CCA SECURE CERTIFICATELESS ENCRYPTION SCHEMES BASED ON RSA

S. Sree Vivek*, S. Sharmila Deva Selvi and C. Pandu Rangan*

Theoretical Computer Science Lab, Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai, India

Keywords: Certificateless encryption, Adaptive chosen ciphertext secure (CCA2), RSA assumption, Random Oracle model.

Abstract: Certificateless cryptography, introduced by Al-Riyami and Paterson eliminates the key escrow problem inherent in identity based cryptosystem. In this paper, we present two novel and completely different RSA based adaptive chosen ciphertext secure (CCA2) certificateless encryption schemes. For the first scheme, the security against Type-I adversary is reduced to RSA problem, while the security against Type-II adversary is reduced to the CCDH problem. For the second scheme both Type-I and Type-II security is related to the RSA problem. The new schemes are efficient when compared to other existing certificateless encryption schemes that are based on the costly bilinear pairing operation and are quite comparable with the certificateless encryption scheme based on multiplicative groups (without bilinear pairing) by Sun et al. (Sun et al., 2007) and the RSA based CPA secure certificateless encryption scheme by Lai et al. (Lai et al., 2009). We consider a slightly stronger security model than the ones considered in (Lai et al., 2009) and (Sun et al., 2007) to prove the security of our schemes.

1 INTRODUCTION

Cryptosystem based on Public Key Infrastructure (PKI) allows any user to choose his own private key and the corresponding public key. The public key is submitted to a certification authority (CA), which verifies the identity of the user and issues certificates linking his identity and the public key. Thus, a PKI based system needs digital certificate management that is too cumbersome to maintain and manage. Adi Shamir introduced the notion of Identity Based Cryptography (IBC) (Shamir, 1984) to reduce the burden of a PKI due to digital certificate management. In IBC, the private key of a user is not chosen by him, instead it is generated and issued by a trusted authority called the Private Key Generator (PKG) or Trust Authority (TA). This private key corresponds to the user’s public key which is generated from strings that represent the user’s identity, avoiding the need for certificates altogether. The PKG is responsible for generating the private keys of all the users in the system and it knows the private keys of all the users in the system. This inherent weakness of IBC is called as the key escrow problem. Certificateless Cryptography (CLC) introduced by Al-Riyami and Paterson (Al-Riyami and Paterson, 2003) addresses this issue to some extent, while avoiding the use of certificates and the need for CA. The principle behind CLC is to partition the private key of a user into two components: an identity based partial private key (generated by the PKG) and a non-certified private key (which is chosen by the user and not known to the PKG). This technique potentially combines the best features of IBC and PKI.

CLC also uses identities that uniquely identify a user in the system as in IBC but the public key of a user is not his identity alone but it is a combination of his identity and the public key corresponding to the non-certified private key chosen by the user. CLC involves a trusted third party as in IBC, named as the Key Generation Center (KGC), who generates partial private keys for the users registered with it. Each user selects his own secret value and a combination of the partial private key and the secret value acts as the full private key of the user. The authors of (Al-Riyami and Paterson, 2003) have shown realization for certificateless encryption (CLE), signature (CLS) and key exchange (CLK) schemes in their pa-
Related Works. Both the aforementioned schemes, namely (Baek et al., 2005) and (Sun et al., 2007) were based on multiplicative groups. Lai et al. in (Lai et al., 2009) proposed the first RSA-based CLE scheme. They have proved their scheme secure against chosen plaintext attack (CPA). In fact they left the design of a CCA secure system based on RSA as open. One may be tempted to think that the CPA secure scheme of Lai et al. in (Lai et al., 2009) can be made CCA secure by using any well known transformations like (Fujisaki and Okamoto, 1999b), (Fujisaki and Okamoto, 1999a) but giving access to the secret value of the target identity and strong decryption oracle to the Type-I adversary makes the resulting scheme insecure. Moreover, the scheme in (Lai et al., 2009) cannot be directly extended to a CLE scheme, whose Type-I and Type-II security relies on RSA assumption without making considerable changes in the scheme, hence we design a totally new scheme from scratch.

Our Contribution. In this paper, we propose two CLE schemes. The Type-I security of the first scheme is based on the RSA assumption and the Type-II security is based on the composite computational Diffie Hellman assumption (CCDH). Both Type-I and Type-II securities of our second scheme are based on the RSA assumption. Thus, we provide a scheme which is partially RSA based (like (Lai et al., 2009), but CCA2 secure) and another scheme which is fully RSA based. We formally prove both our schemes to be Type-I and Type-II secure under adaptive chosen ciphertext attack (CCA2) in the random oracle model. This is the strongest security notion for any encryption scheme. One of the striking features of our schemes is the novel key construction algorithm, which is completely new and different from other key constructs used so far in designing CLE. Moreover, our security model is stronger than the security models considered in the two existing secure schemes, (Lai et al., 2009) and (Sun et al., 2007). First, the existing schemes do not provide access to the secret value corresponding to the target identity during the Type-I confidentiality game, while we provide the secret value to the adversary. Second, we provide the strong decryption oracle for Type-I adversary. Strong decryption oracle means the decryption corresponding to a ciphertext is provided by the challenger even if the public key of a user is replaced after the generation of the ciphertext (Dent, 2008). We provide these oracle queries to the Type-I adversary of both the schemes and prove the security of our schemes in this stronger model. We stress that our second scheme is the major contribution in this paper and the first scheme is a stepping stone towards our fully RSA secure scheme. Even though computation of bilinear pairing has become efficient, finding out pairing friendly curves are difficult (Freeman et al., 2010) and most of the efficient curves and means of compressing are patented. Thus, we have only a hand full of elliptic curves that support pairing for designing cryptosystem. Besides, since the RSA patent expired in the year 2000, designing cryptographic schemes based on RSA assumption gets more attention these days. Hence, the research in pairing free protocol is a very important and worthwhile effort.

