CCA2 Secure Certificateless Encryption Schemes Based on RSA

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

Theoretical Computer Science Lab,
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, India.
{sharmila,svivek,prangan}@cse.iitm.ac.in

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. The new schemes are efficient when compared to other existing certificatless 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. [18] and the RSA based CPA secure certificateless encryption scheme by Lai et al. [11]. We consider a slightly stronger security model than the ones considered in [11] and [18] to prove the security of our schemes.

Keywords. Certificateless encryption, Adaptive Chosen Ciphertext Secure (CCA2), RSA Assumption, Random Oracle model.

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) [15] 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 inherent weakness of IBC is the key escrow problem. 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, which is informally called as the key escrow problem. Certificateless Cryptography (CLC) introduced by Al-Riyami and Paterson [1] 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 [1] have shown realization for certificateless encryption (CLE), signature (CLS) and key exchange (CLK) schemes in their paper. Huang et al. [10] and Castro et al. [4] independently showed that the signature scheme in [1] is not secure against Type-I adversary (explained in later sections), i.e. it is possible to launch a key replacement attack on the scheme and they also gave a new certificateless signature scheme. Lot of CLE schemes were proposed, whose security were proved both in the random oracle model [2, 5, 16, 18] and standard model [12, 14]. Recently, Dent [6] has given a survey on the

* Work supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation sponsored by Department of Information Technology, Government of India

various security models for CLE schemes, mentioning the subtle difference in the level of security offered by each model. Dent has also given the generic construct and an efficient construction for CLE. The initial constructs for certificateless cryptosystem were all based on bilinear pairing [5, 16, 12, 14]. Baek et al. [2] were the first to propose a CLE scheme without bilinear pairing. Certificateless cryptosystem are prone to key replacement attack because the public keys are not certified and anyone can replace the public key of any legitimate user in the system. The challenging task in the design of certificateless cryptosystem is to come up with schemes which resists key replacement attacks. The CLE in [2] did not withstand key replacement attack, which was pointed out by Sun et al. in [18]. Sun et al. fixed the problem by changing the partial key extract and setting public key procedures.

Related Works. Both the aforementioned schemes, namely [2] and [18] were based on multiplicative groups. Lai et al. in [11] 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 [11] can be made CCA secure by using any well known transformations like [9], [8] 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 [11] 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 [11], 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, the security model for the two existing secure schemes, [11] and [18] do not provide access to the secret value corresponding to the target identity during the Type-I confidentiality game. We also 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 [6]. 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 [7] 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. ***(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 $b \in_R \mathbb{Z}_n^*$, finding $a \in \mathbb{Z}_n^*$ such that $a^e \equiv b \ (mod n)$ is referred as the RSA problem.*

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

Definition 2. ***(The Composite Computational Diffie Hellman Problem (CCDH)*** *[17], [13]) Given $p, q, n, \langle g, g^a, g^b \rangle \in \mathbb{Z}_n^*$, 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

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

The CCDH Assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{CCDH}$ 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. [2]. Their definition of CLE is weaker than the original definition by Al-Riyami and Paterson [1] 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^κ 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 PPK_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 $y_A \in \mathcal{S}$, where $\mathcal{S}$ 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. It outputs a public key PK_A for A. This algorithm is run once by the user and the resulting full public key is widely and freely distributed. The full public key of user A consists of PK_A and ID_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 $m \in \mathcal{M}$. The output of this algorithm is the ciphertext $\sigma \in \mathcal{CS}$. Note that $\mathcal{M}$ is the message space and $\mathcal{CS}$ 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 $\sigma \in \mathcal{C}$. It returns either a message $m \in \mathcal{M}$ - 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 $\mathcal{C}$ 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 $\mathcal{C}$:

Partial Key Extract for ID_A: $\mathcal{C}$ 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 $\mathcal{C}$ responds with the secret value y_A for user A. If the adversary has already replaced A's public key, then $\mathcal{C}$ does not provide the corresponding private key to the adversary.

Request Public Key for ID_A: $\mathcal{C}$ 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 $\mathcal{C}$ in any computations or responses.

Decryption for ciphertext σ and identity ID_A: The adversary can issue a decryption query for ciphertext σ and identity ID_A of its choice, $\mathcal{C}$ decrypts σ and returns the corresponding message to the adversary. $\mathcal{C}$ 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, $\mathcal{C}$ 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 [6]. 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 some information about a message from the encryption. The Type-II adversary models the honest-but-curious KGC who tries to break the confidentiality of the scheme. Here, the attacker is allowed to have access to master private key msk. This means that we do not have to give the attacker explicit access to partial key extraction, as the adversary is able to compute these value on its own. The most important point about Type-II security is that the adversary modeling the KGC should not have replaced the public key for the target identity before the challenge is issued.

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

- Type-I adversary $\mathcal{A}_I$ is allowed to change the public keys of users at will but does not have access to the master private key msk.
- Type-II adversary $\mathcal{A}_{II}$ is equipped with the master private key msk 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 $\mathcal{C}$ and the Type-I adversary $\mathcal{A}_I$, is defined below:

Setup: Challenger $\mathcal{C}$ runs the setup algorithm to generate master private key msk and public parameters $params$. $\mathcal{C}$ gives $params$ to $\mathcal{A}_I$ while keeping msk secret. After receiving $params$, $\mathcal{A}_I$ interacts with $\mathcal{C}$ in two phases:

Phase I: $\mathcal{A}_I$ is given access to all the five oracles. $\mathcal{A}_I$ adaptively queries the oracles consistent with the constraints for Type-I adversary described above.

Challenge: At the end of ***Phase I***, $\mathcal{A}_I$ gives two messages m_0 and m_1 of equal length to $\mathcal{C}$ on which it wishes to be challenged. $\mathcal{C}$ randomly chooses a bit $\delta \in_R \{0,1\}$ and encrypts m_δ with the target identity ID^*'s public key to form the challenge ciphertext σ^* and sends it to $\mathcal{A}_I$ as the challenge. (Note that the partial Private Key corresponding to ID^* should not be queried by $\mathcal{A}_I$ but the secret value corresponding to ID^* may be queried. This makes our security model stronger when compared to the security models of [11] and [18].)

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

Guess: $\mathcal{A}_I$ outputs a bit δ' at the end of the game. $\mathcal{A}_I$ wins the IND-CLE-CCA2-I game if $\delta' = \delta$. The advantage of $\mathcal{A}_I$ is defined as -

$$Adv_{\mathcal{A}_I}^{IND-CLE-CCA2-I} = |2Pr[\delta = \delta'] - 1|$$

IND-CLE-CCA2 game for Type-II Adversary: The game is named as IND-CLE-CCA2-II. This game, played between the challenger $\mathcal{C}$ and the Type-II adversary $\mathcal{A}_{II}$, is defined below:

Setup: Challenger $\mathcal{C}$ runs the setup algorithm to generate master private key msk and public parameters $params$. $\mathcal{C}$ gives $params$ and the master private key msk to $\mathcal{A}_{II}$. After receiving $params$, $\mathcal{A}_{II}$ interacts with $\mathcal{C}$ in two phases:

Phase I: $\mathcal{A}_{II}$ is not given access to the *Extract partial Private Key* oracle because $\mathcal{A}_{II}$ knows msk, it can generate the partial private key of any user in the system. All other oracles are accessible by $\mathcal{A}_{II}$. $\mathcal{A}_{II}$ adaptively queries the oracles consistent with the constraints for Type-II adversary described above.

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

Phase II: $\mathcal{A}_{II}$ adaptively queries the oracles consistent with the constraints for Type-II adversary described above. Besides this $\mathcal{A}_{II}$ cannot query *Decryption* on (σ^*, 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: $\mathcal{A}_{II}$ outputs a bit δ' at the end of the game. $\mathcal{A}_{II}$ wins the IND-CLE-CCA2-II game if $\delta' = \delta$. The advantage of $\mathcal{A}_{II}$ is defined as -

$$Adv_{\mathcal{A}_{II}}^{IND-CLE-CCA2-II} = |2Pr[\delta = \delta'] - 1|$$

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 (CCDH).

Notation: We use the notation $\mathbb{Z}_n^{odd}$ 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 $\mathbb{Z}_n$ 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 $\mathbb{Z}_n^{odd}$.

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 $H : \{0,1\}^* \rightarrow \mathbb{Z}_n^*$, $H_1 : \{0,1\}^* \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^{odd}$, $H_2 : \{0,1\}^l \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^{odd}$ and $H_3 : \mathbb{Z}_n^* \times \mathbb{Z}_n^* \times \{0,1\}^* \rightarrow \{0,1\}^{l+|\mathbb{Z}_n^{odd}|}$, where l is the size of the message.
- Now, KGC publicizes the system parameters, $params = \langle n, H, H_1, H_2, H_3 \rangle$ 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 $\mathbb{Z}_n^{odd}$ 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 $\phi(n) = 2^2p'q'$ 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 $\mathbb{Z}_n$ 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 [2] and [18]. 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 $e_A d_A \equiv 1 \ mod \ \phi(n)$ 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.

