle(\mathrm{ID}, w, \mu), e\rangle$ to **H_1List** and return e as answer.

H_2 **queries:** On receiving a query (M, σ) to H_2:
(1) If $\langle(M, \sigma), r\rangle$ exists in **H_2List**, return r as answer.
(2) Otherwise, pick $r \in \mathbb{Z}_q^*$ at random, add $\langle(M, \sigma), r\rangle$ to **H_2List** and return r as answer.

H_3 **queries:** On receiving a query k to H_3:
(1) If $\langle k,R\rangle$ exists in **H_3List**, return R as answer.
(2) Otherwise, pick $R \in \{0,1\}^l$ at random, add $\langle k,R\rangle$ to **H_3List** and return R as answer.

Phase 1. $\mathcal{A}_{II}$ can issue the following oracle queries.
Public-Key-Request: On receiving a query ID:

(1) If $\langle \mathrm{ID},(\mu,w),coin\rangle$ exists in **PublicKeyList**, return $PK_{\mathrm{ID}} = (\mu,w)$ as answer.
(2) Otherwise, pick $coin \in \{0,1\}$ at random, so that $\Pr[coin = 0] = \delta$. (δ will be determined later.)
(3) If $coin = 0$, pick $z,s \in \mathbb{Z}_q^*$ at random and compute $\mu = g^z$, $w = g^s$, and $t = s + xH_1(\mathrm{ID},w,\mu)$; add $\langle \mathrm{ID},(z,t)\rangle$ to **PrivateKeyList** and $\langle \mathrm{ID},(\mu,w),coin\rangle$ to **PublicKeyList**; return $PK_{\mathrm{ID}} = (\mu,w)$ as a answer.
(4) Otherwise (if $coin = 1$), pick $z,s \in \mathbb{Z}_q^*$ at random and compute $\mu = g^z$, $w = (g^b)^s$; add $\langle \mathrm{ID},(z,*)\rangle$ ($*$ denotes that the item can be any value) to **PrivateKeyList** and $\langle \mathrm{ID},(\mu,w),coin\rangle$ to **PublicKeyList**; return $PK_{\mathrm{ID}} = (\mu,w)$ as a answer.

Private-Key-Request: On receiving a query ID:

(1) Run **Public-Key-Request** on ID to get a tuple $\langle \mathrm{ID},(\mu,w),coin\rangle \in$ **PublicKeyList**.
(2) If $coin = 0$, search **PrivateKeyList** for a tuple $\langle \mathrm{ID},(z,t)\rangle$ and return $SK_{\mathrm{ID}} = (z,t)$ as answer.
(3) Otherwise, return "Abort" and terminate.

Decryption queries: On receiving a query $(\mathrm{ID},PK_{\mathrm{ID}},C)$, where $C = (c_1,c_2)$ and $PK_{\mathrm{ID}} = (\mu,w)$:

(1) Search **PublicKeyList** for tuple $\langle \mathrm{ID},(\mu,w),coin\rangle$. If $coin = 0$, search **PrivateKeyList** for a tuple $\langle \mathrm{ID},(z,t)\rangle$. (Note that $\langle \mathrm{ID},(\mu,w),coin\rangle$ must exist in **PublicKeyList** and when $coin = 0$, $\langle \mathrm{ID},(z,t)\rangle$ exist in **PrivateKeyList**.) Then set $SK_{\mathrm{ID}} = (z,t)$ and run **Decrypt** (**param**,SK_{ID},C). Finally, return the result of **Decrypt** algorithm.
(2) Otherwise (if $coin = 1$), run **H_1 query** to get a tuple $\langle(\mathrm{ID},w,\mu),e\rangle$. If there exist $\langle(M,\sigma),r\rangle \in$ **H_2List** and $\langle k,R\rangle \in$ **H_3List** such that $c_1 = g^r, c_2 = R \oplus (M\|\sigma)$ and $k = (\mu w y^e)^r$, return M and "Reject" otherwise.

Challenge. $\mathcal{A}_{II}$ then output two message (M_0,M_1) and a challenge identity ID^*. $\mathcal{B}$ run **Public-Key-Request** taking ID^* as input to get a tuple $\langle \mathrm{ID}^*,(\mu^*,w^*),coin\rangle \in$ **PublicKeyList**.

(1) If $coin = 0$ return "Abort" and terminate.
(2) Otherwise, do the following:
  (a) Search **PrivateKeyList** for a tuple $\langle \mathrm{ID}^*,(z^*,*),s^*\rangle$.
  (b) Pick $\sigma^* \in \{0,1\}^{l_1}, c_2^* \in \{0,1\}^l$ and $\beta \in \{0,1\}$ at random.
  (c) Set $c_1^* = g^a$ and $e^* = H_1(\mathrm{ID}^*,w^*,\mu^*)$.
  (d) Define $a = H_2(M_\beta,\sigma^*)$ and $H_3((\mu^* w^* y^{e^*})^a)$. (Note that $\mathcal{B}$ does not know "a", $(\mu^* w^* y^{e^*})^a = (g^a)^{z^*} \cdot (g^{ab})^{s^*} \cdot (g^a)^{xe^*}$.
(3) Return $C^* = (c_1^*, c_2^*)$ as a target ciphertext.

Phase 2. $\mathcal{B}$ repeats the same method it used in Phase 1.

Guess. Finally, $\mathcal{A}_{II}$ output a guess β'. Now $\mathcal{B}$ choose a tuple $\langle k,R\rangle$ form the **H_3List** and outputs $\left(\frac{k}{(g^a)^{z^*}\cdot(g^a)^{xe^*}}\right)^{1/s^*}$ as the solution the the CDH problem.

