International Journal of Innovative Computing, Information and Control
Volume **9**, Number **11**, November **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 4509–4518

CONSTRUCTING PAIRING-FREE CERTIFICATE-BASED ENCRYPTION

Yang Lu* and Jiguo Li

College of Computer and Information Engineering
Hohai University
No. 8, Focheng Xi Rd., Jiangning Dist., Nanjing 210098, P. R. China
*Corresponding author: luyangnsd@163.com; lijiguo@163.com

Received October 2012; revised February 2013

Abstract. *Certificate-based cryptography is a new paradigm that combines traditional public-key cryptography and identity-based cryptography. It not only simplifies the cumbersome certificate management in traditional public-key cryptography, but also eliminates the key escrow and distribution problems inherent in identity-based cryptography. However, all constructions of certificate-based encryption in the literature so far have to be based on the costly bilinear pairings. Therefore, the previous certificate-based encryption schemes are too expensive computationally to be employed in the computation-limited mobile wireless networks. In this paper, we propose a certificate-based encryption scheme that dose not depend on the bilinear pairings. The proposed scheme is proved to be chosen-ciphertext secure in the random oracle model under the hardness of the RSA problem and the computational Diffie-Hellman problem. Due to avoiding the computationally-heavy bilinear paring operations, the proposed scheme significantly reduces the computational cost and outperforms all the previous certificate-based encryption schemes. This interesting property makes it particularly suitable for the resource-limited mobile devices.*

Keywords: Certificate-based encryption, Chosen-ciphertext security, Bilinear pairing, Random oracle model, Mobile wireless network

1. **Introduction.** In traditional PKC, the cryptographic keys are generated randomly with no connection to the users' identities. It is infeasible to prove that an entity is indeed the owner of a given public key. The usual approach to ensure the authenticity of a public key is to use a certificate. However, the need for public-key certificates is considered as the main difficulty in the deployment of traditional PKC. In 1984, Shamir [1] introduced the notion of identity-based cryptography (IBC). In IBC, each user's public key is derived directly from his identity, such as an IP address or an e-mail address, and his private key is generated by a trusted third party called Private Key Generator (PKG). The main practical benefit of IBC lies in the reduction of the need for public key certificates. However, if the PKG becomes dishonest, it can impersonate any user using its knowledge of the user's private key. This is due to the key escrow problem inherent in IBC. In addition, as private keys must be sent to the users over secure channels, private key distribution in IBC is a very daunting task.

In Eurocrypt 2003, Gentry [2] introduced a new paradigm called certificate-based cryptography (CBC) which represents an interesting and potentially useful balance between traditional PKC and IBC. As in traditional PKC, each user in CBC generates his own public key and private key pair and then requests a certificate from a trusted third party called certifier. The difference is that a certificate in CBC acts not only as a certificate (as in traditional PKC) but also as a private key (as in IBC). This additional functionality provides an efficient implicit certificate mechanism. The feature of implicit certificate allows us to eliminate the third-party queries for the certificate status and simplify the certificate revocation in traditional PKI. As a result, CBC does not need some traditional infrastructures like certificate revocation list (CRL) and online certificate status protocol (OCSP). Furthermore, CBC eliminates the key escrow and key distribution problems inherent in IBC. Since its advent, CBC has attracted much attention in the research community and a number of schemes have been proposed, including many encryption schemes (e.g., [3-10]) and signature schemes (e.g., [11-15]).

Our Motivations and Contributions. In the recent years, there has been an unprecedented growth in mobile wireless networks. Mobile wireless networks have been deployed for a wide variety of applications. Although mobile applications deployed on mobile devices have received significant attention, the security issue will be an important factor for their full adoption. Compared with wired networks, mobile wireless networks are more vulnerable to various attacks due to their nature of wireless communication. In some mobile applications, providing confidentiality for sensitive data is of prime importance. However, as mobile devices in a mobile wireless network usually have very constrained resources and are unable to implement heavy cryptographic algorithms, providing data confidentiality in mobile wireless networks poses more challenges than traditional network security. For this reason, only the high efficient and power-saving cryptographic algorithms can be used to provide mobile wireless networks with security.

The motivation of this paper is to develop a practical certificate-based encryption (CBE) scheme for mobile wireless networks. As introduced above, CBE is a novel public-key cryptographic paradigm with many attractive features and is very suitable for providing data confidentiality services in mobile wireless networks. However, as far as we know, the CBE schemes in the literature so far have to be based on the computationally-heavy bilinear pairings. In spite of the recent advances in the implementation technique [16,17], the bilinear pairing operation is still regarded as the heaviest time-consuming one compared with other operations such as the prime modular exponentiations in the finite fields. Thus, the bilinear pairing operations will greatly aggravate the computation load of a device, which is extremely disliked by the power-constrained mobile wireless networks. Therefore, the existing CBE schemes are too expensive computationally to be employed in mobile wireless networks.

Being aware of the above problem of the current constructions of CBE, we propose a CBE scheme that does not depend on the bilinear pairings in this paper. Our CBE scheme is motivated from the RSA-based key agreement protocol proposed by Okamoto and Tanaka [18], and is proved to be chosen-ciphertext secure under the hardness of RSA problem and the computational Diffie-Hellman problem in the random oracle model [19]. Without bilinear pairing, our scheme is more efficient than all of the CBE schemes proposed so far. This distinct and interesting property makes our scheme particularly suitable to be employed in the computation-limited mobile wireless networks.

2. Notations and Hard Mathematical Problems.

For a positive integer n, let Z_n denote the set $\{0, 1, 2, \ldots, n-1\}$, Z_n^* denote $Z_n \backslash \{0\}$ and Z_n^{odd} denote the set of the odd numbers from Z_n.