We use the following well known hard problems to establish the security of our new schemes:

**Definition 1.1 (The RSA Problem).** Given an RSA public key \((n,e)\), where \(n = pq\), \(p\), \(q\), \((p - 1)/2\) and \((q - 1)/2\) are large prime numbers, \(e\) is an odd integer such that \(gcd(e, \phi(n)) = 1\) and \(b \in \mathbb{Z}_n\), finding \(a \in \mathbb{Z}_n^\ast\) such that \(a^e \equiv b \mod n\) is referred as the RSA problem.

An RSA problem solver with \(\epsilon\) advantage is a probabilistic polynomial algorithm \(A_{RSA}\) which solves the RSA problem and \(\epsilon = \text{Prob}[a -
\( \mathcal{A}_{RSA}(n, e, b = d^e) \).

**Definition 2.2 (The Composite Computational Diffie Hellman Problem (CCDH)).** (Samuel, 1985), (McCyc, 1988). Given \( p, q, n, (g, g^a, g^b) \in \mathbb{Z}_n^* \), where \( n \) is a composite number with two big prime factors \( p \) and \( q \), also \( (p - 1)/2 \) and \( (q - 1)/2 \) are prime numbers, finding \( g^{ab} \mod n \) is the Composite Computational Diffie Hellman Problem in \( \mathbb{Z}_n^* \), where \( a, b \in \mathbb{Z}_n^{odd} \).

The advantage of any probabilistic polynomial time algorithm \( \mathcal{A} \) in solving the CCDH problem in \( \mathbb{Z}_n^* \) is defined as

\[
\text{Adv}^{\text{CCDH}}_\mathcal{A} = \Pr \left( \mathcal{A}(p, q, n, g, g^a, g^b) = g^{ab} \mid a, b \in \mathbb{Z}_n^{odd} \right)
\]

The **CCDH Assumption** is that, for any probabilistic polynomial time algorithm \( \mathcal{A} \), the advantage \( \text{Adv}^{\text{CCDH}}_\mathcal{A} \) is negligibly small.

# 2 FRAMEWORK AND SECURITY MODELS

In this section, we discuss the general framework for CLE. We adopt the definition of certificateless public-key encryption, given by Baek et al. (Baek et al., 2005). Their definition of CLE is weaker than the original definition by Al-Riyami and Paterson (Al-Riyami and Paterson, 2003) because the user has to obtain a partial public key from the KGC before he can create his public key (While in Al-Riyami and Paterson’s original CLE this is not the case). We also review the notion of Type-I and Type-II adversaries and provide the security model for CLE.

## 2.1 Framework for CLE

A certificateless public-key encryption scheme is defined by six probabilistic, polynomial-time algorithms which are defined below:

**Setup.** This algorithm takes as input a security parameter \( 1^k \) and returns the master private key \( msk \) and the system public parameters \( params \). This algorithm is run by the KGC in order to initialize a certificateless system.

**Partial Key Extract.** This algorithm takes as input the public parameters \( params \), the master private key \( msk \) and an identity \( ID_A \in \{0, 1\}^* \) of a user \( A \). It outputs the partial private key \( s_A \) and a partial public key \( PK_A \) of user \( A \). This algorithm is run by the KGC once for each user and the corresponding partial private key and partial public key is given to \( A \) through a secure and authenticated channel.

**Set Private Key.** This algorithm is run once by each user. It takes the public parameters \( params \), the user identity \( ID_A \) and \( A \)'s partial private key \( s_A \) as input. The algorithm generates a secret value \( y_A \in S \), where \( S \) is the secret value space. Now, the full private key \( D_A \) is a combination of the secret value \( y_A \) and the partial private key \( s_A \) of \( A \).

**Set Public Key.** This algorithm run by the user, takes as input the public parameters \( params \), a user, say \( A \)'s partial public key \( PPK_A \) and the full private key \( D_A \).

**Encryption.** This algorithm takes as input the public parameters \( params \), a user \( A \)'s identity \( ID_A \), the user public key \( PK_A \) and a message \( m \in M \). The output of this algorithm is the ciphertext \( \sigma \in C_S \). Note that \( M \) is the message space and \( C_S \) is the ciphertext space.

**Decryption.** This algorithm takes as input the public parameters \( params \), a user, say \( A \)'s private key \( D_A \) and a ciphertext \( \sigma \in C \). It returns either a message \( m \in M \) if the ciphertext is valid, or \( Invalid \) otherwise.

## 2.2 Security Model for CLE

The confidentiality of any CLE scheme is proved by means of an interactive game between a challenger \( C \) and an adversary. In the confidentiality game for certificateless encryption (IND-CLE-CCA2) the adversary is given access to the following five oracles. These oracles are simulated by \( C \):

**Partial Key Extract for \( ID_A \).** \( C \) responds by returning the partial private key \( s_A \) and the partial public key \( PPK_A \) of the user \( A \).