Analysis: From the construction of H_1, it is clear that the simulation of H_1 is perfect. As long as $\mathcal{A}_{II}$ does not query (M_β,σ^*) to H_2 nor $(\mu^* w^* y^{e^*})^a$ to H_3, the simulations of H_2 and H_3 are perfect. By AskH_3^* we denote the event that $(\mu^* w^* y^{e^*})^a$ has not been queried to H_3. Also, by AskH_2^* we denote the event that (M_β,σ^*) has been queried to H_2. If happens then $\mathcal{B}$ will be able to solve the CDH problem by choosing a tuple $\langle k,R\rangle$ form the **H_3List** and computing $\left(\frac{k}{(g^a)^{z^*}\cdot(g^a)^{xe^*}}\right)^{1/s^*}$ with the probability at least $\frac{1}{q_{H_3}}$. Hence we have $\epsilon' \geqslant \frac{1}{q_{H_3}}\Pr[\mathrm{AskH}_3^*]$.

It is easy to notice that if $\mathcal{B}$ does not abort, the simulations of **Public-Key-Request**, **Private-Key-Request** and the simulated target ciphertext is identically distributed as the real one from the construction.

Now, we evaluate the simulation of the decryption oracle. If a public key PK_{ID} has been produced under $coin = 0$, the simulation is perfect as $\mathcal{B}$ knows the private key SK_{ID} corresponding to PK_{ID}. Otherwise, simulation errors may occur while $\mathcal{B}$ running the decryption oracle simulator specified above. Let **DecErr** be this event. We compute the probability of this event: Suppose that $(\mathrm{ID},PK_{\mathrm{ID}},C)$, where $C = (c_1,c_2)$ and $PK_{\mathrm{ID}} = (\mu,w)$, has been issued as a *valid* decryption query. Even if C is valid, there is a possibility that C can be produced without querying $(\mu w y^e)^r$ to H_3, where $e = H_1(\mathrm{ID},w,\mu)$ and $r = H_2(M,\sigma)$. Let **Valid** be an event that C is valid. Let **AskH$_3$** and **AskH$_2$** respectively be events that $(\mu w y^e)^r$ has been queried to H_3 and (M,σ) has been queried to H_2 with respect to $C = (c_1,c_2) = (g^r, H_3((\mu w y^{H_1(\mathrm{ID},w,\mu)})^r) \oplus (M\|\sigma))$ and $PK_{\mathrm{ID}} = (\mu,w)$, where $r = H_2(M,\sigma)$ and $e = H_1(\mathrm{ID},w,\mu)$. We then have $\Pr[\textbf{DecErr}] = q_D\Pr[\textbf{Valid}|\neg\textbf{AskH}_3]$. But

$$\Pr[\text{Valid}|\neg\text{AskH}_3] \leqslant \Pr[\text{Valid} \wedge \text{AskH}_2|\neg\text{AskH}_3] + \Pr[\text{Valid} \wedge \neg\text{AskH}_2|\neg\text{AskH}_3]$$
$$\leqslant \Pr[\text{AskH}_2|\neg\text{AskH}_3] + \Pr[\text{Valid}|\neg\text{AskH}_2 \wedge \neg\text{AskH}_3] \leqslant \frac{q_{H_2}}{2^{l_1}} + \frac{1}{q}.$$

So, $\Pr[\text{DecErr}] \leqslant \frac{q_D q_{H_2}}{2^{l_1}} + \frac{q_D}{q}$.

Now, the event $(\text{AskH}_3^* \vee (\text{AskH}_2^*|\neg\text{AskH}_3^*) \vee \text{DecErr})|\neg\text{Abort}$ denoted by **Good**, where **Abort** denotes an event that $\mathcal{B}$ aborts during the simulation. The probability $\neg$**Abort** that happens is given by $\delta^{q_{prv}}(1-\delta)$ which is maximized at $\delta = 1 - 1/(q_{prv}-1)$. Hence we have $\Pr[\neg\text{Abort}] \leqslant \frac{1}{e(q_{prv}+1)}$, where e denotes the base of the natural logarithm.

If **Good** does not happen, it is clear that $\mathcal{A}_{II}$ does not gain any advantage greater than 1/2 to guess β due to the randomness of the output of the random oracle H_3. Namely, we have $\Pr[\beta' = \beta|\neg\text{Good}] \leqslant \frac{1}{2}$.

By definition of ϵ, we then have

$$\epsilon < \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| = \left|\Pr[\beta' = \beta|\neg\text{Good}]\Pr[\neg\text{Good}] + \Pr[\beta' = \beta|\text{Good}]\Pr[\text{Good}] - \frac{1}{2}\right| \leqslant \left|\frac{1}{2}\Pr[\neg\text{Good}] + \Pr[\text{Good}] - \frac{1}{2}\right|$$
$$\leqslant \frac{1}{2}\Pr[\text{Good}] \leqslant \frac{1}{2\Pr[\neg\text{Abort}]}\left(\Pr[\text{AskH}_3^*] + \Pr[\text{AskH}_2^*|\neg\text{AskH}_3^*] + \Pr[\text{DecErr}]\right) \leqslant \frac{e(q_{prv}+1)}{2}\left(q_{H_3}\epsilon' + \frac{q_{H_2}}{2^{l_1}} + \frac{q_D q_{H_2}}{2^{l_1}} + \frac{q_D}{q}\right).$$

Consequently, we obtain $\epsilon' > \frac{1}{q_{H_2}}\left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{l_1}} - \frac{q_D q_{H_2}}{2^{l_1}} - \frac{q_D}{q}\right)$. The running time of the CDH attacker $\mathcal{B}$ is $t' > t + 2(q_{pub} + q_{prv})\, t_{ex} + 2q_D q_{H_2} q_{H_3} t_{ex} + 3t_{ex}$ where t_{ex} denotes the time for computing exponentiation in $\mathbb{Z}_p^*$. This completes the proof of Theorem.