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

$$(g_A^{x_A})^{e_A} g_A \stackrel{?}{=} (g_A)^{s_A e_A} \qquad (1)$$

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 $y_A \in_R \mathbb{Z}_n^{odd}$ 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 $g_A^{D_A^{(2)}}$.
- Makes $PK_A = \langle PK_A^{(1)}, PK_A^{(2)}, PK_A^{(3)} \rangle = \langle PPK_A, g_A^{D_A^{(2)}}, g_A^{D_A^{(1)}} \rangle$ 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:

$$(PK_A^{(3)})^{e_A} \stackrel{?}{=} (PK_A^{(1)})^{e_A} g_A \tag{2}$$

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 $r \in_R \mathbb{Z}_n^{odd}$.
- 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 $c_2 = (m \| r) \oplus H_3\left((PK_A^{(1)})^{he_A}, (PK_A^{(2)})^h, ID_A\right)$.

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 $\sigma = (c_1, c_2)$:

- Retrieves $(m \| r) = H_3\left(\frac{(c_1)^{D_A^{(1)} e_A}}{c_1}, (c_1)^{D_A^{(2)}}, ID_A\right) \oplus c_2$.
- Computes $h' = H_2(m, r)$ and checks whether $c_1 \stackrel{?}{=} g_A^h$.

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

Correctness of the Public Key verification test:

$$\begin{aligned}
\text{L.H.S} = (PK_A^{(3)})^{e_A} &= g_A^{(x_A + d_A) e_A} \\
&= g_A^{x_A e_A} g_A^{d_A e_A} \\
&= g_A^{x_A e_A} g_A, \text{ since } d_A e_A \equiv 1 \ mod \ \phi(n) \\
&= (PK_A^{(1)})^{e_A} g_A = \text{R.H.S}
\end{aligned}$$

Correctness of the scheme: During decryption there are two key components in the computation of the H_3 hash function, namely $\frac{(c_1)^{D_A^{(1)} e_A}}{c_1}$ and $(c_1)^{D_A^{(2)}}$. The correctness for the first component is shown below:

$$\begin{aligned}
\frac{(c_1)^{D_A^{(1)} e_A}}{c_1} &= \frac{(g_A^h)^{(x_A + d_A) e_A}}{g_A^h} = \frac{(g_A^{x_A + d_A})^{h e_A}}{g_A^h} = \frac{g_A^{x_A h e_A + d_A h e_A}}{g_A^h} \\
&= \frac{g_A^{x_A h e_A + h}}{g_A^h}, \text{ since } d_A e_A \equiv 1 \ mod \ \phi(n) \\
&= g_A^{x_A h e_A} = (PK_A^{(1)})^{h e_A}
\end{aligned}$$

The correctness for the second component follows, because,

$$(c_1)^{D_A^{(2)}} = (g_A^h)^{y_A} = (g_A^{y_A})^h = (PK_A^{(2)})^h$$

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 [11] and [18], 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.

Confidentiality against Type-I Adversary:

Theorem 1. *Our certificateless public key encryption scheme RSA-CLE$_1$ is IND-RSA-CLE$_1$-CCA2-I 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: The challenger $\mathcal{C}$ is challenged with an instance of the RSA problem, say $\langle n, e \in_R \mathbb{Z}_n^{odd}, b\rangle \in \mathbb{Z}_n^*$, 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 $\mathcal{A}_I$ who is capable of breaking the IND-RSA-CLE$_1$-CCA2-I security of the RSA-CLE$_1$ scheme. $\mathcal{C}$ can make use of $\mathcal{A}_I$ to compute a such that $a^e \equiv b \ mod\ n$, by playing the following interactive game with $\mathcal{A}_I$.

Setup: $\mathcal{C}$ begins the game by setting up the system parameters as in the RSA-CLE$_1$ scheme. $\mathcal{C}$ takes n from the instance of the RSA problem that $\mathcal{C}$ has received and sends $params = \langle n\rangle$ to $\mathcal{A}_I$. $\mathcal{C}$ also designs the four hash functions H, H_1, H_2 and H_3 as random oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$. $\mathcal{C}$ maintains four lists L, L_1, L_2 and L_3 in order to consistently respond to the queries to the random oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$ respectively. To maintain the consistency of the private key request and public key request oracle queries, $\mathcal{C}$ maintains lists L_S and L_P respectively. A typical entity in list L_i will have the parameters of H_i (for $i = 1$ to 4) followed by the corresponding hash value returned as the response to the hash oracle query. The list L_S consists of the tuples of the form $\langle ID_i, D_i^{(1)}, PK_i^{(1)}, D_i^{(2)}\rangle$ and that of L_P consists of the tuples of the form $\langle ID_i, PK_i^{(1)}, PK_i^{(2)}, PK_i^{(3)}, H_1(ID_i, g_i^{x_i})\rangle$. The game proceeds as described in the security model for Type-I adversary in section 2.2.

Phase I: $\mathcal{A}_I$ performs a series of queries to the oracles provided by $\mathcal{C}$. The descriptions of the oracles and the responses given by $\mathcal{C}$ to the corresponding oracle queries by $\mathcal{A}_I$ are described below:

Note: We assume that $\mathcal{O}_H(.)$ oracle is queried with ID_i as input, before any other oracle is queried with the corresponding identity, ID_i as one of the inputs.

$\mathcal{O}_H(ID_i)$**:** We follow the proof methodology introduced in [3] and make a simplifying assumption that $\mathcal{A}_I$ queries the $\mathcal{O}_H$ 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 $\mathcal{A}_I$ asks q_H distinct queries for q_H distinct identities. Among this q_H identities, a random identity has to be selected as target identity by $\mathcal{C}$. $\mathcal{C}$ selects a random index γ, where $1 \leq \gamma \leq q_H$ and $\mathcal{C}$ does not reveal γ to $\mathcal{A}_I$. When $\mathcal{A}_I$ generates the γ^{th} query on ID_γ, $\mathcal{C}$ fixes ID_γ as target identity for the challenge phase.

For answering the $\mathcal{O}_H$ query, $\mathcal{C}$ performs the following, for $1 \leq \gamma \leq q_H$

- If a tuple of the form $\langle ID_i, e_i, \beta_i, g_i\rangle$ exists in the list L then $\mathcal{C}$ retrieves the corresponding g_i.
- Else,
  - If $i \neq \gamma$, $\mathcal{C}$ performs the following:
    - $\mathcal{C}$ chooses $e_i \in_R \mathbb{Z}_n^{odd}$, $\beta_i \in_R \mathbb{Z}_n^*$ and computes $g_i = \beta_i^{e_i}$.
    - Generates the partial private key corresponding to ID_i as follows:
      - Chooses $s_i \in_R \mathbb{Z}_n^{odd}$.
      - Computes $\frac{g_i^{s_i}}{\beta_i}$. Let $\frac{g_i^{s_i}}{\beta_i} = g_i^{x_i}$ for some x_i. (Note that x_i is not known to $\mathcal{C}$.)
      - Chooses $y_i \in_R \mathbb{Z}_n^{odd}$ and adds the tuple $\langle ID_i, s_i, g_i^{x_i}, y_i\rangle$ in the list L_S.
    - Adds the tuple $\langle ID_i, g_i^{x_i}, e_i\rangle$ in the list L_1.
    - Computes $g_i^{y_i}$ and $g_i^{s_i}$, adds the tuple $\langle ID_i, g_i^{x_i}, g_i^{y_i}, g_i^{s_i}, e_i\rangle$ into the list L_P.

Lemma 1 below shows that the value in the list L_1, L_S and L_P form a consistent set of private key / public key values for ID_i.

Lemma 1. *The value in the list L_1, L_S and L_P form a consistent set of private key / public key values for ID_i for all $i \neq \gamma$.*

Recall that the structure of L_S and L_P must be of the form $L_S = \langle ID_i, D_i^{(1)}, PK_i^{(1)}, D_i^{(2)}\rangle$ and $L_P = \langle ID_i, PK_i^{(1)}, PK_i^{(2)}, PK_i^{(3)}, H_1(ID_i, g_i^{x_i})\rangle$ respectively.

From L_P we infer that $PK_i^{(1)} = g_i^{x_i}$ and $H_1(ID_i, PK_i^{(1)}) = H_1(ID_i, g_i^{x_i}) = e_i$ and thus is consistent with the entry in the list L_1.

Since the L list entry corresponding to ID_i is $\langle ID_i, e_i, \beta_i, g_i\rangle$, we get $H(ID_i) = g_i$. Recall from equation (1) that the partial private key formed above will be valid set of values if they satisfy the following condition:

$$(g_i^{x_i})^{e_i} g_i \stackrel{?}{=} (g_i)^{s_i e_i} \tag{3}$$

Let d_i be such that

$$e_i d_i \equiv 1 \ mod \ phi(n) \tag{4}$$

Therefore,

$$g_i = \beta_i^{e_i} \Rightarrow \beta_i = g_i^{d_i} \tag{5}$$

Now, $\dfrac{g_i^{s_i}}{\beta_i} = \dfrac{g_i^{s_i}}{g_i^{d_i}} = g_i^{s_i - d_i} = g_i^{x_i}$ and this implies

$$s_i - d_i = x_i \tag{6}$$

Using equation (6) we can show that the validity of equation (3). In fact,

LHS $= g_i^{(s_i - d_i)e_i} g_i = g_i^{(s_i e_i)} g_i^{-1} g_i = g_i^{s_i e_i} =$ RHS

□

- If $i = \gamma$, $\mathcal{C}$ performs the following:
  * $\mathcal{C}$ chooses $\beta_i \in_R \mathbb{Z}_n^*$ and $\omega \in_R \mathbb{Z}_n^{odd}$ and computes $z = \omega^2$. Let $z = x_i^{-1} d^2$, for some x_i. Sets $e_i = e$ and computes $g_i = \beta_i^{ze_i^2}$.