The security of our CBE scheme is based on the hardness of the following RSA problem and the computational Diffie-Hellman (CDH) problem.

Definition 2.1. *The RSA problem in Z_n^* is, given (n, e, b), where $n = pq$ such that p, q, $(p-1)/2$ and $(q-1)/2$ are large prime numbers, e is an odd integer such that $gcd(e, \phi(n)) = 1$ and b is a random integer from Z_n^*, to find $a \in Z_n^*$ such that $b^e = a(\mathrm{mod} n)$.*

Definition 2.2. *The CDH problem in* Z_n^* *is, given* p, q, n, $(g, g^a, g^b) \in Z_n^*$, *where* $n = pq$ *such that* p, q, $(p-1)/2$ *and* $(q-1)/2$ *are large prime numbers, and* $a, b \in Z_n^{odd}$, *to compute* $g^{ab}(\mathrm{mod} n)$.

3. **Scheme Definition and Security Model of Certificate-Based Encryption.** Formally, a CBE scheme is specified by five algorithms (***Setup***, ***KeyPairGen***, ***Certify***, ***Encrypt***, ***Decrypt***) such that:

(1) ***Setup***: Taking a security parameter 1^k as input, the certifier runs this algorithm to generate a master secret key *msk* and a list of public parameters *params*. After running this algorithm, the certifier publishes *params* and keeps *msk* secret.

(2) ***KeyPairGen***: Taking *params* as input, a user with identity *id* runs this algorithm to generate a private key SK_{id} and a partial public key PPK_{id}.

(3) ***Certify***: Taking *params*, *msk*, *id* and PPK_{id} as input, the certifier runs this algorithm to generate a certificate $Cert_{id}$ and a public key PK_{id}. The user *id* should combine $Cert_{id}$ and SK_{id} as his decryption key to decrypt the ciphertext sent to him.

(4) ***Encrypt***: Taking *params*, *id*, PK_{id} and a message M as input, the user as a sender runs this algorithm to create a ciphertext C.

(5) ***Decrypt***: Taking *params*, SK_{id}, $Cert_{id}$, a ciphertext C and optionally PK_{id} as input, the user as a receiver runs this algorithm to get a decryption δ, which is either a plaintext message M or a special symbol $\perp$ indicating a decryption failure.

Our definition of CBE is quite different from the previous ones [2,3]. The main difference is that each user in our definition should first generate a partial public key and then authenticate himself to the certifier to create his full public key, while each user in the previous definitions generates his public key independently. It seems that our definition of CBE is slightly weaker than previous ones. However, we note that the reason why the CBE schemes in the literature so far have to depend on some known pairing-based identity-based encryption (IBE) schemes is that in those schemes, a user need not be certified before generating a public key, which is indeed a feature provided by IBE. By relaxing this requirement, we could construct a very efficient pair-free CBE scheme which does not depend on any existing IBE schemes. Most importantly, our new definition of CBE does not lose the most attractive features of CBC, such as no key-escrow problem, implicit certificates.

The following security model of CBE is modified from the one proposed by Al-Riyami and Paterson [3]. It is defined by two different adversarial games.

Game-I: This game is played between a Type-I adversary A_I and a game challenger.

Setup. The challenger runs the algorithm ***Setup***(1^k) to generate a master secret key *msk* and a list of public parameters *params*. It then returns *params* to A_I and keeps *msk* to itself.

Phase 1. In this phase, A_I can adaptively query the following oracles:

(1) **CreateUser**(*id*): On input an identity *id*, if *id* has already been created, the challenger responds with the partial public key PPK_{id} associated with the identity *id*. Otherwise, the challenger should first generate a set of private key SK_{id}, partial public key PPK_{id}, public key PK_{id} and certificate $Cert_{id}$ for the identity *id*, and then output PPK_{id} to A_I. In this case, *id* is said to be created. We assume that other oracles defined below only respond to an identity which has been created.

(2) **RequestPublicKey**(*id*): On input an identity *id*, the challenger responds with the public key PK_{id} associated with the identity *id*.

(3) **RequestCertificate**(*id*): On input an identity *id*, the challenger responds with the certificate $Cert_{id}$ associated with the identity *id*.

(4) **ExtractPrivateKey**(id): On input an identity id, the challenger responds the private key SK_{id} associated with the identity id.

(5) **Decrypt**(id, C): On input an identity id and a ciphertext C, the challenger responds with the decryption of the ciphertext C.

Challenge. A_I outputs (id^*, M_0, M_1) on which it wants to be challenged, where M_0 and M_1 are two equal length messages. The challenger randomly chooses $b \in \{0, 1\}$ and computes $C^* = \boldsymbol{Encrypt}(params, id^*, PK_{id^*}, M_b)$. It then outputs C^* as the challenge ciphertext to A_I.

Phase 2. In this phase, A_I issues a second sequence of queries as in Phase 1, but with the following restrictions: (1) A_I cannot make query **RequestCertificate**(id^*); (2) A_I cannot make query **Decrypt**(id^*, C^*).

Guess. A_I outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$. A_I's advantage is defined to be $Adv(A_I) = 2|\Pr[b = b'] - 1/2|$.

Game-II: This game is played between a Type-II adversary A_{II} and a game challenger.

Setup. The challenger runs the algorithm $\boldsymbol{Setup}(1^k)$ to generate a master secret key msk and a list of public parameters $params$, and then returns $params$ and msk to A_{II}.

Phase 1. In this phase, A_{II} can adaptively query the following oracles:

(1) **CreateUser**(id): On input an identity id, if id has already been created, the challenger responds with the partial public key PPK_{id} associated with the identity id. Otherwise, the challenger should first generate a set of private key SK_{id} and partial public key PPK_{id}, and then output PPK_{id} to A_{II}. In this case, id is said to be created. We assume that other oracles defined below only respond to an identity which has been created.