**Extract Secret Value for \( ID_A \).** If \( A \)'s public key has not been replaced then \( C \) responds with the secret value \( y_A \) for user \( A \). If the adversary has already replaced \( A \)'s public key, then \( C \) does not provide the corresponding private key to the adversary.

**Request Public Key for \( ID_A \).** \( C \) responds by returning the full public key \( PK_A \) for user \( A \). (First by choosing a secret value if necessary).

**Replace Public Key for \( ID_A \).** The adversary can repeatedly replace the public key \( PK_A \) for a user \( A \) with any valid public key \( PK'_A \) of its choice. The current value of the user’s public key is used by \( C \) in any computations or responses.

**Decryption for Ciphertext \( \sigma \) and Identity \( ID_A \).** The adversary can issue a decryption query for ciphertext \( \sigma \) and identity \( ID_A \) of its choice, \( C \) decrypts \( \sigma \) and returns the corresponding message to the adversary. \( C \)
should be able to properly decrypt ciphertexts, even for those users whose public key has been replaced, i.e. this oracle provides the decryption of a ciphertext, which is generated with the current valid public key. The strong decryption oracle returns Invalid, if the ciphertext corresponding to any of the previous public keys were queried. This is a strong property of the security model (Note that, C may not know the correct private key of the user). However, this property ensures that the model captures the fact that changing a user’s public key to a value of the adversary’s choice may give the adversary an advantage in breaking the scheme. This is called as strong decryption in (Dent, 2008). Our schemes provides strong decryption for Type-I adversary.

There are two types of adversaries (namely Type-I and Type-II) to be considered for any certificateless encryption scheme. The Type-I adversary models the attack by a third party attacker, i.e. anyone except the legitimate receiver or the KGC who is trying to gain the access to master private key \( msu \). This is a strong property of the security model (Note that, \( C \) may not know the correct private key of the user). However, this property ensures that the model captures the fact that changing a user’s public key to a value of the adversary’s choice may give the adversary an advantage in breaking the scheme. This is called as strong decryption in (Dent, 2008). Our schemes provides strong decryption for Type-I adversary.

**Constraints for Type-I and Type-II Adversaries.** The IND-CLE-CCA2 security model distinguishes the two types of adversary Type-I and Type-II with the following constraints.

- Type-I adversary \( A_I \) is allowed to change the public keys of users at will but does not have access to the master private key \( msu \).
- Type-II adversary \( A_H \) is equipped with the master private key \( msu \) but is not allowed to replace public keys corresponding to the target identity.

**IND-CLE-CCA2 Game for Type-I Adversary.** The game is named as IND-CLE-CCA2-I. This game, played between the challenger \( C \) and the Type-I adversary \( A_I \), is defined below:

**Setup.** Challenger \( C \) runs the setup algorithm to generate master private key \( msu \) and public parameters \( params \). \( C \) gives \( params \) to \( A_I \) while keeping \( msu \) secret. After receiving \( params \), \( A_I \) interacts with \( C \) in two phases:

**Phase I.** \( A_I \) is given access to all the five oracles. \( A_I \) adaptively queries the oracles consistent with the constraints for Type-I adversary described above.

**Challenge.** At the end of Phase I, \( A_I \) gives two messages \( m_0 \) and \( m_1 \) of equal length to \( C \) on which it wishes to be challenged. \( C \) randomly chooses a bit \( \delta \in \{0,1\} \) and encrypts \( m_\delta \) with the target identity \( ID^* \)’s public key to form the challenge ciphertext \( c^* \) and sends it to \( A_I \) as the challenge. (Note that the partial Private Key corresponding to \( ID^* \) should not be queried by \( A_I \) but the secret value corresponding to \( ID^* \) may be queried. This makes our security model stronger when compared to the security models of (Lai et al., 2009) and (Sun et al., 2007).)

**Phase II.** \( A_I \) adaptively queries the oracles consistent with the constraints for Type-I adversary described above. Besides this \( A_I \) cannot query Decryption on \( (\sigma^*,ID^*) \) and the partial private key of the receiver should not have been queried to the Extract Partial Private Key oracle.

**Guess.** \( A_I \) outputs a bit \( \delta' \) at the end of the game. \( A_I \) wins the IND-CLE-CCA2-I game if \( \delta' = \delta \). The advantage of \( A_I \) is defined as:

\[
\text{Adv}_{A_I}^{\text{IND-CLE-CCA2-I}} = 2p_r(|\delta - \delta'| - 1)
\]

**IND-CLE-CCA2 Game for Type-II Adversary.** The game is named as IND-CLE-CCA2-II. This game, played between the challenger \( C \) and the Type-II adversary \( A_H \), is defined below:

**Setup.** Challenger \( C \) runs the setup algorithm to generate master private key \( msu \) and public parameters \( params \). \( C \) gives \( params \) and the master private key \( msu \) to \( A_H \). After receiving \( params \), \( A_H \) interacts with \( C \) in two phases:

**Phase I.** \( A_H \) is not given access to the Extract partial Private Key oracle because \( A_H \) knows \( msu \), it can generate the partial private key of any user in the system. All other oracles are accessible by \( A_H \). \( A_H \) adaptively queries the oracle consistent with the constraints for Type-II adversary described above.