4. Our SGC-PKE scheme without pairing

4.1. Construction

We give our Self-Generated-Certificate (SGC) encryption scheme without pairing based on the above certificateless encryption scheme. In fact, our SGC-PKE scheme is obtained from signing the public key of the user in our CL-PKE scheme using his (her) private key. The most algorithms are the same as the algorithms of our CL-PKE scheme, except for **SetPublicKey** and **Encrypt** algorithm.

In order to distinguish the algorithm of CL-encryption, we will add the prefix "**CL.**" to the corresponding algorithms. For example, we use "**CL.Setup**" to denote the encryption algorithm of the CL-encryption scheme. The proposed SGC-encryption scheme is described as follow:

Setup: Same as **CL.Setup**, outputs parameters **param** = $(p,q,g,y = g^x \bmod p, H_1, H_2, H_3)$ and master secret key **mk** = (p,q,g,x, H_1, H_2, H_3).

UserKeyGeneration: Same as **CL.UserKeyGeneration**, outputs

$$(\mathsf{sk}, \mathsf{pk}) = (z, \mu = g^z \bmod p).$$

PartialKeyExtract: We modify **CL.PartialKeyExtract** slightly. Taking **param**, **mk**, ID and **pk** as input, it outputs

$$(P_{\text{ID}}, D_{\text{ID}}) = (w = g^s \bmod p, t = (s + xH_1(\text{ID}, w * \mathsf{pk})) \bmod q = (s + xH_1(\text{ID}, w\mu)) \bmod q).$$

In order to make this changes, it must modify the domain of hash function $H_1 : \{0,1\}^* \times \mathbb{Z}_p^* \to \mathbb{Z}_q^*$.

SetPrivateKey: Same as **CL.SetPrivateKey**, outputs

$$SK_{\text{ID}} = (\mathsf{sk} + D_{\text{ID}}) \bmod q = (z + t) \bmod q.$$

SetPublicKey: Except for taking **param**, P_{ID} and **pk** as input, it includes ID and SK_{ID} as inputs. Chooses a new hash function $H : \{0,1\}^* \times \mathbb{Z}_p^* \times \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{Z}_q^*$, then computes $PK_{\text{ID}}^1 = \mathsf{pk} * P_{\text{ID}} = \mu w \bmod p$ and

$$PK_{\text{ID}}^2 = \mathsf{pk} * P_{\text{ID}} * y^{H_1(\text{ID}, PK_{\text{ID}})^1} = \mu w y^{H_1(\text{ID}, \mu w)} = g^{z+t} = g^{SK_{\text{ID}}} \bmod p.$$

Next, it does the following performances to sign the user's identity ID and PK_{ID}^1, PK_{ID}^2 using the user's private key SK_{ID} and Schnorr's signature scheme [37].

(1) choose a random $r \in \mathbb{Z}_q^*$,
(2) compute $R = g^r \bmod p$, and
(3) set the signature to be (R, σ), where $\sigma = r + SK_{ID} * H(ID, PK_{ID}^1, PK_{ID}^2, R) \bmod q$.

Finally, returns $PK_{\text{ID}} = (PK_{\text{ID}}^1, PK_{\text{ID}}^2, (R, \sigma))$.

Encrypt: Parses PK_{ID} as $(PK_{\text{ID}}^1, PK_{\text{ID}}^2, (R, \sigma))$. If

$$PK_{\text{ID}}^2 \neq PK_{\text{ID}}^1 * y^{H_1(\text{ID}, PK_{\text{ID}})^1} \bmod p \text{ or } g^\sigma \neq R * (PK_{\text{ID}}^2)^{H(\text{ID}, PK_{\text{ID}}^1, PK_{\text{ID}}^2, R)} \bmod p,$$

it returns $\perp$, else outputs **CL.Encrypt**(**param**, ID, PK_{ID}, M).

Decrypt: Same as **CL.Decrypt**, outputs a plaintext M for a valid ciphertext C, or "Reject" otherwise.

Note that nobody can know the user's private key except himself, because the private key SK_{ID} includes **sk** generated by the user. In addition, compared with the requirement of a confidential and authentic channel between the KGC and user in most IBC schemes for the secure distribution of the private keys, an authentic channel is sufficient in our scheme.

In fact, the key pair $(PK^2_{\mathrm{ID}} = g^{SK_{\mathrm{ID}}}, SK_{\mathrm{ID}})$ of the user is the same as any DL-based public key cryptosystem. So, we can use any secure DL-based signature scheme (such as Schnorr's signature scheme [37]) as our signature scheme. All these attributes of our SGC-PKE scheme enable us to design a secure and self-organized key management and authentication system for ad hoc wireless networks with a function of user-controlled key renewal.

4.2. Security analysis

The IND-CCA security depends on our CL-encryption scheme (defined in Section 3). In addition to IND-CCA, we require the scheme to be DoD-Free. Here, we analyze the DoD-Free security.

Theorem 8. *The SGC-encryption scheme proposed in this section in secure against DoD adversary, assuming that the Schnorr's signature scheme is secure against the adaptively chosen message attack in the random oracle model [34].*

Proof 2. Assume there is a DoD adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to break the Schnorr signature scheme.$\mathcal{B}$ is now the Schnorr's signature adversary. Note that in fact, the **PartialKeyExtract** algorithm in our SGC-encryption scheme signs the user's identity $\mathtt{ID}$ using the Schnorr's signature scheme. So using his signing-oracle, $\mathcal{B}$ can answer all oracle queries for $\mathcal{A}$. After a polynomial number of oracle queries, $\mathcal{A}$ outputs a message M^* and an identity $\mathtt{ID}^*$. $\mathcal{A}$ wins if the following conditions fulfill: □

(1) The public key $PK_{\mathrm{ID}^*} = (PK^1_{\mathrm{ID}^*}, PK^2_{\mathrm{ID}^*}, (R^*, \sigma^*))$ of $\mathtt{ID}^*$ is valid.
(2) $\mathrm{Decrypt}(\mathrm{param}, SK_{\mathrm{ID}^*}, C^*) \neq M^*$ where C^*= **Encrypt** (**param**, $\mathtt{ID}^*, PK_{\mathrm{ID}^*}, M^*$).
(3) $\mathcal{A}$ does not query the **Partial-Key-Extract-Oracle** for $\mathtt{ID}^*$.