(2) **ExtractPrivateKey**(id): On input an identity id, the challenger responds the private key SK_{id} associated with the identity id.

(3) **Decrypt**(id, $Cert_{id}$, C): On input an identity id, a certificate $Cert_{id}$ and a ciphertext C, the challenger responds with the decryption of the ciphertext C.

Challenge. A_{II} outputs (id^*, PK_{id^*}, M_0, M_1) on which it wants to be challenged. The challenger randomly chooses $b \in \{0, 1\}$ and computes $C^* = \boldsymbol{Encrypt}(params, id^*, PK_{id^*}, M_b)$. It then outputs C^* as the challenge ciphertext to A_{II}.

Phase 2. In this phase, A_{II} issues a second sequence of queries as in Phase 1, but with the restrictions: (1) A_{II} cannot make query **ExtractPrivateKey**(id^*); (2) A_{II} cannot make query **Decrypt**(id^*, $Cert_{id^*}$, C^*).

Guess. A_{II} outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$. A_{II}'s advantage is defined to be $Adv(A_{II}) = 2|\Pr[b = b'] - 1/2|$.

Definition 3.1. *A CBE scheme is said to be secure against adaptive chosen-ciphertext attacks (or IND-CBE-CCA2 secure) if no PPT adversary has non-negligible advantage in both Game-I and Game-II.*

4. The Proposed CBE Scheme.

Our scheme is motivated from the RSA-based key agreement protocol introduced by Okamoto and Tanaka [18]. It consists of the following five algorithms.

(1) $\boldsymbol{Setup}$: The certifier first generates two primes p and q such that $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are k-bit prime numbers. It then computes $n = pq$ and the Euler totient function $\phi(n) = (p - 1)(q - 1)$. Additionally, it chooses four cryptographic hash functions H_1: $\{0, 1\}^* \to Z_n^*$, H_2: $\{0, 1\}^* \times Z_n^* \times Z_n^* \to Z_n^{odd}$, H_3: $\{0, 1\}^{l_m} \times \{0, 1\}^{l_r} \times \{0, 1\}^* \times Z_n^* \times Z_n^* \to Z_n^*$ and H_4: $Z_n^* \times Z_n^* \to \{0, 1\}^{l_m + l_r}$, where l_m denotes the bit-length of the plaintext and l_r denotes the bit-length of the random value

used in the encryption algorithm. Finally, the certifier sets $params = \{n, H_1, H_2, H_3, H_4\}$ as the public system parameters and $msk = \phi(n)$ as its master secret key.

(2) ***KeyPairGen***: A user with identity id chooses a random value $x \in Z_n^*$ as his private key SK_{id} and computes his partial public key $PPK_{id} = H_1(id)^x$.

(3) ***Certify***: To generate a certificate and a public key for a user with identity id, the certifier performs as follows: Choose a random value $y \in Z_n^*$ and set $PK_{id} = (PK_{id}^{(1)}, PK_{id}^{(2)}) = (PPK_{id}, H_1(id)^y)$; Compute $e = H_2(id, PK_{id}^{(1)}, PK_{id}^{(2)})$ and d such that $ed \equiv 1 \pmod{\phi(n)}$; Set $Cert_{id} = y + d \pmod{\phi(n)}$.

(4) ***Encrypt***: To send a message $M \in \{0,1\}^*$ to a receiver with identity id and public key $PK_{id} = (PK_{id}^{(1)}, PK_{id}^{(2)})$, the sender performs as follows: choose a random value $\sigma \in \{0,1\}^{l_r}$ and compute $r = H_3(M, \sigma, id, PK_{id}^{(1)}, PK_{id}^{(2)})$; compute $k_1 = (PK_{id}^{(1)})^{er}$ and $k_2 = (PK_{id}^{(2)})^{er}$; compute $U = H_1(id)^r$ and $V = (M||\sigma) \oplus H_4(k_1, k_2)$; set $C = (U, V)$ as the ciphertext.

(5) ***Decrypt***: To decrypt a ciphertext $C = (C_1, C_2, C_3)$, the receiver id parses the ciphertext C as (U, V) and computes $M'||\sigma' = V \oplus H_4(U^{SK_{id}\cdot e}, U^{Cert_{id}\cdot e}/U)$. It then checks whether $U = H_1(id)^{H_3(M', \sigma', id, PK_{id}^{(1)}, PK_{id}^{(2)})}$ holds. If it does, output M'; otherwise output $\perp$.

The consistency of the above scheme is easy to check as we have

$$U^{SK_{id}\cdot e} = (H_1(id)^r)^{xe} = (PK_{id}^{(1)})^{er},$$

$$\frac{U^{Cert_{id}\cdot e}}{U} = \frac{(H_1(id)^r)^{(y+d)e}}{H_1(id)^r} \overset{de\equiv 1(\mathrm{mod}\phi(n))}{=} \frac{(H_1(id)^y)^{er} H_1(id)^r}{H_1(id)^r} = (PK_{id}^{(2)})^{er}.$$

5. **Security Proof.** In this section, we prove in the random oracle that our CBE scheme achieves IND-CBE-CCA2 security under the hardness of the RSA problem and the CDH problem.

Theorem 5.1. *The proposed CBE scheme is IND-CBE-CCA2 secure in the random oracle model, assuming that the RSA problem and the CDH problem are both intractable.*

This theorem can be proved by combining the following two lemmas.