    Note: It is to be noted that the tuple $\langle ID_\gamma, e_\gamma, \beta_\gamma, g_\gamma\rangle$ in the list L is equal to $\langle ID_\gamma, e, \beta_\gamma, \beta_\gamma^{ze^2}\rangle$.
  * Chooses $y_i \in_R \mathbb{Z}_n^{odd}$ and computes $PK_i = \langle PK_i^{(1)}, PK_i^{(2)}, PK_i^{(3)}\rangle = \langle \beta_i, g_i^{y_i}, \beta_i \beta_i^{ze_i}\rangle$. $\mathcal{C}$ now adds the tuple $\langle ID_i, \beta_i, g_i^{y_i}, \beta_i \beta_i^{ze_i}, e_i\rangle$ into the list L_P. The public key thus generated passes the verification test done by $\mathcal{A}_I$ as shown below:
    $(PK_i^{(3)})^{e_i} = (\beta_i \beta_i^{ze_i})^{e_i} = (\beta_i^{e_i} \beta_i^{ze_i^2}) = (PK_i^{(1)})^{e_i} g_i$ (Since $\beta_i^{ze_i^2} = g_i$)
  * Adds the tuple $\langle ID_i, g_i^{x_i} = \beta_i, e_i\rangle$ in the list L_1.
- $\mathcal{C}$ adds the tuple $\langle ID_i, e_i, \beta_i, g_i\rangle$ to the list L and returns g_i to $\mathcal{A}_I$.

$\mathcal{O}_{H_1}(ID_i, \Delta_i)$**:** To respond to this query, $\mathcal{C}$ retrieves the tuple that corresponds to ID_i, which is of the form $\langle ID_i, g_i^{x_i}, g_i^{y_i}, g_i^{s_i}, e_i\rangle$ from the list L_P and performs the following:

- If $g_i^{x_i} = \Delta_i$, a tuple of the form $\langle ID_i, \Delta_i, e_i\rangle$ will exist in the list L_1, $\mathcal{C}$ returns the corresponding e_i.
- If $g_i^{x_i} \neq \Delta_i$, $\mathcal{C}$ chooses $\hat{e}_i \in_R \mathbb{Z}_n^{odd}$, adds the tuple $\langle ID_i, \Delta_i, \hat{e}_i\rangle$ in the list L_1 and returns $\hat{e}_i$ as the response.

$\mathcal{O}_{H_2}(m, r)$**:** To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle m, r, h\rangle$ exists in the list L_2. If a tuple of this form exists, $\mathcal{C}$ returns the corresponding h, else chooses $h \in_R \mathbb{Z}_n^{odd}$, adds the tuple $\langle m, r, h\rangle$ to the list L_2 and returns h to $\mathcal{A}_I$.

$\mathcal{O}_{H_3}(k_1, k_2, ID_i)$**:** To respond to this query, $\mathcal{C}$ checks whether a tuple $\langle k_1, k_2, ID_i, h_3\rangle$ exists in the list L_3. If a tuple of this form exists, $\mathcal{C}$ returns the corresponding h_3 else chooses $h_3 \in_R \{0,1\}^{l+|\mathbb{Z}_n^{odd}|}$, adds the tuple $\langle k_1, k_2, ID_i, h_3\rangle$ to the list L_3 and returns h_3 to $\mathcal{A}_I$.

$\mathcal{O}_{PartialKeyExtract}(ID_i)$**:** To respond to this query, $\mathcal{C}$ does the following:

- If $i = \gamma$, $\mathcal{C}$ *aborts* the game.

– If $i \neq \gamma$, $\mathcal{C}$ retrieves the tuple of the form $\langle ID_i, s_i, g_i^{x_i}, y_i \rangle$ from list L_S and returns s_i as the partial private key and $PPK_i = g_i^{x_i}$ as the partial public key corresponding to the identity ID_i.

$\mathcal{O}_{ExtractSecretValue}(ID_i)$**:** $\mathcal{C}$ retrieves a tuple of the form $\langle ID_i, s_i, g_i^{x_i}, y_i \rangle$ from the list L_S and returns the corresponding y_i as the secret value corresponding to the identity ID_i. If the entry corresponding to y_i in the tuple is "−" then $\mathcal{A}_I$ has replaced the private key corresponding to ID_i.

Note: Our security model is stronger when compared to the models in [18] and [11] where the Type-I adversary is not provided the extract secret value query oracle for the target identity ID_γ. It should be noted, that the scheme in [11] is not secure if the secret value of the target identity is revealed to the Type-I adversary. We consider the model wherein the secret value corresponding to the target identity is given to the Type-I adversary $\mathcal{A}_I$, which makes it stronger.

$\mathcal{O}_{RequestPublicKey}(ID_i)$**:** $\mathcal{C}$ retrieves the tuple of the form $\langle ID_i, g_i^{x_i}, g_i^{y_i}, g_i^{s_i}, e_i \rangle$ from the list L_P and returns $PK_i = \langle \beta_i, g_i^{y_i}, \beta_i \beta_i^{e_i} \rangle$ as the public key corresponding to the identity ID_i.

$\mathcal{O}_{ReplacePublicKey}(ID_i, PK_i^{'})$**:** To replace the public key of ID_i with a new public key $PK_i^{'} = \langle PK_i^{'(1)}, PK_i^{'(2)}, PK_i^{'(3)} \rangle$, chosen by $\mathcal{A}_I$, $\mathcal{C}$ does the following:

– Updates the corresponding tuples in the list L_P as $\langle ID_i, PK_i^{'(1)}, PK_i^{'(2)}, PK_i^{'(3)}, e_i \rangle$, only if $(PK_i^{'(3)})^{e_i} = (PK_i^{'(1)})^{e_i} g_i$, where g_i corresponding to ID_i is retrieved from the list L.
– Return $Invalid$, otherwise.

$\mathcal{O}_{StrongDecryption}(\sigma, ID_i, PK_i)$: This oracle provides the decryption of a ciphertext, which is generated with the current valid public key. It should be noted that the strong decryption oracle returns $Invalid$, if the ciphertext corresponds to any of the previous public keys. *"Giving access to the secret value corresponding to the target identity for a Type-I adversary, captures the scenario where the user secret value of the target identity is compromised (some how comes to know) by the adversary"*. Hence it is a stronger type of adversary. $\mathcal{C}$ performs the following to decrypt the ciphertext $\sigma = \langle c_1, c_2 \rangle$:

– $\mathcal{C}$ performs the following to decrypt the ciphertext $\sigma = \langle c_1, c_2 \rangle$:
  - Checks the validity of PK_i and rejects the ciphertext σ if this check fails, else proceeds with the following steps.
  - Retrieves the tuple $\langle ID_i, g_i^{x_i}, e_i \rangle$ from list L.
  - For each $\langle m, r, h \rangle \in L_2$ list performs the following:
    * Checks whether $g_i^h \stackrel{?}{=} c_1$.
    * If `True`, computes $k_1 = (PK_i^{(1)})^{e_i h}$ and $k_2 = (PK_i^{(2)})^h$.
    * 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 $c_2 \oplus h_3 \stackrel{?}{=} (m \| r)$, 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 $Invalid$.

Challenge: At the end of ***Phase I***, $\mathcal{A}_I$ produces two messages m_0 and m_1 of equal length and an identity ID^*. $\mathcal{C}$ *aborts* the game if $ID^* \neq ID_\gamma$, else randomly chooses a bit $\delta \in_R \{0,1\}$ and computes a ciphertext σ^* with ID_γ as the receiver by performing the following steps:

– Set $c_1^* = b^z$, where b is taken from the RSA problem instance received by $\mathcal{C}$ and z is the value chosen during the $\mathcal{O}_H(.)$ oracle query corresponding to ID_γ.
– Choose $c_2^* \in_R \{0,1\}^{l+|\mathbb{Z}_n^{odd}|}$.

Now, $\sigma^* = \langle c_1^*, c_2^* \rangle$ is sent to $\mathcal{A}_I$ as the challenge ciphertext. It should be noted that with overwhelming probability, σ^* is a invalid ciphertext and since $\mathcal{A}_I$ is disallowed to query the strong decryption oracle with σ^* as input, $\mathcal{A}_I$ will not be able to identity whether σ^* is valid or not.

Phase II: $\mathcal{A}_I$ performs the second phase of interaction, where it makes polynomial number of queries to the oracles provided by $\mathcal{C}$ with the following conditions:

– $\mathcal{A}_I$ should not have queried the *Strong Decryption* oracle with $(\sigma^*, PK_\gamma, ID_\gamma)$ as input. (It is to be noted that PK_γ is the public key corresponding to ID_γ during the challenge phase. $\mathcal{A}_I$ can query the decryption oracle with $(\sigma^*, PK^*, ID_\gamma)$ as input, $\forall PK^* \neq PK_\gamma$)

- $\mathcal{A}_I$ should not query the partial private key corresponding to ID_γ.
- $\mathcal{A}_I$ can query the secret value corresponding to ID_γ and PK_γ.

Guess: At the end of ***Phase II***, $\mathcal{A}_I$ produces a bit δ' to $\mathcal{C}$, but $\mathcal{C}$ ignores the response and performs the following to output the solution for the RSA problem instance.