If the public key of $\mathtt{ID}^*$ has not been replaced, due to correctness we always have $\mathrm{Decrypt}(\mathrm{param}, SK_{\mathrm{ID}^*}, C^*) = M^*$. Condition (2) implies the public key of $\mathtt{ID}^*$ has been replaced. Together with condition (1) and (3), it implies that $\langle \tilde{m} = (\mathtt{ID}^*, PK^1_{\mathrm{ID}^*}, PK_{\mathrm{ID}^*})^2,\ \tilde{\sigma} = (R^*, \sigma^*)\rangle$ is a successful forgery for $\mathtt{ID}^*$. $\mathcal{B}$ outputs it.

4.3. Comparison to previous work

Our scheme is the second SGC-encryption scheme. In this section, we compare our scheme with the first scheme in [26].

(1) Our scheme has shorter public keys due to their scheme based on the Water's Identity-Based Encryption scheme [43].
(2) Our scheme is more efficient due to our scheme without pairing computation. In spite of the recent advances in implementation technique, the pairing computation is still considered as expensive compared with "standard" operations such as modular exponentiations in finite fields.
(3) Our scheme reaches Girault's trust level 3 (same as the traditional PKI), but their scheme only reaches Girault's trust level 2 (a cheating KGC could replace an entity's public key bye one for which it knows the secret value without fear of being identified).
(4) Their scheme is $\mathtt{IND\text{-}CCA}^-$ (the challenger is forced to decrypt ciphertexts for which the public key has been replaced) and DoD-Free secure in the standard model. Our scheme is IND-CCA and DoD-Free secure in the random oracle model.

5. Application to ad hoc networks

In this section, we first discuss some of the proposed key management and authentication schemes for ad hoc networks, including public key solutions and identity-based solutions. Next, based on our SGC-PKE scheme, we propose our solution.

5.1. Public key solutions

In order to bind public keys to an identity, a certification authority (CA) is needed to issue, distribute, revoke, and renew public key certificates. Due to the absence of a central CA in wireless ad hoc networks, many proposed solutions emulate a CA in the network. We refer to such a CA as internal CA. Internal CAs can be implemented by (k,n)-threshold schemes to distribute the power and tasks of the CA to a group of special nodes [49] or all network nodes [30]. The weakness of the former solution is that it requires an administrative infrastructure available to distribute the shares to the special nodes. The scheme is further complicated by the normal nodes' need to locate the server nodes. Keeping the n special nodes available when needed makes the system maintenance difficult. In another approach, every network node acts as an internal CA, i.e. nodes

issue and distribute their own public keys and sign others in a PGP manner [22]. The performance of this scheme highly depends on the length of the trusted path and is generally hard to predict.

In another class of certificate-based PKI schemes for wireless ad hoc networks, network devices are initialized by an external off-line KGC. Proposed revocation schemes for PKIs without internal CA introduce so-called 'accusation' schemes [11,30]. Here, each node can accuse other nodes to be malicious or compromised. If the number of accusations is greater than a certain threshold δ, the certificate is considered to be revoked. All accusations need to be frequently broadcasted in order to inform all nodes about recent revocations and changes.

5.2. Identity-based solutions

Recently, identity-based cryptographic (IBC) schemes have been considered for securing ad hoc networks [14,23]. Both papers suggest emulating an internal KGC using (k,n)-threshold schemes, as previously introduced for internal CAs in PKIs. Both schemes do not introduce key revocation and key renewing algorithms for their schemes. Based on their new accusation scheme, Hoeper and Gong [17] proposed the first key revocation and key renewal mechanisms for IBC schemes that are especially designed for wireless ad hoc networks. They also used the threshold cryptography to obtain fully self-organized key management scheme. Due to the inherent property of IBC, all these schemes suffer key escrow problem. Even though adopting the threshold cryptography, it can only lighten the key escrow problem, and a coalition of adequate nodes can breach these schemes thoroughly. In addition, many identity-based cryptographic schemes need pairing computation. In spite of the recent advances in implementation technique, the pairing computation is still considered as expensive compared with "standard" operations such as modular exponentiations in finite fields.

5.3. Our solution

Based on our SGC-PKE scheme, we propose our solution. We first discuss our assumptions about the network, and then describe the key generation, key renewal, key revocation and authentication mechanisms in detail.

Consider an ad hoc network with n nodes in the initial phase. We assume that each mobile node has an identity, which is unique and unchangeable during its lifetime in the ad hoc network. We also assume that each mobile node has a mechanism to discover its one-hop neighborhood and to get the identities of other nodes in the network. The network has a public/private key pair, called master key pair, which is used to provide key generation service to all the nodes in the network. The master key pair is generated in such a manner that the master public key is well known to all the nodes in the network, and the master private key is shared by all of them in a (k,n) threshold fashion.

5.3.1. Master key generation

The master key pair is computed collaboratively by the initial network nodes without constructing the master private key at any single node. The scheme we used [32] is an extension to Shamir's secret sharing [38] without the support of a trust authority. In the scheme, each node $\mathtt{ID}_i$ does the following steps.