Lemma 5.1. *Suppose that $H_1 \sim H_4$ are random oracles and A_I is a Type-I adversary against the IND-CBE-CCA2 security of our CBE scheme with advantage ε when running in time τ, making q_{cu}* ***CreateUser*** *queries, q_{pub}* ***RequestPublicKey*** *queries, q_{pri}* ***ExtractPrivateKey*** *queries, q_{cer}* ***RequestCertificate*** *queries, q_{dec}* ***Decrypt*** *queries and q_i random oracle queries to H_i $(1 \le i \le 4)$. Then there exists an algorithm A_{RSA} to solve the RSA problem in Z_n^* with advantage $\varepsilon' \ge \varepsilon/q_{cu}$ and running time $\tau' \le \tau + (q_1 + q_{dec})(3\tau_{\exp} + O(1)) + (q_2 + q_3 + q_4 + q_{cu} + q_{pub} + q_{cer} + q_{pri})O(1)$, where $\tau_{\exp}$ denotes the time for computing a modular exponentiation in Z_n^*.*

Proof: Assume that A_{RSA} is given a random instance (n, e, b) of the RSA problem. Its goal is to find $a \in Z_n^*$ such that $a^e = b \pmod n$ by interacting with A_I as follows:

In the setup phase, A_{RSA} randomly chooses an index I with $1 \le I \le q_{cu}$ and simulates the setup algorithm by supplying A_I with $params = \{n, H_1, H_2, H_3, H_4\}$, where $H_1 \sim H_4$ are random oracles controlled by A_{RSA}. A_I can make queries to these random oracles at any time during the game. Note that the corresponding master key is the factors of n, namely p and q which are unknown to A_{RSA}. A_{RSA} responds A_I's various queries as follows:

***H*$_1$-queries**: A_{RSA} maintains a list $\mathbf{H_1List}$ of tuples $<id_i, e_i, h_{1,i}>$. On receiving such a query on id_i, A_{RSA} responds as follows. (1) If id_i already appears on $\mathbf{H_1List}$ in a tuple $<id_i, e_i, h_{1,i}>$, then A_{RSA} returns $h_{1,i}$ to A_I. (2) Else, if $i = I$, then A_{RSA} does the following: Randomly choose $\alpha \in Z_n^{odd}$, $\gamma_I \in Z_n^*$ and set $e_I = e$, $h_{1,I} = \gamma_I^{\alpha e^2}$. Randomly choose $x_I \in Z_n^*$ and set $SK_{id_I} = x_I$, $PK_{id_I}^{(1)} = (h_{1,I})^{x_I}$ and $PK_{id_I}^{(2)} = \gamma_I$. Insert $<id_I, e_I, h_{1,I}>$, $<id_I, PK_{id_I}^{(1)}, PK_{id_I}^{(2)}, e_I>$ and $<id_I, SK_{id_I}, PK_{id_I}^{(1)}, PK_{id_I}^{(2)}, ->$ into $\mathbf{H}_1$**List**, $\mathbf{H}_2$**List** and **UserList** respectively. Return $h_{1,I}$ to A_I. Note that the certificate of the identity id_I is unknown to A_{RSA}. (3) Otherwise, A_{RSA} does the following: Randomly choose $e_i \in Z_n^{odd}$, $\gamma_i \in Z_n^*$ and set $h_{1,i} = \gamma_i^{e_i}$. Randomly choose $x_i \in Z_n^*$, $s_i \in Z_n^{odd}$ and set $SK_{id_i} = x_i$, $PK_{id}^{(1)} = (h_{1,i})^{x_i}$, $PK_{id}^{(2)} = (h_{1,i})^{s_i} \cdot \gamma_i^{-1}$ and $Cert_{id_i} = s_i$. Insert $<id_i, e_i, h_{1,i}>$, $<id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, e_i>$ and $<id_i, SK_{id_i}, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, Cert_{id_i}>$ into $\mathbf{H}_1$**List**, $\mathbf{H}_2$**List** and **UserList** respectively. Return $h_{1,i}$ to A_I. It is easy to verify that $(PK_{id_i}^{(2)})^{e_i} \cdot h_{1,i} = (h_{1,i})^{Cert_{id_i} \cdot e_i}$. Therefore, $\{SK_{id_i}, PK_{id_i} = (PK_{id_i}^{(1)}, PK_{id_i}^{(2)}), Cert_{id_i}\}$ is a consistent set of private key, public key and certificate values for the identity id_i.

$\boldsymbol{H}_2$**-queries**: A_{RSA} maintains a list $\mathbf{H}_2$**List** of tuples $<id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, e_i>$. On receiving such a query on $(id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)})$, A_{RSA} retrieves a tuple of the form $<id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, e_i>$ from $\mathbf{H}_2$**List** and returns e_i to A_I.

$\boldsymbol{H}_3$**-queries**: A_{RSA} maintains a list $\mathbf{H}_3$**List** of tuples $< M, \sigma, id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, r >$. On receiving such a query on $(M, \sigma, id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)})$, A_{RSA} responds as follows. (1) If $(M, \sigma, id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)})$ already appears on $\mathbf{H}_3$**List** in a tuple $< M, \sigma, id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, r >$, it returns r to A_I. (2) Otherwise, it returns a random $r \in Z_n^*$ to A_I and inserts $< M, \sigma, id_i, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, r >$ into $\mathbf{H}_3$**List**.

$\boldsymbol{H}_4$**-queries**: A_{RSA} maintains a list $\mathbf{H}_4$**List** of tuples $< k_1, k_2, h_4 >$. On receiving such a query on (k_1, k_2), A_{RSA} responds as follows. (1) If (k_1, k_2) already appears on $\mathbf{H}_4$**List** in a tuple $< k_1, k_2, h_4 >$, it returns h_4 to A_I. (2) Otherwise, it returns a random $h_4 \in \{0,1\}^{l_m + l_r}$ to A_I and inserts $< k_1, k_2, h_4 >$ into $\mathbf{H}_4$**List**.

