An Efficient Confidentiality Preserving Scheme Using Certificateless Encryption with High Trust Level

Rui Guo^{1,2}, Huixian Shi³

(Corresponding author: Rui Guo)

National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications¹

Xi'an 710121, P.R. China

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications²

Beijing 100876, P. R. China

School of Mathematics and Information Science, Shaanxi Normal University³

Xi'an 710062, P. R. China

(Email: guorui@xupt.edu.cn)

(Received Sept. 4, 2016; revised and accepted Jan. 15, 2017)

Abstract

Certificateless public key cryptography with bilinear pairing needs costly operations, which is not appropriate for a practical application. In this paper, we present a certificateless public key encryption scheme without bilinear pairing. This encryption scheme achieves Girault's trust Level 3 as in traditional public key infrastructure based cryptography, and which is proved to be secure in the random oracle model under the hardness assumption of computational Diffie-Hellman problem. Compared with the related schemes, the performance analysis and simulation show that our scheme is more efficient than others. It takes account of security and efficiency.

Keywords: Certificateless Public Key Encryption; Wireless Sensor Networks; IND-CCA Secure; Without Pairing

1 Introduction

In a traditional public key cryptography (PKC), anyone who wants to communicate over a public channel does not share any secret key to each other. A public key infrastructure (PKI) is used to provide an assurance between a public key and the holder of the corresponding private key through the certificates issued by a certification authority (CA). However, a PKI is responsible for managing the certificate, including revocation, storage, distribution and the computational cost of certificate verification, which places a computational burden on the entity.

To simplify the key management and avoid the digital certificates, Shamir [24] proposed the notion of identity based public key cryptography (ID-PKC) which alleviates the existing problems in PKI by getting rid of certificates. In ID-PKC, a user's public key is derived directly from its identity information, such as E-mail address and IP address. The corresponding private key is fully generated by a trusted third party called private key generator (PKG). Nevertheless, there is an obvious drawback that the user must trust the PKG unconditionally (even if a malicious one), which leads to the key escrow problem. Therefore, the hostility PKG could impersonate a user and decrypt the corresponding ciphertext.

In order to resolve the inherent key escrow problem while preserving the advantages of ID-PKC, Al-Riyami and Paterson [2] introduced a new paradigm called certificateless public key cryptography (CL-PKC). Compared with ID-PKC, CL-PKC not only inherits the advantages of it but also resolves the key escrow problem. Specifically, the user combines a secret value picked by itself with the partial private key obtained from the trusted authority (called key generation center, KGC) to generate the full private key. Consequently, the KGC can not obtain the user's private keys to decrypt his ciphertext any more, which has a certain practical application value.

Wireless sensor networks (WSNs) [1] are typically composed of a large number of inexpensive, small and batterypowered sensor nodes [13]. In WSNs, all sensor nodes collaborate together to collect and process certain information such as environment monitoring, health monitoring, military sensing tracking, etc. [9, 14, 17]. In practical application, many situations require the sensor node to be employed in unprotected and even hostile environments, and therefore bring lots of research challenges. One of the important issues is security.

Unfortunately, due to the inherent characteristics of

WSNs mentioned above, adversaries can eavesdrop and distort the transmitted information and disseminate misleading messages into the networks [11]. Hence, encryption mechanisms need to be implemented to protect the message from the malicious attack.

1.1 Related Works

In Al-Rivami and Paterson's work [2], a certificateless public key cryptography was introduced firstly. Based on this work, several certificateless public key encryption (CL-PKE) schemes were proposed in [6, 8, 15, 25, 26, 30]. Cheng and Comley [6] proposed a more efficient encryption scheme on the bases of Al-Rivami and Paterson's work [2]. An interesting feature of this scheme compared with the original CL-PKE is that it can verify the legitimate of the ciphertext in the process of decryption. Libert and Quisquater [15] fixed the model of Al-Riyami and Paterson and gave a method to achieve generic CL-PKE constructions which are provably choose-ciphertext attacks secure (CCA-secure) in the random oracle model. Based on the algebraic properties of Weil pairing, Shi and Li [25] proposed a CL-PKE scheme worked in a kind of parallel model. In [30], Yum and Lee provided a generic secure construction of CL-PKE. In order to resist the strong adversaries in the standard model, Dent et al. [8] presented the first strongly secure CL-PKE scheme in 2008. In 2010, Sun and Li [26] constructed a short-ciphertext CL-PKE scheme in standard model, which achieved adaptive chosen ciphertext security (CCA2-secure). However, the above schemes suffered low efficiency due to the bilinear pairing [5]. Being aware of the above problem of the current constructions of CL-PKE scheme, Baek et al. [3] proposed a CL-PKE scheme without pairing firstly in 2005. In 2011, Lai et al. [12] modified Baek et al.'s scheme to get a Girault's trust Level 3 [10].

Encryption in WSNs is an important mechanism to guarantee the confidence of the transmitted information. Earlier research mainly focused on designing symmetric key based encryption schemes, such as AES-CCM^{*} [31], RC6 [21], Skipjack [19] and so on. There are common issues of key storage and distribution in these schemes, which causes some security threats for WSNs. Considering the security of these symmetric key based encryption schemes, several public key based schemes [4, 7, 20, 29] in WSNs have been proposed to avoid these drawbacks. In 2008, Baek et al. [4] introduced a technique called "indexing", whose performance in WSNs in terms of computation and communication costs is significant superior to normal public key encryption. Watro et al. designed a public key encryption protocol that allows authentication and key agreement between a sensor network and a third party as well as between two sensor networks in [29]. Chu et al. and Oliveria et al. [7, 20] supposed the identitybased encryption scheme for WSNs respectively, which eliminated the problem of certificate management and reduced the costly computation and communication obviously in the public key encryption schemes. Nevertheless,

public key based encryption scheme also has a general shortcoming that it is more complicated than symmetric key based one. Therefore, it becomes vitally important to construct the efficient public key based encryption scheme in WSNs for data computation while preserving the level of its security.

1.2 Our Contributions

In this paper, for the reason of applying in WSNs, we propose a CL-PKE scheme that avoid the use of bilinear pairing. Provided that the computational Diffie-Hellman (CDH) problem is intractable, we also prove that the proposal is secure in the random oracle model. Compared with the existing related schemes by simulation, our scheme offers better performances on running time, energy consumption and communication bandwidth. Furthermore, this work promotes the trust level of KGC to the highest Level 3, which strengthen the security of the whole system.