(1) Generate two large primes p and q such that $q|p-1$. Pick a generator g of $\mathbb{Z}_p^*$.
(2) Choose a random number r_i over $\mathbb{Z}_q$ and a random polynomial $f_i(x)$ over $\mathbb{Z}_q$ of degree $k-1$, such that $f_i(0) = r_i$.
(3) Broadcast $y_i = g^{f_i(0)} = g^{r_i}$.
(4) Compute the sub-share for node $\mathtt{ID}_j$ as $x_{ij} = f_i(j)$ for $j = 1,\ldots,n$ and send x_{ij} to $\mathtt{ID}_j$. After sending the $n-1$ sub-shares, node $\mathtt{ID}_j$ can compute its share of master private key as

$$x_j = \sum_{i=1}^{n} x_{ij} = \sum_{i=1}^{n} f_i(j).$$

That is, the master key share of node $\mathtt{ID}_j$ is combined by the sub-shares from all the nodes, and each of them contributes one piece of that information.

It is easy to see that the jointly generated master private key

$$x = \sum_{i=1}^{n} r_i = \sum_{i=1}^{n} f_i(0) \bmod q,$$

which is not explicitly computed by any node. However, any coalition of k shareholders can jointly recover the secret as in basic secret sharing using $\sum_{i=1}^{k} x_i \lambda_i^n \bmod q$, where λ_i^n is the Lagrange coefficient. The corresponding master public key is

$$y = \prod_{i=1}^{n} y_i = \prod_{i=1}^{n} g^{f_i(0)} = \prod_{i=1}^{n} g^{r_i} = g^{\sum_{i=1}^{n} r_i} = g^x \bmod p.$$

We also deploy the verifiable secret sharing to detect the invalid share that some shareholders generate to prevent the reconstruction of the secret key. In fact, the master private key is shared by all of them in a (k,n) threshold fashion. Each of them

holds a unique secret share of the master private key x, and no one is able to reconstruct the master private key based on its own information. Any k nodes among them can reconstruct the mater private key jointly, whereas it is infeasible for at most $k-1$ nodes to do so, even by collusion.

5.3.2. Distributed private key generation

The way to obtain the private key is to contact at least k neighbor nodes, present the identity and request private key generation (PKG) service. The node that holds the master key share can be the PKG service node. In our scheme, all the network nodes share the master private key, thus each of them can be the PKG service node. To get his private key, node $\mathtt{ID}_i$ and his k neighbor nodes do the following steps.

(1) The node $\mathtt{ID}_i$ picks $z \in \mathbb{Z}_q^*$ at random and computes $\mu = g^z$. He also sets $(usk, upk) = (z, \mu)$.
(2) The node $\mathtt{ID}_i$ broadcasts his network identifier (NID) to his k neighbor nodes. Let Φ be the set of his k neighbor nodes.
(3) The node $\mathtt{ID}_{e_j} \in \Phi$ for $j = 1,\ldots,k$ picks $s_{e_j} \in \mathbb{Z}_q^*$ at random and computes $w_{e_j} = g^{s_{e_j}}$, then sends w_{e_j} to node $\mathtt{ID}_i$.
(4) The node $\mathtt{ID}_i$ computes $w = \prod_{j=1}^{k} w_{e_j}^{\lambda_j^{\Phi}} \ mod\ p$, where λ_j^{Φ} is the Lagrange coefficient. Then, sets

$$ppk = w, \quad pk^{(1)} = upk \cdot ppk = \mu w$$

and broadcasts $pk^{(1)}$ to his k neighbor nodes. If we set $s = \sum_{j=1}^{k} s_{e_j}\lambda_j^{\Phi} \ mod\ q$,

$$w = \prod_{j=1}^{k} w_{e_j}^{\lambda_j^{\Phi}} = \prod_{j=1}^{k} g^{s_{e_j}\lambda_j^{\Phi}} = g^{\sum_{j=1}^{k} s_{e_j}\lambda_j^{\Phi}} = g^{s}.$$

(5) The node $\mathtt{ID}_{e_j} \in \Phi$ for $j = 1,\ldots,k$ computes

$$t_{e_j} = s_{e_j} + x_{e_j} H_1(\mathrm{NID}, pk^{(1)}) = s_{e_j} + x_{e_j} H_1(\mathrm{NID}, w\mu),$$

where x_{e_j} is the share of master private key of node $\mathtt{ID}_{e_j}$, then sends t_{e_j} to node $\mathtt{ID}_i$.
(6) The node $\mathtt{ID}_i$ computes

$$psk = \sum_{j=1}^{k} t_{e_j}\lambda_j^{\Phi} = \sum_{j=1}^{k} s_{e_j}\lambda_j^{\Phi} + H_1(\mathrm{NID}, w\mu) \cdot \sum_{j=1}^{k} x_{e_j}\lambda_j^{\Phi} = s + xH_1(\mathrm{NID}, w\mu).$$

(7) The node $\mathtt{ID}_i$ sets his private key $sk = usk + psk = z + s + xH_1(\mathrm{NID}, w\mu)$.
(8) The node $\mathtt{ID}_i$ computes

$$pk^{(2)} = g^{sk} = g^{z+s+xH_1(\mathrm{NID},w\mu)} = \mu w y^{H_1(\mathrm{NID},w\mu)} = pk^{(1)} y^{H_1(\mathrm{NID},pk^{(1)})}.$$

Next, it does the following performances to sign his network identifier (NID) and $pk^{(1)}$, $pk^{(2)}$ using his private key sk and Schnorr's signature scheme.

(1) choose a random $r \in \mathbb{Z}_q^*$,
(2) compute $R = g^r \bmod p$, and
(3) set the signature to be (R, σ), where $\sigma = r + sk \cdot H(\mathrm{NID}, pk^{(1)}, pk^{(2)}, R)$.