CreateUser: A_{RSA} maintains a list **UserList** of tuples $<id_i, SK_{id_i}, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, Cert_{id_i}>$. On receiving such a query on id_i, if id_i already appears on **UserList** in a tuple $<id_i, SK_{id_i}, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, Cert_{id_i}>$, A_{RSA} returns $PK_{id_i}^{(1)}$ to A_I. Otherwise, it should first query $H_1(id_i)$ to generate a set of private key, partial public key, public key and certificate for the identity id_i.

RequestPublicKey: On receiving such a query on id_i, A_{RSA} retrieves a tuple of the form $<id_i, SK_{id_i}, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, Cert_{id_i}>$ from **UserList** and returns $PK_{id_i} = (PK_{id_i}^{(1)}, PK_{id_i}^{(2)})$ to A_I.

ExtractPrivateKey: On receiving such a query on id_i, A_{RSA} retrieves a tuple of the form $<id_i, SK_{id_i}, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, Cert_{id_i}>$ from **UserList** and returns $SK_{id_i}$to A_I.

RequestCertificate: On receiving such a query on id_i, if $i = I$, A_{RSA} aborts. Otherwise, it retrieves a tuple of the form $<id_i, SK_{id_i}, PK_{id_i}^{(1)}, PK_{id_i}^{(2)}, Cert_{id_i}>$ from **UserList** and returns $Cert_{id_i}$ to A_I.

Decrypt: On receiving such a query on $(id_i, C = (U, V))$, A_{RSA} responds as follows. (1) If $i \neq I$, then A_{RSA} decrypts C in the normal way since it knows the private key SK_{id_i} and the certificate $Cert_{id_i}$ for the identity id_i. (2) Otherwise, A_{RSA} retrieves a tuple of the form $<id_I, e_I, h_{1,I}>$ from $\mathbf{H}_1$**List** and then searches in $\mathbf{H}_3$**List** for all tuples of the form $< M, \sigma, id_I, PK_{id_I}^{(1)}, PK_{id_I}^{(2)}, r >$. If no such tuple is found in $\mathbf{H}_3$**List**, then A_{RSA} returns an invalid symbol $\perp$. Otherwise, for each $< M, \sigma, id_I, PK_{id_I}^{(1)}, PK_{id_I}^{(2)}, r > \in$ $\mathbf{H}_3$**List**, A_{RSA} checks whether $(h_{1,I})^r = U$. If it holds, A_{RSA} computes $k_1 = (PK_{id_I}^{(1)})^{e_I r}$ and $k_2 = (PK_{id_I}^{(2)})^{e_I r}$, retrieves a tuple of the form $< k_1, k_2, h_4 >$ from $\mathbf{H}_4$**List**, and checks whether $M||\sigma = V \oplus h_4$. If it holds, A_{RSA} returns M to A_I as the decryption of C. If no tuple in $\mathbf{H}_3\mathbf{List}$ passes the above verifications, A_{RSA} returns an invalid symbol $\perp$.

At the challenge phase, A_I outputs two messages M_0 and M_1 of equal length and an identity id^*. If $id^* \neq id_I$, then A_{RSA} aborts. Otherwise, A_{RSA} sets $U^* = b^\alpha$ where α is the value obtained during the H_1 query corresponding to id_I, chooses a random $V^* \in \{0,1\}^{l_m+l_r}$, and then returns $C^* = (U^*, V^*)$ to A_I as the challenge ciphertext. Observe that the decryption of C^* is $V^* \oplus H_4((U^*)^{SK_{id_I}\cdot e_I}, \frac{(U^*)^{Cert_{id}\cdot e_I}}{U^*})$.

At the guess phase, A_I outputs a bit which is ignored by A_{RSA}. It is clear that A_I cannot recognize that C^* is not a valid ciphertext unless it queries H_4 on $((U^*)^{SK_{id_I}\cdot e_I}$, $\frac{(U^*)^{Cert_{id}\cdot e_I}}{U^*})$. Standard arguments can show that a successful A_I is very likely to make such a query if the simulation is indistinguishable from a real attack environment. To produce a result, A_{RSA} checks whether $k_2^e = b$ for each tuple $< k_1,\ k_2,\ h_4 >$ in $\mathbf{H}_4\mathbf{List}$, and outputs the k_2 value in the tuple passing the above test as the solution for the given RSA problem instance. We show that the k_2 value obtained as above is indeed a such that $a^e = b \pmod n$. Recall that the second part of the public key corresponding to id_I is set to be $PK_{id_I}^{(2)} = \gamma_I$. Since $H_1(id_I) = h_{1,I} = \gamma_I^{\alpha e^2}$, we get $\gamma_I = (H_1(id_I))^{\alpha^{-1}e^{-2}} = (H_1(id_I))^{\alpha^{-1}d^2}$ (because $d \equiv e^{-1} \pmod{\phi(n)}$) and thus $Cert_{id_I} = \alpha^{-1}d^2 + d$. Therefore, we have

$$\frac{(U^*)^{Cert_{id_I}\cdot e_I}}{U^*} = \frac{(b^\alpha)^{(\alpha^{-1}d^2+d)e}}{b^\alpha} = b^d = a \ \ (\text{since } d \equiv e^{-1} \ (\text{mod}\phi(n))).$$

We now derive the advantage of A_{RSA} in solving the given RSA problem. From the above construction, the simulation fails if any of the following events occurs: (1) E_1: A_I does not choose to be challenged on id_I; (2) E_2: A_I made a **RequestCertificate** query on id_I. We clearly have that $\Pr[\neg E_1] = 1/q_{cu}$ and $\neg E_1$ implies $\neg E_2$. Thus, we have that $\Pr[\neg E_1 \wedge \neg E_2] \geq 1/q_{cu}$. Therefore, the advantage of A_{RSA} is $\varepsilon' \geq \varepsilon/q_{cu}$.