The remainder of this paper is organized as follows. In Section 2, we give some preliminaries such as the definition of the Girault's trust level, the model and the security definitions of CL-PKE and some computational problems. In Section 3, we propose a CL-PKE scheme without bilinear pairing. In Section 4, we analyze the security of our proposal. In Section 5, we make a performance analysis of this scheme. Finally, we conclude the paper in Section 6.

2 Preliminaries

In this section, we review the definition of the Girault's trust level, the model and the security definition of CL-PKE as well as some computational problems which form the basis of the security in our scheme.

2.1 Girault's Trust Level

The Girault's trust level [10] provides the trust hierarchy for public key cryptography, which can be used to evaluate the creditability of the authority.

- Level 1. The authority (e.g. the CA in a PKI, the KGC in an identity-based or certificateless cryptography) knows (or can easily compute) users' secret keys. Therefore, the authority can impersonate any user at any time without being detected.
- Level 2. The authority does not know (or cannot easily compute) users' secret keys. Nevertheless, it can still impersonate users by generating false guarantees (e.g. false certificates in a PKI, false public keys in a certificateless cryptography).
- Level 3. The authority cannot compute users' secret keys, and it can be proven that it generates false guarantees of users if it does so.

According to these definitions, we can easily find that the original certificateless cryptography falls into Level 2, and a traditional PKI achieves Level 3.

2.2 Definition of CL-PKE

The model of CL-PKE in our proposal is similar to that of [3] but with a crucial difference which makes the scheme reach Girault's trust Level 3.

A CL-PKE scheme consists of seven probabilistic, polynomial time (PPT) algorithms: Setup, User-Key-Generation, Partial-Key-Extract, Set-Private-Key, Set-Public-Key, Encrypt and Decrypt.

- **Setup:** Taking security parameter k as input, the KGC returns a randomly chosen master secret key msk and a list of public parameters param.
- **User-Key-Generation:** Taking a list of public parameters param as input, the user returns a secret key sk and a public key pk.
- **Partial-Key-Extract:** Taking param, msk, a user's identity ID and pk received from the user as input, the KGC returns a partial private key D_{ID} and a partial public key P_{ID} .
- **Set-Private-Key:** Taking *param*, D_{ID} and sk as input, the user returns a private key SK_{ID} .
- **Set-Public-Key:** Taking $param, P_{ID}$ and pk as input, the user returns a public key PK_{ID} .
- **Encrypt:** Taking a plaintext M, a list of parameters *param*, a receiver's identity ID and PK_{ID} as inputs, the sender returns a ciphertext C.
- **Decrypt:** Taking *param*, SK_{ID} and the ciphertext C as input, the receiver runs this deterministic algorithm and returns a decryption δ , which is either a plaintext message or a "*Reject*" message.

The algorithm of User-Key-Generation is similar to the algorithm of Set-Secret-Value in Baek et al.'s definition [3]. However, the User-Key-Generation in this definition must be run prior to the Partial-Key-Extract algorithm. According to this operation model, the scheme enjoys the same trust level as the traditional PKI.

2.3 Security Model for CL-PKE

In CL-PKE scheme, as defined in [2], there are two types of adversary with different capabilities. We assume Type I adversary, \mathcal{A}_I acts as a dishonest user who does not have the KGC's master secret key msk but it can replace public keys of arbitrary identities with other public keys of its own choices. While Type II adversary, \mathcal{A}_{II} acts as a malicious KGC who knows the master secret key msk(hence it can compute partial secret key by itself) and is

allowed to obtain full secret keys for arbitrary identities but cannot replace any user's public key.

Definition 1. A CL-PKE scheme Π is said to be secure against adaptive chosen ciphertext attack (IND-CCA secure) if no polynomially bounded adversary \mathcal{A} of Type I and Type II has a non-negligible advantage in the following game played against the challenger:

Setup: The challenger C takes a security parameter k as input and runs the **Setup** algorithm in Section 2.2, then sends the resulting system parameters *param* to A. If A is of Type I, C keeps the master secret key msk to itself. Otherwise (e.g. if A is of Type II), it gives msk to A.

Phase 1: \mathcal{A} is given access to the following oracles:

- 1) **Public-Key-Request-Oracle**: Upon receiving a public key query for a user's identity ID, C computes (sk, pk) and (P_{ID}, D_{ID}) , then computes PK_{ID} and returns it to A.
- 2) **Partial-Key-Extract-Oracle**: Upon receiving a partial key query for a user's identity ID and pk, C computes (P_{ID}, D_{ID}) and returns them to A. (Note that it is only useful to Type I adversary.)
- 3) **Private-Key-Request-Oracle**: Upon receiving a private key query for a user's identity ID, C computes (sk, pk) and (P_{ID}, D_{ID}) , then computes SK_{ID} and returns it to A. It outputs \perp (denotes failure) if the user's public key has been replaced (in the case of Type I adversary).
- 4) **Public-Key-Replace-Oracle**: For identity ID and a valid public key, \mathcal{A} replaces the associated user's public key with the new one of its choice (this is only for Type I adversary). The new value will be recorded and used by \mathcal{C} in the coming computations or responses to the adversary's queries.
- 5) **Decryption-Oracle**: On input a ciphertext and an identity, return the correct decryption of ciphertext using the private key corresponding to the current value of the public key associated with the identity of user, even if the corresponding public key for the user ID has been replaced.
- Challenge Phase: Once \mathcal{A} decides that Phase 1 is over, it outputs and submits two messages (M_0, M_1) , together with a challenge identity ID^{*} of uncorrupted secret key. Note that \mathcal{A} is not allowed to know the private key of ID^{*} in anyway. The challenger \mathcal{C} picks a random bit $\beta \in \{0, 1\}$ and computes C^* , which is the encryption of M_β under the current public key PK_{ID^*} for ID^{*}. If the output of the encryption is \perp, \mathcal{A} immediately losses the game. Otherwise, C^* is delivered to \mathcal{A} .
- **Phase 2:** Now \mathcal{A} issues the second sequence of queries as in **Phase 1**. A decryption query on the challenge

ciphertext C^* for the combination of ID^{*} and PK_{ID^*} is not allowed.

Guess: Finally, \mathcal{A} outputs its guess β' for β . The adversary wins the game if $\beta' = \beta$ and the advantage of \mathcal{A} in this game is defined to be $Adv(\mathcal{A}) = |\Pr(\beta' = \beta) - \frac{1}{2}|$. The adversary \mathcal{A} breaks an IND-CCA secure CL-PKE scheme II with $(t, q_{H_i}, q_{par}, q_{pub}, q_{prv}, q_D, \epsilon)$ if and only if the guessing advantage of \mathcal{A} that makes q_{H_i} times to random oracles H_i , 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** queries is greater than ϵ within running time t. The scheme II is said to be $(t, q_{H_i}, q_{par}, q_{pub}, q_{prv}, q_D, \epsilon)$ - IND-CCA secure if there is no attacker \mathcal{A} that breaks IND-CCA secure scheme II with $(t, q_{H_i}, q_{par}, q_{pub}, q_{prv}, q_{D}, \epsilon)$.