- For each tuple of the form $\langle k_1, k_2, ID_i, h_3\rangle$ in list L_3, $\mathcal{C}$ checks whether $k_1^e \stackrel{?}{=} b$. (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.

Correctness: Below, we show that the k_1 value obtained through the above steps is indeed a, such that $b = a^e$ *mod n*

- The public key corresponding to ID_γ is set to be $PK_\gamma = \langle PK_\gamma^{(1)}, PK_\gamma^{(2)}, PK_\gamma^{(3)}\rangle = \langle \beta_\gamma, g_\gamma^{y_\gamma}, \beta_\gamma \beta_\gamma^{ze}\rangle$ by $\mathcal{C}$. Since $g_\gamma = \beta_\gamma^{ze^2}$, $\beta_\gamma = g_\gamma^{z^{-1}e^{-2}} = g_\gamma^{z^{-1}d^2}$ (because $d \equiv e^{-1} \; mod \; \phi(n)$). Thus $PK_\gamma = \langle g_\gamma^{z^{-1}d^2}, g_\gamma^{y_\gamma}, g_\gamma^{z^{-1}d^2} g_\gamma^d\rangle$.
- The partial private key corresponding to this public key is $s_\gamma = D_\gamma^{(1)} = x_\gamma + d_\gamma = z^{-1}d^2 + d$ which is unknown to $\mathcal{C}$. This is due to the following facts:

  $$\begin{aligned} PK_\gamma^{(3)} &= \beta_\gamma \beta_\gamma^{ze} = g_\gamma^{z^{-1}d^2} g_\gamma^d \\ &= g_\gamma^{z^{-1}d^2+d} = H(ID_\gamma)^{z^{-1}d^2+d}, \text{ (since } H(ID_\gamma) = g_\gamma) \end{aligned}$$

  $$\begin{aligned} PK_\gamma^{(1)} &= \beta_\gamma = g_\gamma^{z^{-1}d^2}, \text{ (since } g_\gamma = \beta_\gamma^{ze^2}) \\ &= H(ID_\gamma)^{z^{-1}d^2}, \text{ (since } H(ID_\gamma) = g_\gamma) \end{aligned}$$

  We know that $PK_\gamma^{(3)} = H(ID_\gamma)^{x_\gamma + d_\gamma}$ and $PK_\gamma^{(1)} = H(ID_\gamma)^{x_\gamma}$. Thus, $x_\gamma = d^2$ and $d_\gamma = d$.
- $\mathcal{C}$ has set the c_1^* component of the challenge ciphertext σ^* as "b^z" (where b is taken from the RSA problem instance) during the challenge phase.
- In order to decrypt the cipher text σ^*, $\mathcal{A}_I$ should have computed a value $\frac{(c_1^*)^{D_\gamma^{(1)} e_\gamma}}{c_1^*}$ and queried the H_3 oracle with it as the k_1 component. (It should be further noted that $\mathcal{A}_I$ could make a number of queries to the $\mathcal{O}_{H_2}$ oracle and see if the resulting h satisfies $c_1^* = (g_\gamma^h)$ but this cannot be true because choosing the correct r will be negligible. Even if r has to be obtained from σ^*, $\mathcal{A}_I$ should have computed the $\mathcal{O}_{H_3}$ oracle.)
- We show that $\frac{(c_1^*)^{D_\gamma^{(1)} e_\gamma}}{c_1^*} = a$, such that $a^e \; mod \; n = b$. (Here e, b and n are the elements from the RSA problem instance.) It is known that $D_\gamma^{(1)} = z^{-1}d^2 + d$ and $e_\gamma = e$. Therefore,

  $$\frac{(c_1^*)^{D_\gamma^{(1)} e_\gamma}}{c_1^*} = \frac{(b^z)^{(z^{-1}d^2+d)e}}{b^z} = \frac{(b^z)^{(z^{-1}d+1)}}{b^z} = b^d = a \text{ (Since } d \equiv e^{-1} \; mod \; \phi(n))$$

Thus, $\mathcal{C}$ obtains the solution to the RSA problem with almost the same advantage of $\mathcal{A}_I$ in the IND-RSA-CLE$_1$-CCA2-I game. □

Analysis: We now derive the advantage of $\mathcal{C}$ breaking the RSA problem using the adversary $\mathcal{A}_I$. The simulations of H, H_1, H_2 and H_3 clearly shows that the hash oracles are perfectly random. Let ϵ be the advantage of $\mathcal{A}_I$ in winning the IND-RSA-CLE$_1$-CCA2-I game.

The events in which $\mathcal{C}$ aborts the game and the respective probabilities are given below:

1. $\mathcal{E}_1$ - The event in which $\mathcal{C}$ *aborts* when $\mathcal{A}_I$ queries the partial private key corresponding to ID_γ.
2. $\mathcal{E}_2$ - The event in which ID_γ is not chosen as the target identity by $\mathcal{A}_I$ for the challenge.

Suppose $\mathcal{A}_I$ has made q_H number of $\mathcal{O}_H$ queries and q_{ppk} number of $\mathcal{O}_{PartialKeyExtract}$ queries, then: Pr[$\mathcal{E}_1$]= $\frac{q_{ppk}}{q_H}$ and Pr[$\mathcal{E}_2$]= $1 - \frac{1}{q_H - q_{ppk}}$.

Therefore, Pr[$\neg abort$]=[$\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2$]= $\left[1 - \frac{q_{ppk}}{q_H}\right] . \left[1 - 1 - \frac{1}{q_H - q_{ppk}}\right] = \frac{1}{q_H}$.

Therefore, the advantage of $\mathcal{C}$ solving the RSA problem is $\epsilon' \geq \left(\epsilon . \frac{1}{q_H}\right)$.

Confidentiality against Type-II Adversary: The master private key of the RSA-CLE$_1$ scheme are the prime factors p and q of the composite modulus n. Since the Type-II adversary in CLE should be given access to the master private key, the security against Type-II adversary of the RSA-CLE$_1$ scheme cannot be reduced to RSA assumption. The situation in the RSA based scheme in [11] is also the same. That is why, the scheme in [11] as well as RSA-CLE$_1$, the Type-II security is related to CCDH problem. However, in our next scheme RSA-CLE$_2$ the security against Type-II adversary is based on RSA problem.

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

Proof: Suppose an adversary $\mathcal{A}_{II}$ is capable of breaking the IND-RSA-CLE$_1$-CCA2-II security of our RSA-CLE$_1$ scheme and a challenger $\mathcal{C}$ is challenged with an instance of the CCDH problem say $p, q, n, \langle g, g^a, g^b \rangle \in \mathbb{Z}_n^*$, where n is a composite number with two big prime factors p and q also $(p-1)/2$ and $(q-1)/2$ are primes. $\mathcal{C}$ can make use of $\mathcal{A}_{II}$ to compute g^{ab}, by playing the following interactive game with $\mathcal{A}_{II}$.

Setup: $\mathcal{C}$ begins the game by setting up the system parameters as in the RSA-CLE$_1$ scheme. $\mathcal{C}$ takes p, q, n and g as in the instance of the CCDH problem and sends $params = \langle n \rangle$ and p, q as the master private key msk to $\mathcal{A}_{II}$. $\mathcal{C}$ also designs the four hash functions H, H_1, H_2 and H_3 as random oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$. $\mathcal{C}$ maintains four lists L, L_1, L_2 and L_3 in order to consistently respond to the queries to the random oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$ respectively and to maintain the consistency of the private key request and public key request oracles, $\mathcal{C}$ maintains lists L_S and L_P respectively.

Phase I: $\mathcal{A}_{II}$ performs a series of queries to the oracles provided by $\mathcal{C}$. The descriptions of the oracles and the responses given by $\mathcal{C}$ to the corresponding oracle queries by $\mathcal{A}_{II}$ are described below. We assume that $\mathcal{O}_H(.)$ oracle is queried with ID_i as input, before any other oracle is queried with the corresponding identity, ID_i as one of the input parameters.

$\mathcal{O}_H(ID_i)$**:** We will make a simplifying assumption that $\mathcal{A}_{II}$ queries the $\mathcal{O}_H$ oracle with distinct identities in each query. If the same identity is repeatedly queried to this oracle, by definition, the oracle consults the list L and gives the same response. Thus, we assume that $\mathcal{A}_{II}$ asks q_H distinct queries for q_H distinct identities. $\mathcal{C}$ selects a random index γ, where $1 \leq \gamma \leq q_H$ and $\mathcal{C}$ does not reveal γ to $\mathcal{A}_{II}$. When $\mathcal{A}_{II}$ generates the γ^{th} query on ID_γ, $\mathcal{C}$ fixes ID_γ as target identity for the challenge phase.

In order to answer a query to the $\mathcal{O}_H$ oracle, $\mathcal{C}$ checks whether a tuple of the form $\langle ID_i, g_i \rangle$ exists in the list L and if a tuple of this form exists, $\mathcal{C}$ returns the corresponding g_i. If it does not exist, $\mathcal{C}$ checks whether $i \stackrel{?}{=} \gamma$:

- If $i \neq \gamma$, $\mathcal{C}$ chooses $g_i \in_R \mathbb{Z}_n^*$, adds the tuple $\langle ID_i, g_i \rangle$ to the list L and returns g_i to $\mathcal{A}_{II}$.
- If $i = \gamma$, $\mathcal{C}$ sets $g_i = g$ (where g is taken from the CCDH instance), adds the tuple $\langle ID_i, g_i \rangle$ to the list L and returns g_i to $\mathcal{A}_{II}$.