Finally, sets his public key $pk = (pk^{(1)}, pk^{(2)}, (R, \sigma))$ and broadcasts it to all nodes in the network.

It is easy to see that our private key generation method is the same as the one in our SGC-PKE scheme. So, any other node does not know the private key of node $\mathtt{ID}_i$, even though other $n-1$ nodes collude. In addition, in order to verify whether $pk = (pk^{(1)}, pk^{(2)}, (R, \sigma))$ is the public key of node $\mathtt{ID}_i$, any other node can test if

$$pk^{(2)} = pk^{(1)} y^{H_1(\mathrm{NID},pk^{(1)})} \quad \text{and } g^{\sigma} = R(pk^{(2)})^{H(\mathrm{NID},pk^{(1)},pk^{(2)},R)}.$$

Note that, because a coalition of k or more nodes can reconstruct the master private key x, they can construct well-formed public key for any node. However, anyone can find the conspiracy easily, due to the existence of two working public keys for a node. This case is equivalent to a CA forging a certificate in a traditional PKI: the existence of two valid certificates would surely implicate the CA.

In fact, if we suppose that less than k nodes are compromised in the initial phase, we can solve the above question easily. When a node renews its key, the node signs the new public key using its current private key and broadcasts it to all nodes. Before accepting the new public key of the node, all other nodes verify the validity of the signature using the current public key of the node firstly. So, even if more than k nodes are compromised after the initial phase, the compromised nodes can not issue the valid public key because they don't know the private key corresponding to the current public key of the node.

5.3.3. New master key share creation

When a new node joins a network, it presents its identity and some other required physical proof (depending on key issuing policy) to k neighbor nodes and requests PKG service, the master public key and his share of the master private key. Each

node in the coalition verifies the validity of the identity of the new node $\mathtt{ID}_m$. If the verification process succeeds, the private key can be generated using the method described in the previous section.

To initialize the share of master key for the requesting node, the requesting node broadcasts its pubic key to its neighbor nodes firstly, and then each coalition node $\mathtt{ID}_i$ generates the partial share $x_{im} = x_i\lambda_i^{\Phi}$ for node $\mathtt{ID}_m$. Here, x_i is the share of master private key of node $\mathtt{ID}_i$ and λ_i^{Φ} is the Lagrange term. It encrypts the partial share using the public key of requesting node and sends it to node $\mathtt{ID}_m$. Node $\mathtt{ID}_m$ obtains its new share by adding the partial shares as $x_m = \sum_{i=1}^{k} x_{im}$. Note that the partial shares may be shuffled before being sent to the joining node to protect the secrecy of the coalition nodes'secret shares [25]. After obtaining the share of the master private key, the new joining node is available to provide PKG service to other joining nodes.

5.3.4. User-controlled key renewal

A node might want to change his keys from time to time. If he has any suspicion that his key is compromised, he will need to change his key pair. In other words, if a node changes his key pair frequently, the risk that he uses compromised key will be reduced a lot. Due to our key pair generation mechanism, node can renew his key pair without any interaction with any other node.

Let $(pk = (pk^{(1)}, pk^{(2)}, (R, \sigma)), sk)$ be the basic key pair of node $\mathtt{ID}$. He can generate a renewed key pair (pk', sk') by the following procedure.

(1) Set $pk'(1) = pk^{(1)}$.
(2) Pick $k' \in \mathbb{Z}_q^*$ at random and compute $pk'(3) = g^{k'}$.
(3) Compute $sk' = sk \cdot H_1(\mathrm{NID}, pk'(3)) + k'$ and

$$pk'(2) = g^{sk'} = g^{sk \cdot H_1(\mathrm{NID}, pk'(3)) + k'} = (pk^{(2)})^{H_1(\mathrm{NID}, pk'(3))} \cdot pk'(3) = (pk^{(1)} y^{H_1(\mathrm{NID}, pk^{(1)})})^{H_1(\mathrm{NID}, pk'(3))} \cdot pk'(3)$$
$$= (pk'(1) y^{H_1(\mathrm{NID}, pk'(1))})^{H_1(\mathrm{NID}, pk'(3))} \cdot pk'(3).$$

(4) Sign his network identifier (NID) and $pk'(1)$, $pk'(2)$, $pk'(3)$ using his private key sk' and Schnorr's signature scheme.
   (a) choose a random $r' \in \mathbb{Z}_q^*$,
   (b) compute $R' = g^{r'} \bmod p$, and
   (c) set the signature to be (R', σ'), where

$$\sigma' = r' + sk' \cdot H'(\mathrm{NID}, pk'(1), pk'(2), pk'(3), R').$$

Finally, set his public key $pk' = (pk'(1), pk'(2), pk'(3), (R', \sigma'))$ and broadcast it to all nodes in the network.

Any other node can explicitly verify the validity of $pk' = (pk'(1), pk'(2), pk'(3), (R', \sigma'))$ by testing if

$$pk'(2) = (pk'(1) y^{H_1(\mathrm{NID}, pk'(1))})^{H_1(\mathrm{NID}, pk'(3))} \cdot pk'(3)$$

and $g^{\sigma'} = R'(pk'(2))^{H'}(\mathrm{NID}, pk'(1), pk'(2), pk'(3), R')$.

5.3.5. Key revocation

In order to provide key revocation in wireless ad hoc networks we need four mechanisms. First, nodes need to be able to revoke their own public key, which we refer to as *harakiri*. Second, nodes can revoke the public keys of compromised or suspicious nodes, which we refer to as *accusation*. Third, we need a mechanism to inform all nodes in the network about these revocations. And last, we need a mechanism for newly joining to obtain a list of all revoked or accused public keys. We can use the same accusation scheme – *neighborhood watch* in [17] to obtain these mechanisms (For details see [17]).