2.4 Computational Problems

Now, it will be introduced the Discrete Logarithm (DL) problem and Computational Diffie-Hellman (CDH) problem that are needed in the security analysis of our scheme.

Definition 2 (DL problem). Let G be a cyclic additive group of prime order p and P be a generator of G. Define Q = xP for uniformly chosen $x \in Z_p^*$. Given (P,Q), adversary \mathcal{A} tries to find the value of x.

Definition 3 (CDH problem). Let G be a cyclic additive group of prime order p and P be a generator of G. Define Q = xP, R = yP for uniformly chosen $x, y \in Z_p^*$. Given (P, Q, R), adversary \mathcal{A} tries to find the value of xyP.

Let \mathcal{A} be a CDH adversary. \mathcal{A} 's advantage to solve the CDH problem is defined as $Adv(\mathcal{A}) = |\Pr[\mathcal{A}(P, xP, yP) = xyP]|$ and the probability is measured over random choices of $x, y \in \mathbb{Z}_p^*$ and the point P. \mathcal{A} solves the CDH problem with (t, ϵ) 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 adversary \mathcal{A} that solves the CDH problem with (t, ϵ) .

3 Our Construction

In this section, we propose a new CL-PKE scheme without pairing in a cyclic additive group G which performs well.

The notations used throughout this paper are listed in Table 1. Our proposed CL-PKE scheme consists of the following seven PPT algorithms.

Setup: Generate a large prime p, which makes the CDH problem in the cyclic additive group G with generator P of order p be intractable. Pick $x \in Z_p^*$ uniformly at random and compute X = xP. Choose hash function $H_1: \{0, 1\}^* \times G^* \times G^* \to Z_p^*, H_2: \{0, 1\}^{l_0} \times \{0, 1\}^{l_1} \to$ $Z_p^*, H_3: G^* \times G^* \to \{0, 1\}^l$, where $l = l_0 + l_1 \in N$.

Table 1: Notation defined and used in this scheme

ID	the public identity of the entity
$H_i(\cdot)$	the collision-resistant hash function $(i=1, 2, 3)$
p	the large prime
G	the cyclic additive group
P	the generator of G
x	the master secret key
X	the master public Key
P_{ID}	the entity ID's partial public key
D_{ID}	the entity ID's partial secret key
PK_{ID}	the entity ID's public key
SK_{ID}	the entity ID's secret key
	the concatenation operation
\oplus	the bitwise XOR operation
N	the set of positive integer

Return $param = \{p, P, G, X, H_1, H_2, H_3\}$ and master secret key $msk = \{x\}$.

- **User-Key-Generation:** Pick $y \in Z_p^*$ at random and compute Y = yP. Return (sk, pk) = (y, Y).
- **Partial-Key-Extract:** Pick $s \in Z_p^*$ at random and compute $\omega = sP$ and $d = (s + xH_1(ID||\omega||pk))$, where ID is a user's identity. Return $(P_{ID}, D_{ID}) = (\omega, d)$.
- **Set-Private-Key:** Set $SK_{ID} = (sk, D_{ID}) = (y, d)$. Return SK_{ID} .
- **Set-Public-Key:** Set $PK_{ID} = (pk, P_{ID}) = (Y, \omega)$. Return PK_{ID} .
- **Encrypt:** Let the bit-length of M be l_0 . Parse PK_{ID} as (Y, ω) , pick $\sigma \in \{0, 1\}^{l_1}$ at random and compute $r = H_2(M \| \sigma)$. Compute

$$Q_{ID} = H_1(ID ||\omega|| pk)X + P_{ID},$$

$$C = (c_1, c_2) = (rQ_{ID}, H_3(z_1 ||z_2) \oplus (M ||\sigma)),$$

where $z_1 = rY$, $z_2 = Q_{ID}$ (Note that the bit-length of $(M||\sigma)$ is equal to $l = l_0 + l_1$). Return $C = (c_1, c_2)$.

Decrypt: To decrypt $C = (c_1, c_2)$, compute

$$M||\sigma = H_3(d^{-1}yc_1||dP) \oplus c_2.$$

If $H_2(M \| \sigma) dP = c_1$, return *M*. Else, return "*Reject*".

The above decryption algorithm is consistent if and only if (c_1, c_2) is the valid ciphertext of M, then we have:

$$\begin{aligned} H_{3}(d^{-1}yc_{1}||dP) \oplus c_{2} \\ &= H_{3}(d^{-1}yrQ_{ID}||dP) \oplus c_{2} \\ &= H_{3}(d^{-1}yr(H_{1}(\mathrm{ID}||\omega||pk)X + P_{ID})||dP) \oplus c_{2} \\ &= H_{3}(d^{-1}yr(H_{1}(\mathrm{ID}||\omega||pk)xP + sP)||dP) \oplus c_{2} \\ &= H_{3}(d^{-1}yrdP||Q_{ID}) \oplus c_{2} \\ &= H_{3}((ryP)||Q_{ID}) \oplus H_{3}(z_{1}||z_{2}) \oplus (M||\sigma) \\ &= H_{3}((rY)||Q_{ID}) \oplus H_{3}(z_{1}||z_{2}) \oplus (M||\sigma) \\ &= M||\sigma. \end{aligned}$$

The intuition behind this construction is as follows. According to the key issuing technique of the Schnorr signature [22], our scheme only requires addition and scalar multiplication instead of transforming the addition group into multiplication group, which avoids the use of the bilinear pairing and raises efficiency of this protocol. Simultaneously, the User-Key-Generation algorithm operates prior to the Partial-Key-Extract algorithm. In this way, the **Partial-Key-Extract** algorithm includes pk generated by the user as input, and the creditability of the authority achieves to Girault's trust Level 3. Specifically, provided that KGC replaces a user's key pk, there will exist two working keys pk and pk' for this user. Furthermore, two working public keys PK_{ID} and PK'_{ID} binding an identity ID can result from two partial private keys, and only the KGC has the ability to generate these two partial private keys. Therefore, the KGC's forgery is easily detected.

4 Security Analysis

In this subsection, we will show that the scheme described in the previous is secure in the random oracle model.

Theorem 1. The CL-PKE scheme is IND-CCA secure in the random oracle model, assuming that the CDH problem is intractable.