This completes the proof of Lemma 5.1.

Lemma 5.2. *Suppose that $H_1\sim H_4$ are random oracles and A_{II} is a Type-II adversary against the IND-CBE-CCA2 security of our CBE scheme with advantage ε when running in time τ, making q_{cu} **CreateUser** queries, q_{pri} **ExtractPrivateKey** queries, q_{dec} **Decrypt** queries and q_i random oracle queries to H_i $(1 \leq i \leq 4)$. Then there exists an algorithm A_{CDH} to solve the CDH problem in Z_n^* with advantage $\varepsilon' \geq \varepsilon/(q_{cu}q_4)$ and running time $\tau' \leq \tau + (q_1+q_2+q_3+q_4+q_{pri})O(1) + q_{cu}\tau_{\exp} + O(1)) + q_{dec}(3\tau_{\exp} + O(1))$, where $\tau_{\exp}$ denotes the time for computing a modular exponentiation in Z_n^*.*

Proof: Assume that A_{CDH} is given a random instance $(p,\ q,\ n,\ g,\ g^a,\ g^b)$ of the CDH problem in Z_n^*. Its goal is to find $g^{ab} \pmod n$ by interacting with A_{II} as follows:

In the setup phase, A_{CDH} randomly chooses an index I with $1 \leq I \leq q_{cu}$ and simulates the setup algorithm by supplying A_{II} with $params = \{n,\ H_1,\ H_2,\ H_3,\ H_4\}$ and the master key $msk = (p-1)(q-1)$, where $H_1 \sim H_4$ are random oracles controlled by A_{CDH}. A_{II} can make queries to these random oracles at any time during the game. A_{CDH} responds A_{II}'s various queries as follows:

H$_1$**-queries**: A_{CDH} maintains a list $\mathbf{H}_1\mathbf{List}$ of tuples $<id_i,\ h_{1,i}>$. On receiving such a query on id_i, A_{CDH} responds as follows: (1) If a tuple of the form $<id_i,\ h_{1,i}>$ already exists in $\mathbf{H}_1\mathbf{List}$, it returns $h_{1,i}$ to A_{II}. (2) Else, if $i = I$, then it sets $h_{1,I} = g$, inserts $<id_I,\ h_{1,I}>$ into $\mathbf{H}_1\mathbf{List}$ and returns $h_{1,I}$ to A_{II}. (3) Otherwise, it returns a random $h_{1,i} \in Z_n^*$ to A_{II} and inserts $<id_i,\ h_{1,i}>$ into $\mathbf{H}_1\mathbf{List}$.

H$_2$**-queries**: A_{CDH} maintains a list $\mathbf{H}_2\mathbf{List}$ of tuples $<id_i,\ PK_{id_i}^{(1)},\ PK_{id_i}^{(2)},\ e_i>$. On receiving such a query on $(id_i,\ PK_{id_i}^{(1)},\ PK_{id_i}^{(2)})$, A_{CDH} responds as follows: (1) If a tuple of the form $<id_i,\ PK_{id_i}^{(1)},\ PK_{id_i}^{(2)},\ e_i>$ already exists in $\mathbf{H}_2\mathbf{List}$, it returns e_i to A_{II}. (2)

Otherwise, it returns a random $e_i \in Z_n^{odd}$ to A_{II} and inserts $<id_i,\ PK_{id_i}^{(1)},\ PK_{id_i}^{(2)},\ e_i>$ into $\mathbf{H}_2$**List**.

H$_3$**-queries**: A_{CDH} maintains a list $\mathbf{H}_3$**List** of tuples $< M,\ \sigma,\ id_i,\ PK_{id_i}^{(1)},\ PK_{id_i}^{(2)},\ r>$. On receiving such a query on $(M,\ \sigma,\ id_i,\ PK_{id_i}^{(1)},\ PK_{id_i}^{(2)})$, A_{CDH} responds as follows. (1) If a tuple of the form $< M,\ \sigma,\ id_i,\ PK_{id_i}^{(1)},\ PK_{id_i}^{(2)},\ r>$ already exists in $\mathbf{H}_3$**List**, it returns r to A_{II}. (2) Otherwise, it returns a random $r \in Z_n^*$ to A_{II} and inserts $< M,\ \sigma,\ id_i,\ PK_{id_i}^{(1)},\ PK_{id_i}^{(2)},\ r>$ into $\mathbf{H}_3$**List**.

H$_4$**-queries**: A_{CDH} maintains a list $\mathbf{H}_4$**List** of tuples $< k_1, k_2,\ h_4>$. On receiving such a query on (k_1, k_2), A_{CDH} responds as follows. (1) If a tuple of the form $< k_1,\ k_2,\ h_4>$ already exists in $\mathbf{H}_4$**List**, it returns h_4 to A_{II}. (2) Otherwise, it returns a random $h_4 \in \{0,1\}^{l_m+l_r}$ to A_{II} and inserts $< k_1,\ k_2,\ h_4>$ into $\mathbf{H}_4$**List**.