In fact, due to the function – *user-controlled key renewal* in our solution, each node can renew his key pair frequently, and key revocation becomes redundant.

5.3.6. Authentication

Authentication enables a mobile node to ensure the identity of the peer node it is communicating with, so that no attacker could masquerade a node, thus gaining unauthorized access to resource and sensitive information.

The first authentication method is a sign-and-encryption procedure, in which digital signature is used for the authentication of messages and encryption is used for the confidentiality of messages. Alternatively, authentication can be obtained using a more efficient method.

Due to our key generation mechanism, the communicating nodes can generate a shared secret (session key) on both sides without additional key exchange. One source node side, the shared secret is generated using source node's private key and the destination node's public key, while the destination node compute this secret using its private key and source nodes public key. Let $\left(pk_s = \left(pk_s^{(1)}, pk_s^{(2)}, (R_s, \sigma_s)\right), sk_s\right)$ and $\left(pk_d = \left(pk_d^{(1)}, pk_d^{(2)}, (R_d, \sigma_d)\right), sk_d\right)$ be the basic key pair of the source node S and destination node D respectively. They can compute the shared secret

$$g^{sk_s \cdot sk_d} = \left(pk_s^{(2)}\right)^{sk_d} = \left(pk_d^{(2)}\right)^{sk_s}.$$

Therefore it is indeed the Diffie-Hellman key agreement protocol. The generated shared secret can be used as a symmetric key to encrypt and authentication.

5.3.7. Performance analysis

The communication overhead in our approach is mainly introduced by key generation component. In the network initial phase, n nodes need to jointly generate the master key pair in a self-organizing way, which increases the network setup time. Moreover, every mobile node needs to broadcast a key generation request to its neighborhood, and each PKG service node needs interaction with the requesting node in order to generate his sub-share of the private key of the requesting node. The PKG service in the network initial phase also causes more communication overhead. But it is a tradeoff between security and communication overhead. In addition, due to the user-controlled key renewal, the communication overhead becomes appropriate. The computation overhead is also appropriate, due to without pairing computation.

6. Concluding remarks

In this paper, we presented the first SGC-encryption scheme that does not depend on the pairing. We proved in the random oracle model that the scheme is IND-CCA and DoD-Free secure, relative to the hardness of the standard CDH problem and DL problem. However, we can only achieve security in the random oracle model, although our scheme has many appealing properties. It is still an open problem to design a CL-PKC and SGC-PKC scheme without pairing that is secure in the standard model.

We also discussed how our SGC-encryption scheme can be used to design a secure and self-organized key management and authentication system for ad hoc wireless networks with a function of user-controlled key renewal.

Acknowledgements

The authors thank anonymous reviewers for their valuable comments.

References

[1] S.S. Al-Riyami, K. Paterson, Certificateless public key cryptography, in: Proc. ASIACRYPT 2003, LNCS, vol. 2894, Springer-Verlag, 2003, pp. 452–473.
[2] S.S. Al-Riyami, K. Paterson, Certificateless public key cryptography, Cryptology ePrint Archive, Report 2003/126, 2003. <http://eprint.iacr.org/2003/126>.
[3] M.H. Au, J. Chen, J.K. Liu, Y. Mu, D.S. Wong, G. Yang, Malicious KGC attacks in certificateless cryptography, in: ASIACCS'07, ACM, 2007, pp. 302–311.
[4] P. Bergamo, P. D'Arco, A. De Santis, L. Kocarev, Security of public key cryptosystems based on Chebyshev polynomials, IEEE Transactions on Circuits and Systems I 52 (7) (2005) 1382–1393.
[5] K. Bentahar, P. Farshim, J. Malone-Lee, Generic constructions of identity-based and certificateless KEMs, Cryptology ePrint Archive, Report 2005/058, 2005. <http://eprint.iacr.org/2005/058>.
[6] M. Bellare, P. Rogaway, Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, in: ACM CCCS'93, 1993, pp. 62–73.
[7] J. Baek, R. Safavi-Naini, W. Susilo, Certificateless public key encryption without pairing, ISC 05, LNCS 3650, Springer-Verlag, 2005, pp. 134–148.
[8] Z. Cheng, R. Comley. Efficient certificateless public key encryption, Cryptology ePrint Archive, Report 2005/012, 2005. <http://eprint.iacr.org/2005/012>.
[9] K.Y. Choi, J.H. Park, J.Y. Hwang, D.H. Lee, Efficient certificateless signature schemes, in: ACNS'07, LNCS, vol. 4521, Springer-Verlag, 2007, pp. 443–458.
[10] S. Chow, C. Boyd, J. Gonzalez, Security-mediated certificateless cryptography, in: PKC 2006, LNCS, vol. 3958, Springer-Verlag, 2006, pp. 508–524.
[11] C. Crépeau, C.R. Davis, A certificate revocation scheme for wireless Ad Hoc Networks, in: Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03), ACM Press, 2003, pp. 54–61.
[12] K.Y. Cheong, T. Koshiba, More on Security of Public-Key Cryptosystems Based on Chebyshev Polynomials, IEEE Transactions on Circuits and Systems II, 54 (9) (2007) 795–799.
[13] W. Diffie, M. E Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22 (1976) 644–654.
[14] H. Deng, A. Mukherjee, D.P. Agrawal, Threshold and identity-based key management and authentication for Wireless Ad Hoc Networks, International Conference on Information Technology: Coding and Computing (ITCC'04) 1 (2004) 107–115.
[15] S. Duan, Certificateless undeniable signature scheme, Information Sciences 178 (3) (2008) 742–755.
[16] M. Girault, Self-certified public keys, in: Proceedings of EUROCRYPT 91, LNCS, vol. 547, Springer-Verlag, 1992, pp. 490–497.
[17] K. Hoeper, G. Gong, Key revocation for identity-based schemes in mobile ad hoc networks, in: Proceedings of Fifth International Conference on AD-HOC Networks and Wireless-AD HOC NOW, LNCS, vol. 4104, 2006, pp. 224–237.
[18] B.C. Hu, D.S. Wong, Z. Zhang, X. Deng, Key replacement attack against a generic construction of certificateless signature, in: ACISP'06, LNCS, vol. 4058, Springer-Verlag, 2006, pp. 235–246.
[19] X. Huang, W. Susilo, Y. Mu, F. Zhang, Certificateless designated verifier signature schemes, in: AINA 2006, IEEE Computer Society, 2006, pp. 15–19.
[20] X. Huang, W. Susilo, Y. Mu, F. Zhang, On the security of certificateless signature schemes from Asiacrypt 2003, in: CANS 2005, LNCS, vol. 3810, Springer-Verlag, 2005, pp. 13–25.
[21] X. Huang, Yi Mu, W. Susilo, D.S. Wong, W. Wu, Certificateless signature revisited, ACISP'07, vol. 4586, Springer-Verlag, 2007, pp. 308–322.
[22] J.-P. Hubaux, L. Buttyán, S. Čapkun, The quest for security in mobile Ad Hoc networks, ACM Symposium on Mobile Ad Hoc and Computing-MobiHoc 2001 (2001) 146–155.
[23] A. Khalili, J. Katz, W.A. Arbaugh, Toward secure key distribution in truly Ad-Hoc Networks, in: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops), IEEE Computer Society, 2003, pp. 342–346.
[24] L. Kocarev, M. Sterjev, A. Fekete, G. Vattay, Public-key encryption with chaos. Chaos 14 (4) (2004) 1078–1082.
[25] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks, in: Proceedings of the IEEE Nineth International Conference on Network Protocols (ICNP'01), 2001.
[26] J.K. Liu, M.H. Au, W. Susilo, Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model, in: ASIACCS'07, 2007, pp. 273-283. Full paper: <http://eprint.iacr.org/2006/373>.
[27] Y. Long, K. Chen, Certificateless threshold cryptosystem secure against chosen–ciphertext attack, Information Sciences 177 (24) (2007) 5620–5637.
[28] B. Lee, K. Kim, Self-Certificate: PKI using Self-Certified Key, in: Proceedings of the Conference on Information Security and Cryptology 2000, Vol. 10, No. 1, 2000, pp. 65–73.