Proof. In order to prove this theorem, we prove that our CL-PKE scheme is secure against the Type I and Type II attackers (A_I and A_{II}) whose behaviors are as described in **Definition 1**.

Assuming there exists an adversary \mathcal{A} . Suppose that another PPT \mathcal{B} can make use of \mathcal{A} to solve the CDH problem with probability at least ϵ' and in the time at most t'.

- **Stage 1:** Suppose that \mathcal{A} in this stage is the Type I adversary \mathcal{A}_I and \mathcal{B} is given (p, P, aP, xP) as an instance of the CDH problem. In order to solve the problem by using of \mathcal{A}_I , \mathcal{B} needs to simulate a challenger to execute each phase of IND-CCA game for \mathcal{A}_I as follows:
- **Setup:** \mathcal{B} sets X = xP, where $x \in Z_p^*$ is the master secret key which is unknown to \mathcal{B} , then gives \mathcal{A}_I

 (p, P, X, H_1, H_2, H_3) as *param*, where H_1, H_2, H_3 are random oracles. Adversary \mathcal{A}_I may make queries of all random oracles at anytime during its attacks as follows:

- H_1 queries: On receiving a query (ID, ω , Y) to H_1 :
 - 1) If $\langle (\text{ID}, \omega, Y), e \rangle$ exists in H_1 List L_1 , return e as answer.
 - 2) Otherwise, pick $e \in Z_p^*$ at random, add $\langle (\mathrm{ID}, \omega, Y), e \rangle$ to L_1 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_2 List L_2 , return r as answer.
- 2) Otherwise, pick $r \in Z_p^*$ at random, add $\langle (M, \sigma), r \rangle$ to L_2 and return r as answer.

 H_3 queries: On receiving a query (z_1, z_2) to H_3 :

- 1) If $\langle (z_1, z_2), R \rangle$ exists in H_3 List L_3 , return R as answer.
- 2) Otherwise, pick $R \in \{0, 1\}^l$ at random, add $\langle (z_1, z_2), R \rangle$ to L_3 and return R as answer.

Phase 1: A_I can issue the following oracle queries.

Partial-Key-Extract: On receiving a query ID:

- 1) If $(\text{ID}, (\omega, d))$ exist in **PartialKeyList**, return (ω, d) as answer.
- 2) Otherwise, pick $d, e \in Z_p^*$ at random and compute $\omega = dP eX$. Add (ID, ω, e) to L_1 (That is, e is defined to be $H_1(\text{ID} || \omega || Y)$.) and $\langle \text{ID}, (\omega, d) \rangle$ to **PartialKeyList**, return (ω, d) as answer.

Note that we have $\omega + XH_1(ID||\omega||pk) = dP$ in the above simulation which holds in the real attack too.

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

- 1) If $\langle \text{ID}, (\omega, Y), coin \rangle$ exists in **PublicKeyList**, return $PK_{ID} = (\omega, Y)$ 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, do the following:
 - a. If $\langle \text{ID}, (\omega, d) \rangle$ exists in **PartialKeyList**, pick $y \in Z_p^*$ at random and compute Y = yP, add $\langle \text{ID}, (y, d) \rangle$ to **PrivateKeyList** and $\langle \text{ID}, (\omega, Y), coin \rangle$ to **PublicKeyList**, return $PK_{ID} = (\omega, Y)$ as answer.
 - b. Otherwise, run the above simulation algorithm for partial key extraction taking ID as input to get partial key (ω, d) , pick $y \in Z_p^*$ at random and compute Y = yP, add $\langle \text{ID}, (y, d) \rangle$ to **PrivateKeyList** and $\langle \text{ID}, (\omega, Y), coin \rangle$ to **PublicKeyList**, return (ω, Y) as answer.

4) Otherwise (if coin = 1), pick $y, s \in Z_p^*$ at random and compute $\omega = sP, Y = yP$, add $\langle \text{ID}, (y, *), s \rangle$ to **PrivateKeyList**, and $\langle \text{ID}, (\omega, Y), coin \rangle$ to **PublicKeyList**, return $PK_{ID} = (\omega, Y)$ as answer.

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

- 1) Run **Public-Key-Request** on ID to get a tuple $(ID, (\omega, Y), coin) \in \mathbf{PublicKeyList}.$
- 2) If coin = 0, search **PrivateKeyList** for a tuple (ID, (y, d)) and return $SK_{ID} = (y, d)$ as answer.
- 3) Otherwise, return "Abort" and terminate.
- **Decryption:** On receiving a query (ID, PK_{ID} , C), **Phase 2:** In this phase, \mathcal{B} answers \mathcal{A}_I 's queries in the where $C = (c_1, c_2)$ and $PK_{ID} = (\omega, Y)$: same way as it have done in **Phase 1**. Note that
 - 1) Search **PublicKeyList** for a tuple $\langle \text{ID}, (\omega, Y), coin \rangle$.
 - 2) If such a tuple exists and coin = 0,
 - a. Search **PrivateKeyList** for a tuple $\langle \text{ID}, (y, d) \rangle$. (Note that $\langle \text{ID}, (\omega, Y), coin \rangle$ must exist in **PublicKeyList** and when coin = 0, $\langle \text{ID}, (y, d) \rangle$ exist in **PrivateKeyList**.)
 - b. Compute $M || \sigma = H_3(d^{-1}yc_1 || dP) \oplus c_2$.
 - c. If $c_1 = H_2(M \| \sigma) dP$, return M and "Reject" otherwise.
 - 3) Else if such a tuple exists and coin = 1,
 - a. Run H_1 queries to get a tuple $\langle (\mathrm{ID}, \omega, Y), e \rangle$.
 - b. If there exist $\langle (M,\sigma),r\rangle \in L_2$ and $\langle (z_1,z_2),R\rangle \in L_3$ such that $c_1 = rQ_{ID}, c_2 = R \oplus (M||\sigma), z_1 = rY, z_2 = Q_{ID},$ return M and "Reject" otherwise. The pair $\langle (M,\sigma),r\rangle$ satisfies the above condition uniquely exists in L_2 as the encryption function is injective with respect to $\langle ID, \omega, Y \rangle$.
 - 4) Else if such a tuple does not exist (This is the case when the public key of a target user is replaced by \mathcal{A}_I),
 - a. Run H_1 **queries** to get a tuple $\langle (\mathrm{ID}, \omega, Y), e \rangle$.
 - b. If there exist $\langle (M,\sigma),r\rangle \in L_2$ and $\langle (z_1, z_2), R\rangle \in L_3$ such that $c_1 = rQ_{ID}, c_2 = R \oplus (M||\sigma), z_1 = rY, z_2 = Q_{ID},$ return M and "Reject" otherwise.
- **Challenge:** Once \mathcal{A}_I decides that **Phase 1** is over, then it outputs two messages (M_0, M_1) and a challenge identity ID^{*}. On receiving a challenge query $\langle \text{ID}^*, (M_0, M_1) \rangle$, \mathcal{B} does the following:
 - 1) Run **Public-Key-Request** on ID^{*} to get a tuple $(ID^*, (\omega^*, Y^*), coin) \in \mathbf{PublicKeyList}.$
 - 2) If coin = 0, return "Abort" and terminate.