$\mathcal{O}_{H_1}(ID_i, g_i^{x_i})$**:** To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle ID_i, g_i^{x_i}, e_i \rangle$ exists in the list L_1. If it exists, $\mathcal{C}$ returns the corresponding e_i, else $\mathcal{C}$ chooses $e_i \in_R \mathbb{Z}_n^{odd}$, adds the tuple $\langle ID_i, g_i^{x_i}, e_i \rangle$ to the list L_1 and returns e_i to $\mathcal{A}_{II}$.

$\mathcal{O}_{H_2}(m, r)$**:** To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle m, r, h \rangle$ exists in the list L_2. If a tuple of this form exists, $\mathcal{C}$ returns the corresponding h else chooses $h \in_R \mathbb{Z}_n^{odd}$, adds the tuple $\langle m, r, h \rangle$ to the list L_2 and returns h to $\mathcal{A}_{II}$.

$\mathcal{O}_{H_3}(k_1, k_2, ID_i)$**:** To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle k_1, k_2, ID_i, h_3 \rangle$ exists in the list L_3. If a tuple of this form exists, $\mathcal{C}$ returns the corresponding h_3 else chooses $h_3 \in_R \{0,1\}^{l+|\mathbb{Z}_n^{odd}|}$, adds the tuple $\langle k_1, k_2, ID_i, h_3 \rangle$ to the list L_3 and returns h_3 to $\mathcal{A}_{II}$.

$\mathcal{O}_{PartialKeyExtract}(ID_i)$**:** To respond to this query, $\mathcal{C}$ does the following: $\mathcal{C}$ checks whether a tuple of the form $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, y_i \rangle$ exists in list L_S.

- If it exists then, $\mathcal{C}$ outputs the corresponding s_i as the partial private key and $PPK_i = g_i^{x_i}$ as the partial public key corresponding to the identity ID_i.
- If a tuple does not exist then, $\mathcal{C}$ performs the following:
  - Chooses $x_i, e_i \in_R \mathbb{Z}_n^{odd}$.
  - Retrieves the tuple $\langle ID_i, g_i \rangle$ from the list L.

- Computes $g_i^{x_i}$, $d_i = e_i^{-1} \ mod\ \phi(n)$ (Note that this is possible because $\mathcal{C}$ knows the prime factors of n) and $s_i = x_i + d_i$
- Stores the tuple $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, - \rangle$ in the list L_S and the tuple $\langle ID_i, g_i^{x_i}, e_i \rangle$ in the list L_1.
- Returns s_i as the partial private key and $PPK_i = g_i^{x_i}$ as the partial public key corresponding to the identity ID_i.

$\mathcal{O}_{ExtractSecretValue}(ID_i)$**:** $\mathcal{C}$ checks whether $i \stackrel{?}{=} \gamma$:

- If $i = \gamma$ then, $\mathcal{C}$ *aborts* the game.
- If $i \neq \gamma$ then, $\mathcal{C}$ checks for a tuple of the form $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, y_i \rangle$ in list L_S and returns the corresponding y_i as the secret value corresponding to the identity ID_i. If the entry corresponding to y_i in the tuple is $'' - ''$ then, $\mathcal{C}$ chooses $y_i \in_R \mathbb{Z}_n^{odd}$, updates the corresponding tuple in the list L_S as $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, y_i \rangle$ and returns y_i to $\mathcal{A}_{II}$.

$\mathcal{O}_{RequestPublicKey}(ID_i)$**:** $\mathcal{C}$ checks for an entry of the form $\langle ID_i, g_i^{x_i}, g_i^{y_i}, g_i^{s_i} \rangle$ in list L_P and performs the following accordingly:

- If an entry of this form exists, $\mathcal{C}$ returns $\langle g_i^{x_i}, g_i^{y_i}, g_i^{s_i} \rangle$ as the public key corresponding to the identity ID_i.
- If no tuple exists then, $\mathcal{C}$ checks whether $i \stackrel{?}{=} \gamma$:
  - If $i \neq \gamma$, $\mathcal{C}$ retrieves the tuple $\langle ID_i, g_i \rangle$ from the list L and the tuple $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, y_i \rangle$ from the list L_S and computes $g_i^{y_i}$ and $g_i^{s_i}$, adds the tuple $\langle ID_i, g_i^{x_i}, g_i^{y_i}, g_i^{s_i} \rangle$ into the list L_P and returns $\langle g_i^{x_i}, g_i^{y_i}, g_i^{s_i} \rangle$ as the public key corresponding to the identity ID_i.
  - If $i = \gamma$, $\mathcal{C}$ retrieves the tuple $\langle ID_i, g_i \rangle$ from the list L and the tuple $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, - \rangle$ from the list L_S. $\mathcal{C}$ sets $PK_i = \langle PK_i^{(1)}, PK_i^{(2)}, PK_i^{(3)} \rangle = \langle g_i^{x_i}, g^a, g_i^{s_i} \rangle$. $\mathcal{C}$ now adds the tuple $\langle ID_i, g_i^{x_i}, g^a, g_i^{s_i} \rangle$ into the list L_P and returns $\langle g_i^{x_i}, g^a, g_i^{s_i} \rangle$ as the public key corresponding to the identity ID_i. (Note that g^a is taken from the CCDH problem instance and $g_i = g$, set during the $\mathcal{O}_H(ID_i)$ query.)

$\mathcal{O}_{ReplacePublicKey}(ID_i, PK_i^{'})$**:** To replace the public key of ID_i with a new public key $PK_i^{'} = \langle PK_i^{'(1)}, PK_i^{'(2)}, PK_i^{'(3)} \rangle$, chosen by $\mathcal{A}_{II}$, $\mathcal{C}$ checks whether $i \stackrel{?}{=} \gamma$ and does the following:

- If $i = \gamma$ then, $\mathcal{C}$ *aborts* the game.
- If $i \neq \gamma$ then, updates the corresponding tuples in the list L_P as $\langle ID_i, PK_i^{'(1)}, PK_i^{'(2)}, PK_i^{'(3)} \rangle$, only if $(PK_i^{'(3)})^{e_i} = (PK_i^{'(1)})^{e_i} g_i$, where g_i corresponding to ID_i is retrieved from the list L.

Note: The replace public key oracle for Type-II adversary was not considered in both [18] and [11].

$\mathcal{O}_{Decryption}(\sigma, ID_i, PK_i)$: $\mathcal{C}$ performs the following to decrypt the ciphertext $\sigma = \langle c_1, c_2 \rangle$:

- If $i \neq \gamma$, $\mathcal{C}$ performs decryption in the normal way since $\mathcal{C}$ knows the private key corresponding to ID_i.
- If $i = \gamma$, $\mathcal{C}$ performs the following to decrypt the ciphertext $\sigma = \langle c_1, c_2 \rangle$:
  - Retrieves the tuple $\langle ID_i, g_i^{x_i}, e_i \rangle$ from list L_1.
  - For each $\langle m, r, h \rangle \in L_2$ list performs the following
    - Checks whether $g_i^h \stackrel{?}{=} c_1$.
    - If `True`, computes $k_1 = (PK_i^{(1)})^{e_i h}$ and $k_2 = (PK_i^{(2)})^h$.
    - 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 $c_2 \oplus h_3 \stackrel{?}{=} (m \| r)$, where m, r is retrieved from the list L_2.
    - If `True`, outputs m as the message.
  - If no tuple satisfies all the above tests, returns *Invalid*.

Challenge: At the end of ***Phase I***, $\mathcal{A}_{II}$ gives $\mathcal{C}$ two messages m_0, m_1 of equal length. $\mathcal{C}$ *aborts* the game if $ID^* \neq ID_\gamma$, else $\mathcal{C}$ randomly chooses a bit $\delta \in_R \{0,1\}$ and computes a ciphertext σ^* with ID_γ as the receiver by performing the following steps:

- Set $c_1^* = g^b$. (Where g^b is taken from the CCDH instance.)
- Choose $c_2^* \in_R \{0,1\}^{l+|\mathbb{Z}_n^{odd}|}$.

Now, $\sigma^* = \langle c_1^*, c_2^* \rangle$ is sent to $\mathcal{A}_{II}$ as the challenge ciphertext.

Phase II: Again $\mathcal{A}_{II}$ can perform polynomially bounded number of queries to the oracles provided by $\mathcal{C}$ with the following conditions:

- $\mathcal{A}_{II}$ should not have queried the decryption oracle with $(\sigma^*, PK_\gamma, ID_\gamma)$ as input. (It is to be noted that PK_γ is the public key corresponding to ID_γ during the challenge phase. $\mathcal{A}_{II}$ is allowed to query the decryption oracle with $(\sigma^*, PK^*, ID_\gamma)$ as input, $\forall PK^* \neq PK_\gamma$).
- $\mathcal{A}_{II}$ should not have queried the secret value corresponding to ID_γ.
- $\mathcal{A}_{II}$ should not have replaced the public key corresponding to the identity ID_γ.

Guess: At the end of ***Phase II***, $\mathcal{A}_{II}$ produces a bit δ' to $\mathcal{C}$ but $\mathcal{C}$ ignores the response and performs the following to output the solution to the CCDH problem instance.

- Randomly picks a k_2 value from the list L_3 and outputs it as the solution to the CCDH problem instance.