CreateUser: A_{CDH} maintains a list **UserList** of tuples $< id_i,\ SK_{id_i},\ PPK_{id_i}>$. On receiving such a query on id_i, A_{CDH} responds as follows. (1) If a tuple of the form $< id_i,\ SK_{id_i},\ PPK_{id_i}>$ already exists in **UserList**, it returns PPK_{id_i} to A_{II}. (2) Else, if $i = I$, then it sets $PPK_{id_I} = g^a$, inserts $< id_I, -, PPK_{id_I}>$ into **UserList** and returns PPK_{id_I} to A_{II}. (3) Otherwise, it randomly chooses $x_i \in Z_n^*$, sets $SK_{id_i} = x_i$ and $PPK_{id_i} = (h_{1,i})^{x_i}$, inserts $< id_i,\ SK_{id_i},\ PPK_{id_i}>$ into **UserList** and returns PPK_{id_i} to A_{II}.

ExtractPrivateKey: On receiving such a query on id_i, A_{CDH} responds as follows. (1) If $i = I$, then it aborts. (2) Otherwise, it retrieves a tuple of the form $< id_i,\ SK_{id_i},\ PPK_{id_i}>$ from **UserList** and returns SK_{id_i} to A_{II}.

Decrypt: On receiving such a query on $(id_i,\ Cert_{id_i},\ C = (U, V))$, A_{CDH} responds as follows. (1) If $i \neq I$, A_{CDH} decrypts the ciphertext C in the normal way since it knows the private key SK_{id_i} and A_{II} provides the certificate $Cert_{id_i}$ for the identity id_i. (2) Otherwise, A_{CDH} first retrieves a tuple of the form $< id_I,\ h_{1,I}>$ from $\mathbf{H}_1$**List**. It then searches in $\mathbf{H}_3$**List** for all tuples of the form $< M,\ \sigma,\ id_I,\ PK_{id_I}^{(1)},\ PK_{id_I}^{(2)},\ r>$. If no such tuple is found, it returns an invalid symbol $\perp$. Otherwise, for each $< M,\ \sigma,\ id_I,\ PK_{id_I}^{(1)},\ PK_{id_I}^{(2)},\ r> \in \mathbf{H}_3$**List**, A_{CDH} checks whether $(h_{1,I})^r = U$. If it holds, A_{CDH} computes $k_1 = (PK_{id_I}^{(1)})^{e_I r}$, $k_2 = (PK_{id_I}^{(2)})^{e_I r}$ and searches a tuple of the form $< k_1,\ k_2,\ h_4>$ in $\mathbf{H}_4$**List**. If such a tuple exists, A_{CDH} retrieves the h_4 value from the tuple and checks whether $M||\sigma = V \oplus h_4$. If it holds, A_{CDH} returns M to A_{II} as the decryption of C. If no tuple in $\mathbf{H}_3$**List** passes the above verifications, then A_{CDH} returns an invalid symbol $\perp$.

At the challenge phase, A_{II} outputs $< id^*,\ PK_{id^*},\ M_0,\ M_1>$ where $PK_{id^*} = (PK_{id^*}^{(1)},\ PK_{id^*}^{(2)})$. If $id^* \neq id_I$ and $PK_{id^*}^{(1)} \neq PPK_{id_I}$, then A_{CDH} aborts. Otherwise, A_{CDH} sets $U^* = g^b$, chooses a random $V^* \in \{0,1\}^{l_m+l_r}$, and then returns $C^* = (U^*, V^*)$ to A_{II} as the challenge ciphertext. Observe that the decryption of C^* is $V^* \oplus H_4((U^*)^{SK_{id_I} \cdot e_I},\ \frac{(U^*)^{Cert_{id_I} \cdot e_I}}{U^*})$.

At the guess phase, A_{II} outputs a bit which is ignored by A_{CDH}. It is clear that A_{II} cannot recognize that C^* is not a valid ciphertext unless it queries H_4 on $((U^*)^{SK_{id_I} \cdot e_I},\ \frac{(U^*)^{Cert_{id_I} \cdot e_I}}{U^*})$. Standard arguments can show that a successful A_{II} is very likely to make such a query if the simulation is indistinguishable from a real attack environment. To produce a result, A_{CDH} randomly chooses a tuple $< k_1,\ k_2,\ h_4>$ from $\mathbf{H}_4$**List** and outputs $k_1^{e_I^{-1}}$ as the solution for the given CDH problem. It is clear that $k_1^{e_I^{-1}} = g^{ab}$ if $k_1 = (U^*)^{SK_{id_I} \cdot e_I} = (g^{ab})^{e_I}$. Since $\mathbf{H}_4$**List** contains q_4 tuples, the chosen tuple will contain the correct k_1 value with probability $1/q_4$.

We now derive the advantage of A_{CDH} in solving the CDH problem. From the above construction, the simulation fails if any of the following events occurs: (1) E_1: A_{II} does not choose to be challenged on id_I; (2) E_2: A_{II} made a **ExtractPrivateKey** query on id_I. We clearly have that $\Pr[\neg E_1] = 1/q_{cu}$ and $\neg E_1$ implies $\neg E_2$. Thus, we have that $\Pr[\neg E_1 \wedge \neg E_2] \geq 1/q_{cu}$. Therefore, the advantage of A_{CDH} is $\varepsilon' \geq \varepsilon/(q_{cu}q_4)$.

This completes the proof of Lemma 5.2.

6. **Performance Comparison.** In this section, we make a comparison of our scheme with the previous CBE schemes. The details of the compared CBE schemes are listed in Table 1, where we compare the schemes on security model, computation efficiency and underlying hard problems. Note that we do not list all known CBE schemes in the literature but some secure and representative ones.

In the computation efficiency comparison, we consider three atomic operations: bilinear pairing, modular exponentiation and multiplication. For simplicity, we denote the computational cost of these operations by τ_p, τ_e and τ_m respectively. In addition, we denote the computation cost of a one-time signature signing and verification algorithms used in [6] by τ_s and τ_v respectively. As usual, some symmetric cryptographic operations (such as hash, message authentication code) are ignored as they can be computed efficiently. From the table, we can see that our scheme outperforms all the compared CBE schemes. Due to avoiding the computationally-heavy bilinear paring operations, our scheme is more suitable to be employed in the computation-limited mobile wireless networks.