- 3) Otherwise, do the following:
 - a. Search **PrivateKeyList** for a tuple $\langle \text{ID}^*, (y^*, *), s^* \rangle$. (In this case, we know that $Y^* = y^* P, \omega^* = s^* P$.)
 - b. Pick $\sigma^* \in \{0,1\}^{l_1}$, $c_2^* \in \{0,1\}^l$ and $\beta \in \{0,1\}$ at random.
 - c. Set $c_1^* = aQ_{ID^*}$, $\Omega = a\omega^*$ and $e^* = H_1(\mathrm{ID}^* || \omega^* || Y^*)$.
 - d. Define $a = H_2(M_\beta || \sigma^*)$ and $H_3(aY^* || \omega^* + XH_1(\mathrm{ID}^* || \omega^* || Y^*)) = c_2^* \oplus (M_\beta || \sigma^*)$. (Note that \mathcal{B} does not know "a".)
- 4) Return $C^* = (c_1^*, c_2^*)$ as a target ciphertext.
- **Phase 2:** In this phase, \mathcal{B} answers \mathcal{A}_I 's queries in the same way as it have done in **Phase 1**. Note that there is no **Partial-Key-Extract** query or **Private-Key-Request** query on ID* to be issued. Also, no **Decryption** query should be made on C^* for the combination of ID* and PK_{ID^*} that encrypted plaintext M_{β} .

Guess: \mathcal{A}_I outputs a guess β' . Now \mathcal{B} returns the set

 $S = \{ \frac{1}{e^*} (az_{2i} - \Omega) | z_{2i} \text{ is the second component}$ of queries to H_3 for $i \in [1, q_{H_3}]$ such that $e^* = H_1(\mathrm{ID}^* || \omega^* || Y^*) \}.$

Then, \mathcal{B} will be able to solve the CDH problem by picking $\frac{1}{c^*}(az_{2i} - \Omega)$ from S.

Analysis: From the construction of H_1 , it is clear that the simulation of H_1 is perfect. As long as \mathcal{A}_I does not query (M_β, σ^*) to H_2 nor aY^* and $\omega^* + Xe^*$ to H_3 where $e^* = H_1(\mathrm{ID}^* || \omega^* || Y^*)$, the simulations of H_2 and H_3 are perfect. By Ask H_3^* we denote the event that $(aY^*, \omega^* + Xe^*)$ has been queried to H_3 . Also, by Ask H_2^* we denote the event that (M_β, σ^*) has been queried to H_2 . If it happens, \mathcal{B} will be able to solve the CDH problem by choosing a tuple $\langle (z_1, z_2), R \rangle$ from L_3 and computing $\frac{1}{e^*}(az_{2i} - \Omega)$ with the probability at least $\frac{1}{q_{H_3}}$. Hence, we have $\epsilon' \geq \frac{1}{q_{H_2}} \Pr[\mathsf{Ask}H_3^*]$.

It is easy to notice that if \mathcal{B} does not abort, the simulations of **Partial-Key-Extract**, **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 not been replaced or PK_{ID} has not been produce under coin = 1, the simulation is perfect as \mathcal{B} knows the private key SK_{ID} corresponding to PK_{ID} . Otherwise, simulation error may occur if \mathcal{B} runs the decryption oracle simulation specified above. Let DecErr be this event. Suppose ID, PK_{ID} and C, where $C = (c_1, c_2)$ and $PK_{ID} = (\omega, Y)$, have been issued as a valid decryption query. Even if C is valid, there is a possibility that C can be produced without querying (rY, Q_{ID}) to H_3 , where $r = H_2(M \| \sigma)$. Let Valid be an event that C is valid, $\mathsf{Ask}H_3$ and $\mathsf{Ask}H_2$ respectively be events that $(aY, \omega + Xe)$ has been queried to H_3 and (M, σ) has been queried to H_2 with $C = (c_1, c_2) = (rQ_{ID}, H_3(rY||Q_{ID}) \oplus (M||\sigma))$ and $PK_{ID} = (\omega, Y)$, where $r = H_2(M||\sigma)$. Since DecErr is an event that $\mathsf{Valid}|\neg\mathsf{Ask}H_3$ happens during the entire simulation and q_D decryption oracle queries are made, we have $\Pr[\mathsf{DecErr}] = q_D \Pr[\mathsf{Valid}|\neg\mathsf{Ask}H_3]$. However,

$$\begin{split} \Pr[\mathsf{Valid}|\neg\mathsf{Ask}H_3] &\leq \Pr[\mathsf{Valid} \land \mathsf{Ask}H_2|\neg\mathsf{Ask}H_3] \\ &\quad + \Pr[\mathsf{Valid} \land \neg\mathsf{Ask}H_2|\neg\mathsf{Ask}H_3] \\ &\leq \Pr[\mathsf{Ask}H_2|\neg\mathsf{Ask}H_3] \\ &\quad + \Pr[\mathsf{Valid}|\neg\mathsf{Ask}H_2 \land \neg\mathsf{Ask}H_3] \\ &\leq \frac{q_{H_2}}{2^{l_1}} + \frac{1}{p} \end{split}$$

The event $(\mathsf{Ask}H_3^* \lor (\mathsf{Ask}H_2^* | \neg \mathsf{Ask}H_3^*) \lor \mathsf{DecErr}) | \neg \mathsf{Abort}$ be denoted by E, where Abort denotes an event that \mathcal{B} aborts during the simulation. The probability $\neg \mathsf{Abort}$ that happens is given by $\delta^{q_{prv}}(1-\delta)$ which is maximized at $\delta = 1 - \frac{1}{q_{prv}+1}$. Hence, we have $\Pr[\neg \mathsf{Abort}] \leq \frac{1}{e(q_{prv}+1)}$, where *e* denotes the base of the natural logarithm.