Correctness: Below, we show that the k_2 value obtained through the above step is indeed g^{ab}.

- The public key component $PK_\gamma^{(2)}$ corresponding to ID_γ is set to be g^a by $\mathcal{C}$ during the request public key query and thus $H(ID_\gamma)^{y_\gamma} = g^a$.
- Since $H(ID_\gamma) = g_\gamma = g$, $D_\gamma^{(2)} = y_\gamma = a$
- $\mathcal{C}$ has set the c_1^* component of the challenge ciphertext σ^* as g^b during the challenge phase.
- In order to decrypt the ciphertext σ^*, $\mathcal{A}_{II}$ should have computed a value $(c_1^*)^{D_\gamma^{(2)}}$ and queried the H_3 oracle with it as the k_2 component.
- Now, $(c_1^*)^{D_\gamma^{(2)}} = (g^b)^a = g^{ab}$.

Thus, $\mathcal{C}$ obtains the solution to the CCDH problem with almost the same advantage of $\mathcal{A}_{II}$ in the IND-RSA-CLE$_1$-CCA2-II game. □

Analysis: We now derive the advantage of $\mathcal{C}$ breaking the CCDH problem using the adversary $\mathcal{A}_{II}$. The simulations of H, H_1, H_2 and H_3 clearly shows that the hash oracles are perfectly random. Let ϵ be the advantage of $\mathcal{A}_{II}$ in winning the IND-RSA-CLE$_1$-CCA2-I game.

The events in which $\mathcal{C}$ aborts the game and the respective probabilities are given below:

1. $\mathcal{E}_1$ - The event in which $\mathcal{C}$ *aborts* when $\mathcal{A}_{II}$ queries the secret value corresponding to ID_γ.
2. $\mathcal{E}_2$ - The event in which $\mathcal{C}$ *aborts* when $\mathcal{A}_{II}$ replaces the public key corresponding to ID_γ.
3. $\mathcal{E}_3$ - The event in which ID_γ is not chosen as the target identity by $\mathcal{A}_{II}$ for the challenge.

Suppose $\mathcal{A}_{II}$ has made q_H number of $\mathcal{O}_H$ queries, q_{sv} number of $\mathcal{O}_{ExtractSecretValue}$ queries, q_{rpk} number of identities for which $\mathcal{O}_{ReplacePublicKey}$ queries and q_{srk} be the total number of identities for which the secret value is extracted and the public key is replaced, then:
$\Pr[\mathcal{E}_1]= \frac{q_{sv}}{q_H}$, $\Pr[\mathcal{E}_2]=\frac{q_{rpk}}{q_H}$ and $\Pr[\mathcal{E}_3]= 1 - \frac{1}{q_H - q_{sv}}$.
Therefore,

$$\Pr[\neg abort]=[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2 \wedge \neg\mathcal{E}_3]$$
$$=\left[1 - \frac{q_{sv}}{q_H}\right].\left[1 - \frac{q_{rpk}}{q_H}\right].\left[1 - \left[1 - \frac{1}{q_H - q_{srk}}\right]\right] =\left[1 - \frac{q_{sv}}{q_H}\right].\left[1 - \frac{q_{rpk}}{q_H}\right].\left[\frac{1}{q_H - q_{srk}}\right].$$

Therefore, the advantage of $\mathcal{C}$ solving the CCDH problem is $\epsilon' \geq \left(\epsilon.\left[1 - \frac{q_{sv}}{q_H}\right].\left[1 - \frac{q_{rpk}}{q_H}\right].\left[\frac{1}{q_H - q_{srk}}\right]\right)$.

4 Fully RSA Based CLE Scheme (RSA-CLE$_2$)

In this section, we propose the fully RSA based certificateless encryption scheme RSA-CLE$_2$. The Type-I security is similar to that of the Type-I security proof of RSA-CLE$_1$. 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-CLE$_2$ 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 $H : \{0,1\}^* \rightarrow \mathbb{Z}_n^*$, $H_1 : \{0,1\}^* \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^{odd}$, $H_2 : \{0,1\}^l \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^{odd}$ and $H_3 : \mathbb{Z}_n^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^{l+|\mathbb{Z}_n^*|}$, where l is the size of the message.
- Now, KGC publicizes the system parameters, $params = \langle n, H, H_1, H_2, H_3 \rangle$ 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 $e_A d_A \equiv 1\ mod\ \phi(n)$ 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 $N_A = P_A Q_A$ and the Euler's totient function $\phi(N_A) = (P_A - 1)(Q_A - 1)$.
- Chooses $\hat{e}_A \in_R \mathbb{Z}_{N_A}^{odd}$ as the user public key and computes $\hat{d}_A \equiv \hat{e}_A^{-1}\ mod\ \phi(N_A)$.
- Sets the private key as $D_A = \langle D_A^{(1)}, D_A^{(2)}, D_A^{(3)}, D_A^{(4)} \rangle = \langle s_A, \hat{d}_A, P_A, Q_A \rangle$.

Set Public Key: The user with identity ID_A computes the public key corresponding to his private key as $PK_A = \langle PK_A^{(1)}, PK_A^{(2)}, PK_A^{(3)}, PK_A^{(4)} \rangle = \langle PPK_A, g_A^{D_A^{(1)}}, \hat{e}_A, N_A \rangle$ 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 $(PK_A^{(2)})^{e_A} \stackrel{?}{=} (PK_A^{(1)})^{e_A} g_A$

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 $r \in_R \mathbb{Z}_n^{odd}$ and $\hat{g} \in_R \mathbb{Z}_{N_A}^*$.
- Compute $e_A = H_1(ID_A, PK_A^{(1)})$, $g_A = H(ID_A)$ and $h = H_2(m, r)$.
- Compute $c_1 = g_A^h$, $c_2 = \hat{g}^{PK_A^{(3)}}\ mod\ N_A$ and $c_3 = (m \| r) \oplus H_3\left((PK_A^{(1)})^{he_A}, \hat{g}, ID_A\right)$.

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 $k_1 = \dfrac{(c_1)^{D_A^{(1)} e_A}}{c_1}$ and $k_2 = (c_2)^{D_A^{(2)}}\ mod\ N_A$.
- Retrieves $(m\|r) = H_3(k_1, k_2, ID_A) \oplus c_3$.
- Computes $h' = H_2(m, r)$ and checks whether $c_1 \stackrel{?}{=} g_A^h$.

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

<u>*Correctness of the scheme:*</u> During decryption there are two key components in the computation of the H_3 hash function, namely k_1 and k_2. The correctness for the first component is shown below:

$$k_1 = \frac{(c_1)^{D_A^{(1)} e_A}}{c_1} = \frac{(g_A^h)^{(x_A + d_A)e_A}}{g_A^h} = \frac{(g_A^{x_A + d_A})^{he_A}}{g_A^h} = \frac{g_A^{x_A h e_A + d_A h e_A}}{g_A^h}$$
$$= \frac{g_A^{x_A h e_A + h}}{g_A^h} \text{ (since } d_A e_A \equiv 1\ mod\ \phi(n))$$
$$= g_A^{x_A h e_A} = (PK_A^{(1)})^{he_A}$$

The correctness for the second component follows, because,

$$k_2 = (c_1)^{D_A^{(2)}}\ mod\ N_A = (\hat{g}^{\hat{e}_A})^{\hat{d}_A}\ mod\ N_A = \hat{g} \text{ (Since } \hat{e}_A \hat{d}_A \equiv 1\ mod\ \phi(N_A))$$

Confidentiality against Type-I Adversary:

Theorem 3. *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 $\mathbb{Z}_n^*$, 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).

Confidentiality against Type-II Adversary:

Theorem 4. *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 $\mathbb{Z}_N^*$, where $N = PQ$ and P, Q, $(P-1)/2$, $(Q-1)/2$ are large prime numbers.*

Proof: Suppose an adversary $\mathcal{A}_{II}$ is capable of breaking the IND-RSA-CLE$_2$-CCA2-II security of our RSA-CLE$_2$ scheme and a challenger $\mathcal{C}$ is challenged with an instance of the RSA problem say $\langle N, \hat{e}, b\rangle$, where N is a composite number with two big prime factors P and Q, $(P-1)/2$ and $(Q-1)/2$ are also primes, $\hat{e} \in_R \mathbb{Z}_N^{odd}$ and $b \in \mathbb{Z}_N^*$. $\mathcal{C}$ can make use of $\mathcal{A}_{II}$ to compute a, such that $a^{\hat{e}} \equiv b \ mod \ N$, by playing the following interactive game with $\mathcal{A}_{II}$. It is to be noted that both P and Q are not known to $\mathcal{C}$.

Setup: $\mathcal{C}$ begins the game by setting up the system parameters as in the IND-RSA-CLE$_2$-CCA2-II scheme. $\mathcal{C}$ chooses two big primes p and q, computes $n = pq$ and sends $params = \langle n\rangle$ and, p and q as the master private key to $\mathcal{A}_{II}$. $\mathcal{C}$ also designs the four hash functions H, H_1, H_2 and H_3 as random oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$. $\mathcal{C}$ maintains three lists L, L_1, L_2 and L_3 in order to consistently respond to the queries to the random oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$ and $\mathcal{O}_{H_3}$ respectively and to maintain the consistency of the private key request and public key request oracles, $\mathcal{C}$ maintains lists L_S and L_P respectively.