[29] B. Libert, J. Quisquater, On constructing certificateless cryptosystems from identity based encryption, in: PKC 2006, LNCS, vol. 3958, Springer-Verlag, 2006, pp. 474–490.
[30] H. Luo, P. Zerfos, J. Kong, S. Lu, L. Zhang, Self-Securing Ad Hoc Wireless Networks, in: Seventh IEEE Symposium on Computers and Communications (ISCC'02), 2002.
[31] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.
[32] T.P. Pedersen, A threshold cryptosystem without a trusted party, in: Proc. EUROCRYPT 91, LNCS, vol. 547, Springer-Verlag, 1992, pp. 522–526.
[33] H. Petersen, P. Horster, Self-certified keys – concepts and applications, in: Third International Conference on Communications and Multimedia Security, Chapnam and Hall, 1997, pp. 102–116.
[34] D. Pointcheval, J. Stern, Security proofs for signature schemes, in: Proc. Eurocrypt 96, LNCS, vol. 1070, Springer-Verlag, 1996, pp. 387–398.
[35] B. Ranjan, Novel public key encryption technique based on multiple chaotic systems, Physical Review Letters 95 (2005).
[36] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communication ACM 21 (1978) 120–126.
[37] C.P. Schnorr, Efficient signature generation by smart cards, Journal of Cryptology 4 (3) (1991) 161–174.
[38] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–613.
[39] A. Shamir, Identity-based cryptosystems and signature schemes, in: Crypto'84, LNCS, vol. 196, Springer-Verlag, 1984, pp. 47–53.
[40] Y. Shi, J. Li, Provable efficient certificateless public key encryption, Cryptology ePrint Archive, Report 2005/287, 2005. <http://eprint.iacr.org/2005/287>.
[41] K. Shim, Breaking the short certificateless signature scheme, Information Sciences 179 (3) (2009) 303–306.
[42] D. Stinson, Cryptography: Theory and Practice, Second ed., Springer, 2002.
[43] B. Waters, Efficient identity-based encryption without random oracles, in: Proc. EUROCRYPT 2005, LNCS, vol. 3494, Springer-Verlag, 2005, pp. 114–127.
[44] L. Wang, Z. Cao, X. Li, H. Qian, Simulatability and security of certificateless threshold signatures, Information Sciences 177 (6) (2007) 1382–1394.
[45] K. Wang, W. Pei, L. Zou, Y. Cheung, Z. He, Security of public key encryption technique based on multiple chaotic systems, Physics Letters A 360 (2) (2006) 259–262.
[46] D.H. Yum, P.J. Lee, Generic construction of certificateless encryption, in: ICCSA'04, LNCS, vol. 3040, Springer-Verlag, 2004, pp. 802–811.
[47] D.H. Yum, P.J. Lee, Generic construction of certificateless signature, in: ACISP'04, LNCS, vol. 3108, Springer-Verlag, 2004, pp. 200–211.
[48] Z. Zhang, D. Wong, J. Xu, D. Feng, Certificateless public-key signature: security model and efficient construction, in: ACNS'06, LNCS, vol. 3989, Springer-Verlag, 2006, pp. 293–308.
[49] L. Zhou, Z.J. Haas, Securing Ad Hoc Networks, IEEE Network Journal 13 (6) (1999) 24–30.