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

By definition of ϵ , we have

$$\begin{aligned} \epsilon &< |\Pr[\beta' = \beta] - \frac{1}{2}| \\ &= |\Pr[\beta' = \beta|\neg\mathsf{E}]\Pr[\neg\mathsf{E}] + \Pr[\beta' = \beta|\mathsf{E}]\Pr[\mathsf{E}] - \frac{1}{2}| \\ &\leq |\frac{1}{2}\Pr[\neg\mathsf{E}] + \Pr[\mathsf{E}] - \frac{1}{2}| = |\frac{1}{2}(1 - \Pr[\mathsf{E}]) + \Pr[\mathsf{E}] - \frac{1}{2}| \\ &= \frac{1}{2}\Pr[\mathsf{E}] \\ &\leq \frac{1}{2\Pr[\neg\mathsf{Abort}]}(\Pr[\mathsf{Ask}H_3^*] + \Pr[\mathsf{Ask}H_2^*|\neg\mathsf{Ask}H_3^*] + \Pr[\mathsf{DecErr}]) \end{aligned}$$

$$\leq \frac{e(q_{prv}+1)}{2}(q_{H_3}\epsilon' + \frac{q_{H_2}}{2^{l_1}} + \frac{q_Dq_{H_2}}{2^{l_1}} + \frac{q_D}{p}).$$

Consequently, we obtain $\epsilon' > \frac{1}{q_{H_3}} \left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{l_1}} - \frac{q_{DQ}}{2^{l_1}} - \frac{q_D}{p}\right)$. The running time of the CDH adversary \mathcal{B} is $t' > t + 2(q_{pub} + q_{prv})t_{sm} + q_{par}t_{sm} + 2q_Dq_{H_2}q_{H_3}t_{sm} + 3t_{sm}$, where t_{sm} denotes the time for computing scalar multiplication on the cyclic addition group G.

- **Stage 2:** Suppose that \mathcal{A} in this stage is the Type II adversary \mathcal{A}_{II} and \mathcal{B} is given (p, P, aP, bP) as an instance of the CDH problem. In order to solve the problem by using of \mathcal{A}_{II} , \mathcal{B} needs to simulate a challenger to execute each phase of IND-CCA game for \mathcal{A}_{II} as follows:
 - Setup: \mathcal{B} picks $x \in \mathbb{Z}_p^*$ at randomly and computes X = xP, where x is the master key, then gives \mathcal{A}_{II} (p, P, X, H_1, H_2, H_3) as param, where H_1, H_2, H_3 are random oracles. Adversary \mathcal{A}_{II} may make queries of all random oracles at any-time during its attacks as Stage 1.

Phase 1: A_{II} can issue the following oracle queries. **Public-Key-Request:** On receiving a query ID:

- 1) If $(ID, (\omega, Y), coin)$ exists in **PublicK-eyList**, return $PK_{ID} = (\omega, Y)$ as answer.
- 2) Otherwise, pick $coin \in \{0, 1\}$ at random, so that $\Pr[coin = 0] = \delta$. (δ is the same as it in the proof of **Stage 1**.)
- 3) If coin = 0, pick $y, s \in Z_p^*$ at random and compute Y = yP, $\omega = sP$ and $d = s + xH_1(\mathrm{ID}||\omega||pk)$, add $\langle \mathrm{ID}, (y, d) \rangle$ to **PrivateKeyList** and $\langle \mathrm{ID}, (\omega, Y), coin \rangle$ to **PublicKeyList**, return $PK_{ID} = (\omega, Y)$ as answer.
- 4) Otherwise (if coin = 1), pick $y, s \in Z_p^*$ at random and compute Y = yP, $\omega = s(bP)$, add $\langle ID, (y, *) \rangle$ to **PrivateKeyList** and $\langle ID, (\omega, Y), coin \rangle$ to **PublicKeyList**, return $PK_{ID} = (\omega, Y)$ as answer.

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

- 1) Run **Public-Key-Request** on ID to get a tuple $(ID, (\omega, Y), coin) \in PublicKeyList.$
- 2) If coin = 0, search **PrivateKeyList** for a tuple $\langle ID, (y, d) \rangle$ and return $SK_{ID} = (y, d)$ as answer.
- 3) Otherwise, return "Abort" and terminate.
- **Decryption:** On receiving a query (ID, PK_{ID} , C), where $C = (c_1, c_2)$ and $PK_{ID} = (\omega, Y)$:
 - 1) Search **PublicKeyList** for a tuple $\langle \text{ID}, (\omega, Y), coin \rangle$. If coin = 0, search **PrivateKeyList** for a tuple $\langle \text{ID}, (y, d) \rangle$ (Note that $\langle \text{ID}, (\omega, Y), coin \rangle$ must exist in **PublicKeyList** and when coin = 0, $\langle \text{ID}, (y, d) \rangle$ exist in **PrivateKeyList**). Then, set $SK_{ID} = (y, d)$ and run **Decrypt**. Finally, return the results of **Decrypt** algorithm.
 - 2) Otherwise (if coin = 1), run H_1 queries to get a tuple $\langle (\text{ID}, \omega, Y), e \rangle$. If there exist $\langle (M, \sigma), r \rangle \in L_2$ and $\langle (z_1, z_2), R \rangle \in L_3$ such that $c_1 = rQ_{ID}, c_2 = R \oplus (M||\sigma), z_1 = rY,$ $z_2 = Q_{ID}$, return M and "Reject" otherwise.
- **Challenge:** A_{II} then outputs two messages (M_0, M_1) and a challenge identity ID^{*}. On receiving a challenge query $(\text{ID}^*, (M_0, M_1))$:
 - 1) \mathcal{B} runs **Public-Key-Request** taking ID^* as input to get a tuple $\langle \mathrm{ID}^*, (\omega^*, Y^*), coin \rangle \in \mathbf{PublicKeyList}.$
 - 2) If coin = 0, return "Abort" and terminate.
 - 3) Otherwise, do the following,
 - a. Search **PrivateKeyList** for a tuple $\langle \text{ID}^*, (y^*, *), s^* \rangle$.(In this case, we know that $Y^* = y^*P, \omega^* = s^*bP$.)
 - b. Pick $\sigma^* \in \{0, 1\}^{l_1}, c_2^* \in \{0, 1\}^l$ and $\beta \in \{0, 1\}$ at random.