Phase I: $\mathcal{A}_{II}$ performs a series of queries to the oracles provided by $\mathcal{C}$. The descriptions of the oracles and the responses given by $\mathcal{C}$ to the corresponding oracle queries by $\mathcal{A}_{II}$ are described below. We assume that $\mathcal{O}_H(.)$ oracle is queried with ID_i as input, before any other oracle is queried with the corresponding identity, ID_i as one of the input parameters.

$\mathcal{O}_H(ID_i)$**:** We will make a simplifying assumption that $\mathcal{A}_{II}$ queries the $\mathcal{O}_H$ oracle with distinct identities in each query. If the same identity is repeatedly queried to this oracle, by definition, the oracle consults the list L and gives the same response. Thus, we assume that $\mathcal{A}_{II}$ asks q_H distinct queries for q_H distinct identities. $\mathcal{C}$ selects a random index γ, where $1 \leq \gamma \leq q_H$ and $\mathcal{C}$ does not reveal γ to $\mathcal{A}_{II}$. When $\mathcal{A}_{II}$ generates the γ^{th} query on ID_γ, $\mathcal{C}$ fixes ID_γ as target identity for the challenge phase.

In order to answer a query to the $\mathcal{O}_H$ oracle, $\mathcal{C}$ checks whether a tuple of the form $\langle ID_i, g_i\rangle$ exists in the list L and if a tuple of this form exists, $\mathcal{C}$ returns the corresponding g_i. If it does not exist, $\mathcal{C}$ chooses $g_i \in_R \mathbb{Z}_n^*$, adds the tuple $\langle ID_i, g_i\rangle$ to the list L and returns g_i to $\mathcal{A}_{II}$.

$\mathcal{O}_{H_1}(ID_i, g_i^{x_i})$**:** To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle ID_i, g_i^{x_i}, e_i\rangle$ exists in the list L_1. If it exists, $\mathcal{C}$ returns the corresponding e_i, else $\mathcal{C}$ chooses $e_i \in_R \mathbb{Z}_n^{odd}$, adds the tuple $\langle ID_i, g_i^{x_i}, e_i\rangle$ to the list L_1 and returns e_i to $\mathcal{A}_{II}$.

$\mathcal{O}_{H_2}(m, r)$**:** To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle m, r, h\rangle$ exists in the list L_2. If a tuple of this form exists, $\mathcal{C}$ returns the corresponding h else chooses $h \in_R \mathbb{Z}_n^{odd}$, adds the tuple $\langle m, r, h\rangle$ to the list L_2 and returns h to $\mathcal{A}_{II}$.

$\mathcal{O}_{H_3}(k_1, k_2, ID_i)$**:** To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle k_1, k_2, ID_i, h_3\rangle$ exists in the list L_3. If a tuple of this form exists, $\mathcal{C}$ returns the corresponding h_3 else retrieves N_i corresponding to ID_i from the list L_P, chooses $h_3 \in_R \{0,1\}^{l+|\mathbb{Z}_{N_i}^*|}$, adds the tuple $\langle k_1, k_2, ID_i, h_3\rangle$ to the list L_3 and returns h_3 to $\mathcal{A}_{II}$.

$\mathcal{O}_{PartialKeyExtract}(ID_i)$**:** To respond to this query, $\mathcal{C}$ does the following: $\mathcal{C}$ checks whether a tuple of the form $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, \hat{d}_i, P_i, Q_i\rangle$ exists in list L_S.

- If it exists then, $\mathcal{C}$ outputs the corresponding s_i as the partial private key and $PPK_i = g_i^{x_i}$ as the partial public key of the identity ID_i.
- If a tuple does not exist then, $\mathcal{C}$ performs the following:
  - Chooses $x_i, e_i \in_R \mathbb{Z}_n^{odd}$.
  - Retrieves the tuple $\langle ID_i, g_i\rangle$ from the list L.

- Computes $g_i^{x_i}$, $d_i = e_i^{-1} \ mod \ \phi(n)$ (Note that this is possible because $\mathcal{C}$ knows the prime factors of n) and $s_i = x_i + d_i$
- Stores the tuple $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, -, -, -, \rangle$ in the list L_S and the tuple $\langle ID_i, g_i^{x_i}, e_i\rangle$ in the list L_1.
- Stores the tuple $\langle ID_i, g_i^{x_i}, g_i^{s_i}, e_i, -, -, \rangle$ in the list L_P.
- Returns s_i as the partial private key and $PPK_i = g_i^{x_i}$ as the partial public key corresponding to the identity ID_i.

$\mathcal{O}_{ExtractSecretValue}(ID_i)$**:** $\mathcal{C}$ checks whether $i \stackrel{?}{=} \gamma$:

- If $i = \gamma$ then, $\mathcal{C}$ *aborts* the game.
- If $i \neq \gamma$ then, $\mathcal{C}$ checks for a tuple of the form $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, \hat{d}_i, P_i, Q_i\rangle$ in list L_S and returns the corresponding $\hat{d}_i, P_i, Q_i$ as the secret values corresponding to the identity ID_i. If the entries corresponding to $(\hat{d}_i, P_i, Q_i)$ in the tuple are $(''-'', ''-'', ''-'')$ then, $\mathcal{C}$ performs the following:
  - Chooses two big primes P_i and Q_i and computes $N_i = P_i Q_i$,
  - Chooses $\hat{d}_i \in_R \mathbb{Z}^*_{N_i}$ and computes $\hat{e}_i \equiv \hat{d}_i^{-1} \ mod \ \phi(N_i)$,
  - Updates the corresponding tuple in the list L_P as $\langle ID_i, g_i^{x_i}, g_i^{s_i}, e_i, \hat{e}_i, N_i, \rangle$,
  - Updates the corresponding tuple in the list L_S as $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, \hat{d}_i, P_i, Q_i\rangle$ and
  - Returns $(\hat{d}_i, P_i, Q_i)$ as the secret values to $\mathcal{A}_{II}$.

$\mathcal{O}_{RequestPublicKey}(ID_i)$**:** We assume that this query is executed only after $\mathcal{O}_{PartialKeyExtract}(ID_i)$ and $\mathcal{O}_{ExtractSecretValue}(ID_i)$ queries. $\mathcal{C}$ performs the following to respond to this query:

- If $i \neq \gamma$ then an entry of the form $\langle ID_i, g_i^{x_i}, g_i^{s_i}, e_i, \hat{e}_i, N_i, \rangle$ must exist in the list L_P, $\mathcal{C}$ returns $\langle g_i^{x_i}, g_i^{s_i}, \hat{e}_i, N_i\rangle$ as the public key corresponding to the identity ID_i.
- If $i = \gamma$, $\mathcal{C}$ performs the following:
  - Retrieves the tuples $\langle ID_i, g_i\rangle$, $\langle ID_i, g_i^{x_i}, e_i\rangle$ and $\langle ID_i, x_i, d_i, s_i, g_i^{x_i}, -, -, -\rangle$ from the lists L, L_1 and L_S respectively.
  - $\mathcal{C}$ sets $PK_i = \langle PK_i^{(1)}, PK_i^{(2)}, PK_i^{(3)}, PK_i^{(4)}\rangle = \langle g_i^{x_i}, g_i^{s_i}, \hat{e}, N\rangle$. (It is to be noted that $\hat{e}$ and N are taken from the RSA problem instance received by $\mathcal{C}$.)
  - $\mathcal{C}$ now adds the tuple $\langle ID_i, g_i^{x_i}, g_i^{s_i}, e_i, \hat{e}, N\rangle$ into the list L_P and returns $\langle g_i^{x_i}, g_i^{s_i}, \hat{e}, N\rangle$ as the public key corresponding to the identity ID_i.

$\mathcal{O}_{ReplacePublicKey}(ID_i, PK_i^{'})$**:** To replace the public key of ID_i with a new public key $PK_i^{'} = \langle PK_i^{'(1)}, PK_i^{'(2)}, PK_i^{'(3)}, PK_i^{'(4)}\rangle$, chosen by $\mathcal{A}_{II}$, $\mathcal{C}$ checks whether $i \stackrel{?}{=} \gamma$ and does the following:

- If $i = \gamma$ then, $\mathcal{C}$ *aborts* the game.
- If $i \neq \gamma$ then, updates the corresponding tuples in the list L_P as $\langle ID_i, PK_i^{'(1)}, PK_i^{'(2)}, e_i, PK_i^{'(3)} PK_i^{'(4)}\rangle$, only if $(PK_i^{'(2)})^{e_i} = (PK_i^{'(1)})^{e_i} g_i$, where g_i corresponding to ID_i is retrieved from the list L.

Note: This oracle was not considered in both [18] and [11] for Type-II adversary.