**Challenge.** At the end of Phase I, \( A_H \) gives two messages \( m_0 \) and \( m_1 \) of equal length to \( C \) on which it wishes to be challenged. \( C \) randomly chooses a bit \( \delta \in \{0,1\} \) and encrypts \( m_\delta \) with the target identity \( ID^* \)’s public key to form the challenge ciphertext \( c^* \) and sends it to \( A_H \) as the challenge. (Note that the Secret Value Corresponding to \( ID^* \) should not be queried by \( A_H \) and the public key corresponding to \( ID^* \) should not be replaced during Phase I.)

**Phase II.** \( A_H \) adaptively queries the oracles consistent with the constraints for Type-II adversary described above. Besides this \( A_H \) cannot query Decryption on \( (\sigma^*,ID^*) \) and the Secret Value corresponding to the receiver should not be queried to the Extract Secret Value oracle and the public key corresponding to \( ID^* \) should not be replaced during Phase I.
Guess. $A_H$ outputs a bit $\delta'$ at the end of the game. $A_H$ wins the IND-CLE-CCA2-II game if $\delta' = \delta$. The advantage of $A_H$ is defined as:

$$\text{Adv}^{\text{IND-CLE-CCA2-II}}_{A_H} = |2Pr[\delta = \delta'] - 1|$$

3 BASIC RSA-BASED CLE SCHEME (RSA-CLE$_1$)

In this section, we propose the basic RSA based certificateless encryption scheme RSA-CLE$_1$ and also prove the security of the scheme against both Type-I and Type-II adversaries under adaptive chosen ciphertext attack (CCA2). For this scheme the Type-I security relies on the RSA assumption and the Type-II security is based on the composite computational Diffie Hellman assumption (CDH).

Notation. We use the notation $Z_p^{\text{odd}}$ to represent the odd numbers from $[0,n]$. Throughout the paper, in order to choose a random odd number from the range $[1,n]$, we randomly pick an element in $Z_n$ and check whether it is odd, if it is odd, we accept it, else we subtract 1 from the chosen number. These numbers are represented as $Z_n^{\text{odd}}$.

3.1 The RSA-CLE$_1$ Scheme

The proposed scheme comprises the following six algorithms. Unless stated otherwise, all computations except those in the Setup algorithm are done mod $n$.

Setup. The KGC does the following to initialize the system and to setup the public parameters.

- Chooses two primes $p$ and $q$, such that $p = 2p' + 1$ and $q = 2q' + 1$ where $p'$ and $q'$ are also primes.
- Computes $n = pq$ and the Euler’s totient function $\phi(n) = (p-1)(q-1)$.
- It also chooses four cryptographic hash functions $H : \{0, 1\}^* \rightarrow Z_n^*$, $H_1 : \{0, 1\}^* \times Z_n^* \rightarrow Z_n^{\text{odd}}$, $H_2 : \{0, 1\}^* \times Z_n^* \rightarrow Z_n^{\text{odd}}$ and $H_3 : Z_n^* \times Z_n^* \times \{0, 1\}^* \rightarrow \{0, 1\}^{1+[\frac{\text{size of message}}{2}]}$, where $l$ is the size of the message.
- Now, KGC publicizes the system parameters, $\text{params} = (n, H, H_1, H_2, H_3)$ and keeps the factors of $n$, namely $p$ and $q$ as the master private key.

Note. Since $n$ is a product of two strong primes, a randomly chosen number in $Z_n^{\text{odd}}$ is relatively prime to $\phi(n)$ with overwhelming probability. The RSA modulus $n$ is set to $n = pq$ and $p$, $q$ are chosen such that $p = 2p' + 1$, $q = 2q' + 1$ where both $p'$ and $q'$ are also large primes. Considering $\phi(n) = 2p'q'$ with only three factors $2, p', q'$, the probability of any odd number being co-prime to $\phi(n)$ is overwhelming, because finding a number not co-prime to $4p'q'$ is equivalent to finding $p'$ or $q'$ or finding $p$ or $q$. Thus, hardness of factoring implies that the random odd number in $Z_n$ is relatively prime to $\phi(n)$ with very high probability.

Partial Key Extract. Our partial key extraction is not a deterministic algorithm, i.e. this algorithm gives different partial keys for the same identity when queried more than once. Examples for this type of key extraction can be found in (Baek et al., 2005) and (Sun et al., 2007). This algorithm is executed by the KGC and upon receiving the identity ID$_A$ of a user $A$ the KGC performs the following to generate the corresponding partial private key $d_A$.

- Chooses $x_A \in_R Z_n^{\text{odd}}$.
- Computes $g_A = H(ID_A)$.
- Computes the partial public key $PPK_A = g_A^{x_A}$.
- Computes the value $e_A = H_1(ID_A, PPK_A)$.
- Computes $d_A$ such that $e_A d_A = 1 \mod \phi(n)$ and sends the partial private key $s_A = x_A + d_A \mod \phi(n)$ and the partial public key $PPK_A$ to the user through a secure channel.