TABLE 1. Comparison of the CBE schemes

Scheme	Without Random Oracles?	Encryption Cost	Decryption Cost	Underlying Hard Problems
Ours	$\times$	$3\tau_e$	$3\tau_e$	RSA + CDH
[2]	$\times$	$2\tau_p + \tau_m$	$\tau_p + \tau_m$	BDH
[8]	$\times$	$2\tau_e + 2\tau_m$	$\tau_p + \tau_e + \tau_m$	p-BDHI + 1-BDHI
[4]	$\surd$	$5\tau_m$	$3\tau_p + 3\tau_m$	DBDH
[6]	$\surd$	$5\tau_m + \tau_s$	$3\tau_p + 3\tau_m + \tau_v$	DBDH
[7]	$\surd$	$8\tau_e + 2\tau_m$	$2\tau_p + 2\tau_e + \tau_m$	q-ABDHE + DBDH

7. **Conclusions.** In this paper, we have presented a new CBE scheme that does not depend on the bilinear pairings. We have proved in the random oracle model that our scheme is chosen-ciphertext secure under the hardness of the RSA problem and the CDH problem. As our scheme does not require any costly bilinear pairing operation, it is particularly suitable to be employed in mobile wireless networks. However, a limitation of our scheme is that its security can only be achieved in the random oracle model. So, it is an interesting open problem to design a chosen-ciphertext secure CBE scheme without bilinear pairings in the standard model.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (No. 61272542) and the Fundamental Research Funds for the Central Universities of China (No. 2010B06414). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] A. Shamir, Identity-based cryptosystems and signature schemes, *Proc. of CRYPTO 1984*, Santa Barbara, CA, USA, pp.47-53, 1984.

[2] C. Gentry, Certificate-based encryption and the certificate revocation problem, *Proc. of EUROCRYPT 2003*, Warsaw, Poland, pp.272-293, 2003.

[3] S. S. Al-Riyami and K. G. Paterson, CBE from CL-PKE: A generic construction and efficient schemes, *Proc. of PKC 2005*, Les Diablerets, Switzerland, pp.398-415, 2005.

[4] P. Morillo and C. Ràfols, Certificate-based encryption without random oracles, *Cryptology ePrint Archive, Report 2006/12*, 2006.

[5] C. Sur, C. D. Jung and K. H. Rhee, Multi-receiver certificate-based encryption and application to public key broadcast encryption, *Proc. of 2007 ECSIS Symposium on Bio-Inspired, Learning, and Intelligent Systems for Security*, Edinburgh, UK, pp.35-40, 2007.

[6] D. Galindo, P. Morillo and C. Ràfols, Improved certificate-based encryption in the standard model, *Journal of Systems and Software*, vol.81, no.7, pp.1218-1226, 2008.

[7] J. K. Liu and J. Zhou, Efficient certificate-based encryption in the standard model, *Proc. of the 6th International Conference on Security and Cryptography for Networks*, Amalfi, Italy, pp.144-155, 2008.

[8] Y. Lu, J. Li and J. Xiao, Constructing efficient certificate-based encryption with paring, *Journal of Computers*, vol.4, no.1, pp.19-26, 2009.

[9] Y. Lu and J. Li, Forward-secure certificate-based encryption and its generic construction, *Journal of Network*, vol.5, no.5, pp.527-534, 2010.

[10] Z. Shao, Enhanced certificate-based encryption from pairings, *Computers and Electrical Engineering*, vol.37, no.2, pp.136-146, 2011.

[11] B. G. Kang, J. H. Park and S. G. Hahn, A certificate-based signature scheme, *Proc. of the Cryptographers' Track at the RSA Conference 2004*, San Francisco, CA, USA, pp.99-111, 2004.

[12] M. H. Au, J. K. Liu, W. Susilo and T. H. Yuen, Certificate based (linkable) ring signature, *Proc. of the 3rd Information Security Practice and Experience Conference*, Hong Kong, China, pp.79-92, 2007.

[13] J. Li, X. Huang, Y. Mu, W. Susilo and Q. Wu, Certificate-based signature: Security model and efficient construction, *Proc. of the 4th European PKI Workshop*, Palma De Mallorca, Spain, pp.110-125, 2007.

[14] J. K. Liu, J. Baek, W. Susilo and J. Zhou, Certificate based signature schemes without pairings or random oracles, *Proc. of the 11th International Conference on Information Security*, Taipei, China, pp.85-297, 2008.

[15] J. Li, X. Huang, Y. Mu, W. Susilo and Q. Wu, Constructions of certificate-based signature secure against key replacement attacks, *Journal of Computer Security*, vol.18, no.3, pp.421-449, 2010.

[16] P. S. L. M. Barreto, B. Lynn and M. Scott, Efficient implementation of pairing-based cryptosystems, *Journal of Cryptology*, vol.17, no.4, pp.321-334, 2004.

[17] I. Blake, V. Murty and G. Xu, Refinements of Miller's algorithm for computing the Weil/Tate pairing, *Journal of Algorithms*, vol.58, no.2, pp.134-149, 2006.

[18] E. Okamoto and K. Tanaka, Key distribution system based on identification information, *IEEE Journal on Selected Areas in Communications*, vol.7, no.4, pp.481-485, 1989.

[19] M. Bellare and P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, *Proc. of the 1st ACM CCCS*, Fairfax, VA, USA, pp.62-73, 1993.