Schemes	Enc	Dec	Sec-Lev	Sec-Ass
[2]	3P+1S+1E	1P+1S	IND-CCA	GBDHP
[6]	1P+2S+1E	1P+2S	IND-CCA	BDH
[25]	3S+1E	1P+3S	IND-CCA	K-BDHI
[8]	1P+3S+1E	4P	Strong Type I/II	3-DDH
[26]	2P+2S+2E	2P+1S	IND-CCAII	BDH
[3]	$4\mathrm{E}$	3E	IND-CCA	CDH
[12]	2S+3E	2E	IND-CCA	CDH
[27]	6E	3E	IND-CCA	CDH
Ours	4S	3S	IND-CCA	CDH

Table 2: Comparison of the related schemes

c. Set $c_1^* = aQ_{ID^*}$, $e^* = H_1(\text{ID}^* || \omega^* || Y^*)$. d. Define $a = H_2(M_\beta || \sigma^*)$ and $a = H_3(aY^* || \omega^* + XH_1(\text{ID}^* || \omega^* || Y^*))$ $= c_2^* \oplus (M_\beta || \sigma^*)$. (Note that \mathcal{B} does not know "a".)

4) Return $C^* = (c_1^*, c_2^*)$ as a target ciphertext.

- **Phase 2:** \mathcal{B} repeats the same algorithms that it operated in **Phase 1** of **Stage 2**.
- **Guess:** \mathcal{A}_{II} outputs a guess β' . Now \mathcal{B} returns the set

 $S = \left\{ \frac{1}{s^*} (az_{2i} - (aP)xe^*) | z_{2i} \text{ is the second component of queries to } H_3 \text{ for } i \in [1, q_{H_3}] \text{ such that } e^* = H_1(\mathrm{ID}^* \| \omega^* \| Y^*) \right\}.$

Then, \mathcal{B} will be able to solve the CDH problem by picking $\frac{1}{s^*}(az_{2i} - (aP)xe^*)$ from S.

Analysis: Similar to Analysis in the proof of Stage 1.

Consequently, we obtain $\epsilon' > \frac{1}{q_{H_3}} \left(\frac{2\epsilon}{e(q_{prv}+1)} - \frac{q_{H_2}}{2^{l_1}} - \frac{q_{D}q_{H_2}}{p}\right)$. The running time of the CDH adversary \mathcal{B} is $t' > t + 2(q_{pub} + q_{prv})t_{sm} + 2q_Dq_{H_2}q_{H_3}t_{sm} + 3t_{sm}$.

To sum up the two stages above, we complete the proof of **Theorem 1**. \Box

5 Performance Analysis

In this subsection, we compare the proposed scheme with other existing CL-PKE schemes on the computation complexity of encryption(**Enc**), decryption(**Dec**), security level(**Sec-Lev**) and security assumption(**Sec-Ass**). Without considering the additional of two points and hash function in the cyclic additive group, each scheme has three major types of operation, i.e., Pairing(P), Scalar Multiplication(S) and Exponentiation(E). From Table 2, we can see that our scheme calculates four scalar multiplications in **Encrypt** and three scalar multiplications in **Decrypt**, which denotes that it needs a lower computation cost than others.

We simulate the cryptographic operations by using of MIRACL (version 5.6.1, [23]) on a laptop using the Intel Core i5-2400 at a frequency of 3.10 GHz with 3GB memory and a Windows XP operation system, and then obtain

the average running time in Table 3. For pairing-based schemes, considering to be implemented in practice efficiently, we use the Fast-Tate-Pairing in MIRACL, which is defined over the MNT curve E/F_q [18] with embedding degree 4 and q is a 160 bits prime, and its security level achieves the difficulty of discrete log problem in 640 bits. For ECC-based scheme, we employed the parameter secp192k1 [28], where $p = 2^{192} - 2^{32} - 2^{12} - 2^8 - 2^7 - 2^6 - 2^3 - 1$. Moreover, the length of an element in multiplication group is set to be 1024 bits.

Table 3: Cryptographic operations time (in millisecond)

Fast-Tate	Exponentiation	Scalar	
-Pairing		Multiplication	
2.65	3.91	0.78	

Based on these settings above, we can simulate the total running time of one round of **Encrypt-Decrypt** in different schemes respectively as shown in Table 4. As for energy consumption, it can be calculated as $W = U \times I \times t$ based on the execution time (t), the voltage (U) and current draw (I). Suppose that the voltage is 3v and the current draw is 8mA in sensor platform MICAz [16], the energy consumption of one round in every protocol is also demonstrated in Table 4. For instance, in Al-Riyami and Paterson's work [2], it needs 4 pairing operations, 2 scalar multiplications and 1 exponentiation altogether, then the total running time in MICAz is $4 \times 2.65 + 2 \times 0.78 + 1 \times 3.91 = 16.07$ ms, and the energy consumption is $3 \times 8 \times 16.07 = 385.68 \ \mu$ J.

The communication cost of different schemes are compared in terms of bandwidth of transmitted ciphertext. Assuming the output of one way Hash function is 192 bits and the symmetric cipher is 128 bits (such as AES). In our protocol and [25], each ciphertext contains one point and one Hash value, thus the bandwidths of our protocol and [25] are (192+192)/8=48 bytes respectively. In [2] and [6], each ciphertext contains one point and two Hash values, thus the bandwidths of [2] and [6] are $(192+192\times 2)/8=72$ bytes respectively. In [8], the ciphertext contains one pairing and three modular exponentiations, thus the bandwidth of [8] is $(160+1024\times3)/8=404$ bytes. In [26], the ciphertext contains one point and one symmetric cipher, thus the bandwidth of [26] is (192+128)/8=40 bytes. In [3], [12] and [27], each ciphertext contains one modular exponentiations and one Hash value, thus the bandwidths of [3], [12] and [27] are (1024+192)/8=152 bytes respectively. The detailed comparison results are listed in Table 4.

Table 4: Comparison of the CL-PKE Schemes

Schemes	Running	Energy	Bandwidth
	Time	Consump-	(byte)
	(ms)	tion $(\mu \mathbf{J})$	
[2]	16.07	385.68	72
[6]	12.33	295.92	72
[25]	11.24	269.76	48
[8]	19.5	468	404
[26]	20.76	498.24	40
[3]	27.37	656.88	152
[12]	21.11	506.64	152
[27]	35.19	844.56	152
Ours	5.46	131.04	48

To sum up, our protocol is more suitable to be applied in WSNs with the characteristics of low cost and low power sensor nodes that are small in size and communicate wirelessly with each other nodes in short distances.

6 Conclusions

In this paper, we propose a CL-PKE scheme that does not depend on the pairing and prove that the scheme is IND-CCA secure in the random oracle model, relative to the hardness of the CDH problem. Besides, this scheme can achieve the highest trust Level 3. The comparison and simulation in Section 5 illustrate that our proposed scheme is advantageous over the related schemes on computation cost, communication overhead and energy consumption. Due to the appealing properties, the proposal could be applied in WSNs.