The validity of the partial private key can be verified by user $A$ by performing the following check:

$$\left( g_A^{x_A} \right)^{s_A} g_A = \left( g_A \right)^{e_A}$$

Note. However, this can be made deterministic by obtaining the randomness used in the computation of the partial public key through a secure MAC (Message Authentication Code) with the identity of the user as input and the master private key as the key to the MAC.

Set Private Key. On receiving the partial private key the user with identity ID$_A$ does the following to generate his full private key.

- Chooses $y_A \in_R Z_n^{\text{odd}}$ as his secret value.
- Sets the private key as $D_A = \langle D_A^{(1)}, D_A^{(2)} \rangle = \langle s_A, y_A \rangle$. (Note that both the KGC and the corresponding user knows $D_A^{(1)}$ and the user with identity ID$_A$ alone knows $D_A^{(2)}$).

Set Public Key. The user with identity ID$_A$ computes the public key corresponding to his private key as described below:

- Computes $g_A = H(ID_A)$.
- Computes the value $g_A^{s_A}$.
- Makes $PK_A = \langle PK_A^{(1)}, PK_A^{(2)} \rangle = \langle PPK_A, g_A \rangle$ public.
Note that $g_A^{x_A}$ was sent by KGC to the user while setting $ID_A$’s partial private key. The validity of the public key can be publicly verified using the following verification test:

- Compute $e_A = H_1(ID_A, PK_A^{(1)})$.
- Check whether the following holds:
  \[
  (PK_A^{(1)})^{x_A} = (PK_A^{(1)})^{e_A} g_A^{x_A}
  \]

### Encryption

To encrypt a message $m$ to a user with identity $ID_A$, one has to perform the following steps:

- Choose $r \in \mathbb{Z}_n^{odd}$.
- Compute $e_A = H_1(ID_A, PK_A^{(1)})$, $g_A = H(ID_A)$ and $h = H_2(m, r)$.
- Compute $c_1 = g_A^h$ and $c_2 = (m||r) \oplus H_3((PK_A^{(1)})^{h}, (PK_A^{(2)})^h, ID_A)$.

Now, $\sigma = (c_1, c_2)$ is send as the ciphertext to the user $A$.

### Decryption

The receiver with identity $ID_A$ does the following to decrypt a ciphertext $\sigma = (c_1, c_2)$:

- Find $(m||r) = c_2 \oplus H_3((c_1)^{g_A^{-1}} - c_1, (c_1)^{DP_2}, ID_A)$.
- Computes $h' = H_2(m, r)$ and checks whether $c_1 \equiv g_A^h$.

User $A$ accepts the message only if the above check holds.

### 3.2 Security Proof

In order to prove the confidentiality of a certificateless encryption scheme, it is required to consider the attacks by Type-I and Type-II adversaries. In the two existing secure schemes (Lai et al., 2009) and (Sun et al., 2007), the Type-I adversary is not allowed to extract the secret value corresponding to the target identity. In order to capture the ability of the adversary who can access the secret keys of the target identity, we give access to the user secret value of the target identity to the Type-I adversary. We also state that, allowing the extract secret value query corresponding to the target identity makes the security model for Type-I adversary more stronger.

#### 3.2.1 Confidentiality against Type-I Adversary

**Theorem 3.1** Our certificateless public key encryption scheme RSA-CLE$_1$ is IND-RSA-CLE$_1$-CCA2-1 secure in the random oracle model, if the RSA problem is intractable in $\mathbb{Z}_n$, where $p$, $q$, $(p-1)/2$ and $(q-1)/2$ are large prime numbers.

**Proof Sketch.** The challenger $C$ is challenged with an instance of the RSA problem, say $(n, e, r^{odd}, b) \in \mathbb{Z}_n$, where $n$ is a composite number with two big prime factors $p$ and $q$. $(p-1)/2$ and $(q-1)/2$ are also primes. Let us consider that there exists an adversary $A_I$ who is capable of breaking the IND-RSA-CLE$_1$-CCA2-1 security of the RSA-CLE$_1$ scheme. $C$ can use $A_I$ to compute a such that $a^e \equiv b \mod n$, by playing the following interactive game with $A_I$.

**Setup.** $C$ begins the game by setting up the system parameters as in the RSA-CLE$_1$ scheme. $C$ takes $n$ from the instance of the RSA problem that $C$ has received and sends $params = (n)$ to $A_I$. $C$ also designs the four hash functions $H$, $H_1$, $H_2$ and $H_3$ as random oracles $O_H$, $O_{H_1}$, $O_{H_2}$, and $O_{H_3}$.

**Phase I.** $A_I$ performs a series of queries to the oracles provided by $C$. The descriptions of the oracles and the responses given by $C$ to the corresponding oracle queries by $A_I$ are described below:

**Note.** We assume that $O_H(.)$ oracle is queried with $ID_1$, as input, before any other oracle is queried with the corresponding identity, $ID_1$ as one of the inputs.

- **$O_H(ID_1)$:** We follow the proof methodology introduced in (Boyen, 2003) and make a simplifying assumption that $A_I$ queries the $O_H$ oracle with distinct identities in each query. This is because, if the same identity is repeated, by definition, the oracle consults the list $L$ and gives the same response. Thus, we assume that $A_I$ asks $q_{H}^I$ distinct queries for $q_{H}^I$ distinct identities. Among this $q_{H}^I$ identities, a random identity has to be selected as target identity by $C$. $C$ selects a random index $\gamma$, where $1 \leq \gamma \leq q_{H}^I$ and $C$ does not reveal $\gamma$ to $A_I$. When $A_I$ generates the $\gamma^h$ query on $ID_{\gamma}$, $C$ fixes $ID_{\gamma}$ as target identity for the challenge phase.