Acknowledgments

This study was supported by the National Nature Science Foundation of China under grant NSFC 11501343, and the Open Foundation of State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under grant SKLNST-2016-2-11. The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

- I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, pp. 102–114, 2002.
- [2] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryp*tology (ASIACRYPT'03), LNCS 2894, pp. 452–473, Springer, 2003.
- [3] J. Baek, R. Safavi-Naini and W. Susilo, "Certificateless public key encryption without pairing," in 8th International Conference on Information Security, LNCS 3650, pp. 134–148, Springer, 2005.
- [4] J. Baek, H. C. Tan, J. Zhou and J. W. Wong, "Realizing stateful public key encryption in wireless sensor network," in *The IFIP TC-11 23rd International Information Security Conference*, vol. 278, pp. 95–107, 2008.
- [5] P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott, "Efficient algorithms for pairingbased cryptosystems," in *Advances in Cryptology* (*CRYPTO'02*), LNCS 2442, pp. 354–368, Springer, 2002.
- [6] Z. H. Cheng and R. Comley, "Efficient certificateless public key encryption," *IACR Cryptology ePrint Archive*, 2005. (http://eprint.iacr.org/ 2005/012.pdf)
- [7] C. K. Chu, J. K. Liu, J. Zhou, F. Bao and R. H. Deng, "Practical ID-based encryption for wireless sensor network," in *Proceedings of the 5th ACM Symposium* on Information, Computer and Communications Security, pp. 337–340, 2010.
- [8] A. W. Dent, B. Libert and K. G. Paterson, "Certificateless encryption schemes strongly secure in the standard model," in 11th International Conference on Theory and Practice in Public-Key Cryptography, LNCS 4939, pp. 344–359, Springer, 2008.
- [9] T. H. Feng, W. Li and M. S. Hwang, "A false data report filtering scheme in wireless sensor networks: A survey," *International Journal of Network Security*, vol. 17, pp. 229–236, 2015.
- [10] M. Girault, "Self-certified public keys," in Advances in Cryptology (EUROCRYPTO'91), vol. 547, pp. 490–497, Springer, 1992.
- [11] M. Kumar, K. Dutta, I. Chopra, "Impact of wormhole attack on data aggregation in hieracrchical WSN," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 70–77, 2014.
- [12] J. Lai, W. Kou and K. Chen, "Self-generatedcertificate public key encryption without pairing and its application," *Information Sciences*, vol. 181, pp. 2422–2435, 2011.
- [13] C. T. Li, M. S. Hwang, "A lightweight anonymous routing protocol without public key en/decryptions for wireless ad hoc networks", *Information Sciences*, vol. 181, no. 23, pp. 5333–5347, Dec. 2011.
- [14] G. Li, Q. Jiang, Y. Shi and F. Wei, "Anonymous network information acquirement protocol for mobile users in heterogeneous wireless networks," *In-*100 (2010) 100 (20

193-200, 2016.

- [15] B. Libert and J. Quisquater, "On constructing certificateless cryptosystems from identity based encryption," in 9th International Conference on Theory and Practice in Public-Key Cryptography, LNCS 3958, pp. 474–490, Springer, 2006.
- [16] A. Liu and P. Ning, "TinyECC: A configurable libraty for elliptic curve cryptography in wireless sensor networks," in 2008 International Conference on Information Processing in Sensor Networks, pp. 245– 256, 2008.
- [17] Y. Lu, X. Yang and X. Wu, "A secure anonymous authentication scheme for wireless communications using smart cards," International Journal of Network Security, vol. 17, pp. 237–245, 2015.
- [18] A. Miyaji, M. Nakavayashi and S. Takano, "New explicit conditions of elliptic curve traces for FRreduction," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E84-A, no. 5, pp. 1234–1243, 2001.
- [19] NIST, SkipJack and KEA Algorithm Specifications, National Institute of Standards and Technology, May 29, 1998. (http://csrc.nist.gov/groups/ ST/toolkit/documents/skipjack/skipjack.pdf)
- [20] L. B. Oliveira, R. Dahab, J. López, F. Daguana and A. A. F. Loureiro "Identity-based encryption for sensor networks," in The Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 290-294, 2007.
- [21] R. L. Rivest, M. J. B. Robshaw, R. Sidney and Y. L. Yin, The RC6 Block Cipher v1.1, Aug. 20, 1998. (https://people.csail.mit.edu/ rivest/pubs/RRSY98.pdf)
- [22] C. P. Schnorr, "Efficient identifications and signature for smart cards," in Advances in Cryptology (CRYPTO'89), LNCS 435, pp. 239–251, Springer, 1990.
- [23] M. Scott, Miracl Library, Sept. 4, 2017. (http:// certivox.com/)
- "Identity-based cryptosystems and [24] A. Shamir, signature schemes," in Advances in Cryptology (CRYPTO'84), LNCS 196, pp. 47–53, Springer, 1985.
- [25] Y. Shi and J. Li, "Provable efficient certificateless public key encryption," IACR Cryptology ePrint 2005. (http://eprint.iacr.org/2005/ Archive, 287.pdf)

- ternational Journal of Network Security, vol. 18, pp. [26] Y. Sun and H. Li, "Short-ciphertext and BDHbased CCA2 secure certificateless encryption," Science China: Information Sciences, vol. 53, pp. 2005-2015, 2010.
 - [27] Y. Sun, F. Zhang and J. Baek, "Strongly secure certificateless public key encryption without pairing," in 6th International Conference on Cryptology and Network Security, vol. 4856, pp. 194-208, 2007.
 - [28]The Certicom Research, SEC2: Recommended Elliptic Curve Domain Parameters, Version 2.0, Jan. 27, 2010. (http://www.secg.org/sec2-v2.pdf)
 - [29] R. J. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn and P. Kruus, "TinyPK: Securing sensor networks with public key technology," in 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks, pp. 59-64, 2004.
 - D. H. Yum and P. J. Lee, "Generic construction [30]of certificateless encryption," International Conference on Computational Science and Its Applications (ICCSA'04), LNCS 3040, pp. 802-811, Springer, 2004.
 - [31] ZigBee Alliance, ZigBee Specifications, Document 053474r17, 2008.

Biography

Rui Guo received the Ph.D degrees in the Department of State Key Laboratory of Networking and Switch Technology, Beijing University of Posts and Telecommunications in 2014. Now, he is a lecturer in National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications. His present research interests include cryptography, information security and WSN.

Huixian Shi received the B.S. and Ph.D degrees in Department of Mathematics and Information Science from Shaanxi Normal University, Xi'an, China, in 2007 and 2013, respectively. Now she is a associate professor in Department of Department of Mathematics and Information Science in Shaanxi normal University. Her present research interests include model checking, fuzzy logic and uncertainty reasoning.