For answering the $O_H$ query, $C$ performs the following, for $1 \leq \gamma \leq q_{H}^I$:

- If a tuple of the form $(ID_{\gamma}, e_{\gamma}, b_{\gamma}, g_{\gamma})$ exists in the list $L$ then $C$ retrieves the corresponding $g_{\gamma}$.
- Else,
  - If $i \neq \gamma$, $C$ performs the following:
    - $C$ chooses $e_{i} \in \mathbb{Z}_n^{odd}$, $b_{i} \in \mathbb{Z}_n^{*}$ and computes $g_{i} = b_{i}^{e_{i}}$.
    - Generates the partial private key corresponding to $ID_1$ as follows:
      - Chooses $s_{i} \in \mathbb{Z}_n^{odd}$.
      - Computes $g_{i}^{s_{i}/b_{i}}$. Let $g_{i}^{s_{i}/b_{i}} = g_{i}^{x_{i}}$ for some $x_{i}$.
        (Note that $x_{i}$ is not known to $C$.)
      - Chooses $y_{i} \in \mathbb{Z}_n^{odd}$ and adds the tuple $(ID_{\gamma}, s_{i}, g_{i}^{x_{i}}, y_{i})$ in the list $L_{\gamma}$.
If a tuple of this form exists, if \( \exists i \neq i \), then \( A_i \) has replaced the private key corresponding to \( ID_i \).

\[ \text{O}_{\text{ExtractSecretValue}}(ID_i), \quad \text{C} \text{ retrieves the tuple of the form } (ID_i, s_i, g_i, y_i) \text{ from the list } L_5 \text{ and returns the corresponding } y_i \text{ as the secret value corresponding to the identity } ID_i. \]

If the entry corresponding to \( y_i \) in the tuple is "-1" then \( A_i \) has replaced the private key corresponding to \( ID_i \).

\[ \text{O}_{\text{RequestPublicKey}}(ID_i), \quad \text{C} \text{ retrieves the tuple of the form } (ID_i, g_i, y_i, g_i, e_i) \text{ from the list } L_3 \text{ and returns } PK_i = (g_i, y_i, g_i, e_i) \text{ as the public key corresponding to the identity } ID_i. \]

\[ \text{O}_{\text{ReplacePublicKey}}(ID_i, PK_i'). \quad \text{To replace the public key of } ID_i \text{ with a new public key } PK_i' = (PK_i'^{(1)}, PK_i'^{(2)}, PK_i'^{(3)}), \text{ chosen by } A_i, \text{ C does the following:} \]

- Updates the corresponding tuples in the list \( L_3 \) as \( (ID_i, g_i, y_i, g_i, e_i) \), only if \( (PK_i'^{(1)})^{e_i} = (PK_i^{(1)})^{e_i} g_i \), where \( g_i \) corresponding to \( ID_i \) is retrieved from the list \( L_3 \).
- Returns \text{Invalid}, otherwise.

\[ \text{O}_{\text{StrongDecryption}}(G, ID_i, PK_i), \quad \text{C performs the following to decrypt the ciphertext } \sigma = (c_1, c_2): \]

- Checks the validity of \( PK_i \) and rejects the ciphertext \( \sigma \) if this check fails, else proceeds with the following steps.
- Retrieves the tuple \( (ID_i, g_i, e_i) \) from list \( L_3 \).
- For each \( (m, r, h) \in L_2 \) list performs the following:
  - Checks whether \( g_i^r = c_1 \).
  - If True, computes \( k_1 = (PK_i'^{(1)})^{e_i} h \) and \( k_2 = (PK_i'^{(2)})^{c_2} h \).
- Checks in list \( L_3 \), for an entry corresponding to \((k_1, k_2, ID_i)\). If a tuple exists then retrieves the corresponding \( h_3 \) value and checks whether \( c_2 \oplus h_3 = m \) where \( m, r \) are retrieved from the list \( L_2 \).
- If True, outputs \( m \) as the message.
- If no tuple satisfies all the above tests, returns \text{Invalid}.

**Challenge.** At the end of Phase I, \( A_i \) produces two messages \( m_0 \) and \( m_1 \) of equal length and an identity \( ID' \). \( C \) aborts the game if \( ID' \neq ID_i \), else randomly chooses a bit \( \delta \in \{0, 1\} \) and computes a ciphertext \( \sigma' \) with \( ID_i' \) as the receiver by performing the following steps:

- Set \( c_1' = b^w \), where \( b \) is taken from the RSA problem instance received by \( C \) and \( z \) is the value chosen during the \( \text{O}_{H(\cdot)} \) oracle query corresponding to \( ID_i \).
- Choose \( c_2' \in \{0, 1\}^{\ell + \lceil 2 \ell \rceil} \).
Now, $\sigma^* = (c_1^*, c_2^*)$ is sent to $A_1$ as the challenge ciphertext. It should be noted that with overwhelming probability, $\sigma^*$ is a valid ciphertext and since $A_1$ is disallowed to query the strong decryption oracle with $\sigma^*$ as input, $A_1$ will not be able to identify whether $\sigma^*$ is valid or not.

**Phase II.** $A_1$ performs the second phase of interaction, where it makes polynomial number of queries to the oracles provided by $C$ with the following conditions:
- $A_1$ should not have queried the Strong Decryption oracle with $(\sigma^*, PK_q, ID_q)$ as input. (It is to be noted that $PK_q$ is the public key corresponding to $ID_q$ during the challenge phase. $A_1$ can query the decryption oracle with $(\sigma^*, PK^*, ID_1)$ as input, $\forallPK^* \neq PK_q$)
- $A_1$ should not query the partial private key corresponding to $ID_q$.
- $A_1$ can query the secret value corresponding to $ID_q$ and $PK_q$.

**Guess.** At the end of **Phase II**, $A_1$ produces a bit $b'$ to $C$, but $C$ ignores the response and performs the following to output the solution for the RSA problem instance.
- For each tuple of the form $(k_1, k_2, ID_3, h_3)$ in list $L_3$, $C$ checks whether $k_1 \equiv h_3$ (where $e$ and $b$ are taken from the RSA problem instance).
- Outputs the corresponding $k_1$ value for which the above check holds as the solution (i.e., $a = k_1$) for the RSA problem instance.

### 3.2.2 Confidentiality against Type-II Adversary

**Theorem 3.2.** Our certificateless public key encryption scheme RSA-CLE1 is IND-RSA-CLE1-CCA2-II secure in the random oracle model, if the CDH problem is intractable in $\mathbb{Z}_n^*$, where $n = pq$ and $p, q, (p - 1)/2, (q - 1)/2$ are large prime numbers.

Due to page limitation we present the formal proof of this theorem in the full version of the paper (Selvi et al., 2010).

## 4 FULLY RSA BASED CLE SCHEME (RSA-CLE2)

In this section, we propose the fully RSA based certificateless encryption scheme RSA-CLE2. The Type-I security is similar to that of the Type-I security proof of RSA-CLE1. We prove the security of the scheme against Type-II attacks under adaptive chosen ciphertext attack (CCA2) assuming the hardness of RSA problem.

### 4.1 The RSA-CLE2 Scheme

The proposed scheme comprises the following six algorithms. Unless stated otherwise all computations except those in the setup algorithm are done $mod \ n$.

**Setup.** The KGC does the following to initialize the system and to setup the public parameters.
- Chooses two primes $p$ and $q$, such that $p = 2p' + 1$ and $q = 2q' + 1$ where $p'$ and $q'$ are also primes.
- Computes $n = pq$ and the Euler’s totient function $\phi(n) = (p - 1)(q - 1)$.
- It also chooses three cryptographic hash functions $H_1: \{0, 1\}^* \rightarrow \mathbb{Z}_n^*$, $H_2: \{0, 1\}^* \times \mathbb{Z}_n \rightarrow \mathbb{Z}_n^{odd}$, $H_3: \{0, 1\}^* \times \mathbb{Z}_n^* \rightarrow \{0, 1\}^{f(i)}$, where $i$ is the size of the message.
- Now, KGC publicizes the system parameters, $params = (n, H_1, H_2, H_3)$ and keeps the factors of $n$, namely $p$ and $q$ as the master private key.

**Partial Key Extract.** This algorithm is executed by the KGC and upon receiving the identity $ID_A$ of a user $A$ the KGC performs the following to generate the corresponding partial private key $d_A$.
-Chooses $x_A \in R_{\mathbb{Z}_n^{odd}}$.
-Computes $g_A = H(ID_A)$.
-Computes the partial public key $PPK_A = g_A^{x_A}$
-Computes the value $e_A = H_1(ID_A, PPK_A)$.
-Computes $d_A$ such that $e_dA \equiv 1 \ mod \ \phi(n)$ and sends the partial private key $s_A = x_A + d_A \ mod \ \phi(n)$ and the partial public key $PPK_A$ to the user through a secure channel.

**Set Private Key.** On receiving the partial private key the user with identity $ID_A$ does the following to generate his secret key.
- Chooses two primes $P_A$ and $Q_A$, such that $P_A = 2P_A' + 1$ and $Q_A = 2Q_A' + 1$, where $P_A'$ and $Q_A'$ are also primes.
-Computes $N_A = P_AQ_A$ and the Euler’s totient function $\phi(N_A) = (P_A - 1)(Q_A - 1)$.
-Chooses $\hat{e}_A \in R_{\mathbb{Z}_n^{odd}}$ as the user public key and computes $d_A = \hat{e}_A^{-1} \ mod \ \phi(N_A)$.
- Sets the private key as $D_A = \langle D_A^{(1)}, D_A^{(2)}, D_A^{(3)}, D_A^{(4)} \rangle = \langle s_A, d_A, P_A, Q_A \rangle$.

**Set Public Key.** The user with identity $ID_A$ computes the public key corresponding to his private key as $PK_A = \langle PK_A^{(1)}, PK_A^{(2)}, PK_A^{(3)}, PK_A^{(4)} \rangle = \langle D_A^{(1)}, g_A^{\hat{e}_A}, e_A, N_A \rangle$ and makes it public.
Note that \( g_A^{x_A} \) was sent by KCG to the user while setting \( ID_A \)’s partial private key. The validity of the public key can be publicly verified using the following verification test:
- Compute \( e_A = H_1(ID_A, PK_A^{(1)}) \) and \( g_A = H(ID_A) \).
- Check whether \( (PK_A^{(2)})^{x_A} \equiv (PK_A^{(1)})^{x_A} g_A \)

**Encryption.** To encrypt a message \( m \) to a user with identity \( ID_A \), one has to perform the following steps:
- Check the validity of the public key corresponding to \( ID_A \).
- Choose \( r \in \mathbb{Z}_{p}^{\text{odd}} \) and \( g \in \mathbb{Z}_N^{\ast} \).
- Compute \( e_A = H_1(ID_A, PK_A^{(1)}), g_A = H(ID_A) \) and \( h = H_2(m, r) \).
- Compute \( c_1 = g_A^{r}, c_2 = g^{PK_A^{(3)} \mod N_A} \) and \( c_3 = (m||r) \oplus H_2((PK_A^{(1)})^{x_A} g_A, ID_A) \).

Now, \( \sigma = (c_1, c_2, c_3) \) is send as the ciphertext to the user \( A \).

**Decryption.** The receiver with identity \( ID_A \) does the following to decrypt a ciphertext \( \sigma = (c_1, c_2, c_3) \):
- Computes \( k_1 = \left(\frac{c_1}{k_2}\right)^{PK_A^{(1)}} \mod N_A \).
- Retrieves \( (m||r) = H_3(k_1, k_2, ID_A) \oplus c_3 \).
- Computes \( h' = H_2(m, r) \) and checks whether \( c_1 = g_A^{r} \).

User \( A \) accepts the message only if the above check holds.

### 4.1.1 Confidentiality against Type-I Adversary

**Theorem 4.1** Our certificateless public key encryption scheme RSA-CLE\(_2\) is IND-RSA-CLE\(_2\)-CCA2-I secure in the random oracle model, if the RSA problem is intractable in \( \mathbb{Z}_N^{\ast} \) where \( p, q, (p-1)/2 \) and \( (q-1)/2 \) are large prime numbers.

The proof for this theorem is similar to that of the Type-I proof of RSA-CLE\(_1\) (IND-RSA-CLE\(_1\)-CCA2-I).

### 4.1.2 Confidentiality against Type-II Adversary

**Theorem 4.2** Our certificateless public key encryption scheme RSA-CLE\(_2\) is IND-RSA-CLE\(_2\)-CCA2-II secure in the random oracle model, if the RSA problem is intractable in \( \mathbb{Z}_N^{\ast} \) where \( N = PQ \) and \( P, Q, (P-1)/2, (Q-1)/2 \) are large prime numbers.

Due to page limitation we present the formal proof of this theorem in the full version of the paper (Selvi et al., 2010).

## 5 COMPARISON STUDY

In this paper, we have proposed two CCA2 secure certificateless encryption schemes. For the first scheme the Type-I security is based on the RSA assumption and Type-II security is based on the composite computational Diffie Hellman assumption. Both Type-I and Type-II securities of our second scheme are based on the RSA assumption. Our schemes are quite novel and based on entirely different key construct and protocol. It should be further noted that the existing schemes (Lai et al., 2009) and (Sun et al., 2007) consider a security model in which the Type-I adversary is not provided the extract secret value oracle, for the target identity. Our security model is stronger because we permit the extract secret value oracle corresponding to the target identity to the Type-I adversary. In fact, the scheme in (Lai et al., 2009) is not

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Security</th>
<th>Assumption</th>
<th>Type-I</th>
<th>Type-II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lai et al.</td>
<td>CPA</td>
<td>RSA</td>
<td>CCDH</td>
<td></td>
</tr>
<tr>
<td>(Lai et al., 2009)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun et al.</td>
<td>CCA2</td>
<td>CDH</td>
<td>CDH</td>
<td></td>
</tr>
<tr>
<td>(Sun et al., 2007)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSA-CLE(_1)</td>
<td>CCA2</td>
<td>RSA</td>
<td>CCDH</td>
<td></td>
</tr>
<tr>
<td>RSA-CLE(_2)</td>
<td>CCA2</td>
<td>RSA</td>
<td>CCDH</td>
<td></td>
</tr>
</tbody>
</table>

## 6 CONCLUSIONS
secure with this oracle access. However, in our security model the secret value corresponding to the target identity is given to the Type-I adversary, which makes it stronger. Moreover, we provide strong decryption oracle for Type-I adversary, i.e. the decryption of a ciphertext is provided by the challenger even if the public key of the corresponding user is replaced after the generation of the ciphertext. Thus we provide a CCA2 secure CLE whose security is partly based on RSA and another scheme which is fully based on RSA assumption. We have proved the security of the schemes in the random oracle model. We leave it an interesting open problem to design a CLE scheme in the original model (Al-Riyami and Paterson, 2003) with the security of the scheme fully based on RSA assumption.

ACKNOWLEDGEMENTS

We would like to extend our sincere thanks to the anonymous referees of the PROVSEC-2010 program committee for giving us insightful remarks which helped us to improve the security proof of the schemes.

REFERENCES


Shmuel, Z. (February, 1985.). Composite diffie-hellman public-key generating systems are hard to break. Technical Report No. 356, Computer Science Department, Technion-Israel Institute of Technology.