$\mathcal{O}_{Decryption}(\sigma, ID_i, PK_i)$: $\mathcal{C}$ performs the following to decrypt the ciphertext $\sigma = \langle c_1, c_2, c_3\rangle$:

- If $i \neq \gamma$, $\mathcal{C}$ performs decryption in the normal way since $\mathcal{C}$ knows the private key corresponding to ID_i.
- If $i = \gamma$, $\mathcal{C}$ performs the following to decrypt the ciphertext $\sigma = \langle c_1, c_2, c_3\rangle$:
  - Retrieves the tuple $\langle ID_i, g_i^{x_i}, e_i\rangle$ from the list L_1.
  - For each $\langle m, r, h\rangle \in L_2$ list performs the following:
    - Checks whether $g_i^h \stackrel{?}{=} c_1$.
    - If `True`, computes $k_1 = (PK_i^{(1)})^{e_i h}$.
    - Checks in list L_3, for the tuples of the form $\langle k_1, k_2, ID_i, h_3\rangle$ (Note that there may be more than one tuple in the list with the same k_1). Picks up the tuple for which the value $k_2^{e_i} = c_2$ and retrieves the corresponding h_3 value. Checks whether $c_3 \oplus h_3 \stackrel{?}{=} (m \| r)$, where m, r is retrieved from the list L_2.
    - If `True`, outputs m as the message.
  - If no tuple satisfies all the above tests, returns *Invalid*.

Challenge: At the end of ***Phase I***, $\mathcal{A}_{II}$ gives $\mathcal{C}$ two messages m_0, m_1 of equal length and an identity ID^* for the challenge. $\mathcal{C}$ *aborts* the game if $ID^* \neq ID_\gamma$, else $\mathcal{C}$ randomly chooses a bit $\delta \in_R \{0,1\}$ and computes a ciphertext σ^* with ID_γ as the receiver by performing the following steps:

- Chooses $h \in \mathbb{Z}_n^{odd}$
- Computes $c_1^* = g_i^h$.
- Sets $c_2^* = b$ (Here b is taken from the RSA problem instance.)
- Choose $c_3^* \in_R \{0,1\}^{l+|\mathbb{Z}_n^*|}$.

Now, $\sigma^* = \langle c_1^*, c_2^*, c_3^* \rangle$ is sent to $\mathcal{A}_{II}$ as the challenge ciphertext.

Phase II: Again $\mathcal{A}_{II}$ can perform polynomially bounded number of queries to the oracles provided by $\mathcal{C}$ with the following conditions:

- $\mathcal{A}_{II}$ should not have queried the decryption oracle with $(\sigma^*, PK_\gamma, ID_\gamma)$ as input. (It is to be noted that PK_γ is the public key corresponding to ID_γ during the challenge phase. $\mathcal{A}_{II}$ is allowed to query the decryption oracle with $(\sigma^*, PK^*, ID_\gamma)$ as input, $\forall PK^* \neq PK_\gamma$).
- $\mathcal{A}_{II}$ should not have queried the secret value corresponding to ID_γ.
- $\mathcal{A}_{II}$ should not have replaced the public key corresponding to the identity ID_γ.

Guess: At the end of ***Phase II***, $\mathcal{A}_{II}$ produces a bit δ' to $\mathcal{C}$ but $\mathcal{C}$ ignores the response and performs the following to output the solution to the RSA problem instance.

- Randomly picks a k_2 value from the list L_3 and outputs it as the solution to the RSA problem instance.

Correctness: Below, we show that the k_2 value obtained through the above step is indeed a, such that $a^{\hat{e}} = b$.

- The public key components $PK_\gamma^{(3)}$ and $PK_\gamma^{(4)}$ corresponding to ID_γ were set to be $\hat{e}$ and N by $\mathcal{C}$ during the request public key query, therefore the private key component $D_\gamma^{(2)} = \hat{d} \; mod \; N$, such that $\hat{d} \equiv \hat{e} \; mod \; \phi(N)$ (It is to be noted that $\mathcal{C}$ does not know $\hat{d}$).
- $\mathcal{C}$ has set the c_2^* component of the challenge ciphertext σ^* as b during the challenge phase.
- In order to decrypt the ciphertext σ^*, $\mathcal{A}_{II}$ should have computed a value $(c_2^*)^{D_\gamma^{(2)}} \; mod \; N$ and queried the H_3 oracle with it as the k_2 component.
- We show that $(c_2^*)^{D_\gamma^{(2)}} \; mod \; N = a$,

  $(c_2^*)^{D_\gamma^{(2)}} \; mod \; N = b^{\hat{d}} \; mod \; N = a^{\hat{e}\hat{d}} \; mod \; N = a \; mod \; N$ (Since $\hat{d} \equiv \hat{e} \; mod \; \phi(N)$)

Thus, $\mathcal{C}$ obtains the solution to the RSA problem with almost the same advantage of $\mathcal{A}_{II}$ in the IND-RSA-CLE$_2$-CCA2-II game. □

Analysis: We now derive the advantage of $\mathcal{C}$ breaking the CCDH problem using the adversary $\mathcal{A}_{II}$. The simulations of H, H_1, H_2 and H_3 clearly shows that the hash oracles are perfectly random. Let ϵ be the advantage of $\mathcal{A}_{II}$ in winning the IND-RSA-CLE$_2$-CCA2-I game.

The events in which $\mathcal{C}$ aborts the game and the respective probabilities are given below:

1. $\mathcal{E}_1$ - The event in which $\mathcal{C}$ *aborts* when $\mathcal{A}_{II}$ queries the secret value corresponding to ID_γ.
2. $\mathcal{E}_2$ - The event in which $\mathcal{C}$ *aborts* when $\mathcal{A}_{II}$ replaces the public key corresponding to ID_γ.
3. $\mathcal{E}_3$ - The event in which ID_γ is not chosen as the target identity by $\mathcal{A}_{II}$ for the challenge.

Suppose $\mathcal{A}_I$ has made q_H number of $\mathcal{O}_H$ queries, q_{sv} number of $\mathcal{O}_{ExtractSecretValue}$ queries, q_{rpk} number of identities for which $\mathcal{O}_{ReplacePublicKey}$ queries and q_{srk} be the total number of identities for which the secret value is extracted and the public key is replaced, then:

$\Pr[\mathcal{E}_1] = \frac{q_{sv}}{q_H}$, $\Pr[\mathcal{E}_2] = \frac{q_{rpk}}{q_H}$ and $\Pr[\mathcal{E}_3] = 1 - \frac{1}{q_H - q_{sv}}$.

Therefore,

$$\Pr[\neg abort] = [\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2 \wedge \neg\mathcal{E}_3] = \left[1 - \frac{q_{sv}}{q_H}\right] . \left[1 - \frac{q_{rpk}}{q_H}\right] . \left[1 - \left[1 - \frac{1}{q_H - q_{srk}}\right]\right] = \left[1 - \frac{q_{sv}}{q_H}\right] . \left[1 - \frac{q_{rpk}}{q_H}\right] . \left[\frac{1}{q_H - q_{srk}}\right].$$

Therefore, the advantage of $\mathcal{C}$ solving the CCDH problem is $\epsilon' \geq \left(\epsilon . \left[1 - \frac{q_{sv}}{q_H}\right] . \left[1 - \frac{q_{rpk}}{q_H}\right] . \left[\frac{1}{q_H - q_{srk}}\right]\right)$.

5 Comparison Study

We compare our schemes with the two existing secure schemes [11] and [18]. We compare the level of security offered by each schemes and the assumptions used to prove the security against the two adversaries. The Type-I security of the scheme in [11] is based on RSA assumption and thus operates on composite groups and is CPA secure against both Type-I and Type-II adversaries. The Type-II security is based on the composite computational Diffie Hellman Assumption (CCDH). Both Type-I and Type-II securities of the scheme in [18] are based on the CDH assumption in multiplicative groups with prime order. Our schemes are based on RSA assumption and operates on composite groups. The major operations in all the schemes are multiplication and exponentiation, still, we do not consider them for the comparison due to the fact that the security parameters are different for RSA based schemes and schemes based on multiplicative groups with prime order.

Scheme	Security	Assumption	
		Type-I	Type II
Lai et al. [11]	CPA	RSA	CCDH
Sun et al. [18]	CCA2	CDH	CDH
RSA-CLE_1	CCA2	RSA	CCDH
RSA-CLE_2	CCA2	RSA	RSA

Table-1: Comparison of level of security and assumptions

6 Acknowledgment:

We would like to extend our sincere thanks to the anonymous referees of the PROVSEC-2010 program committee who have given us insightful remarks which helped us in improving the security proofs of the scheme.

7 Conclusion

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 [11] and [18] 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 [11] is not 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 our schemes in the random oracle model. We leave it an interesting open problem to design a CLE scheme in the original model [1] with the security of the scheme fully based on RSA assumption.

References

1. Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.
2. Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Certificateless public key encryption without pairing. In *Information Security - ISC 2005*, volume 3650 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2005.
3. Xavier Boyen. Multipurpose identity-based signcryption (a swiss army knife for identity-based cryptography). In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2003.

4. Rafael Castro and Ricardo Dahab. Two notes on the security of certificateless signatures. In *Provable Security - ProvSec 2007*, volume 4784 of *Lecture Notes in Computer Science*, pages 85–102. Springer, 2007.
5. Zhaohui Cheng and Richard Comley. Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012, 2005. http://eprint.iacr.org/.
6. Alexander W. Dent. A survey of certificateless encryption schemes and security models. *International Journal of Information Security*, 7(5):349–377, 2008.
7. David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(DOI: 10.1007/s00145-009-9048-z):224280, 2010.
8. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In *Public Key Cryptography, PKC '99*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 1999.
9. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
10. Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. On the security of certificateless signature schemes from asiacrypt 2003. In *Cryptology and Network Security - CANS 2005*, volume 3810 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 2005.
11. Junzuo Lai, Robert H Deng, Shengli Liu, and Weidong Kou. Rsa-based certificateless public key encryption. In *Information Security Practice and Experience - ISPEC 2009*, volume 5451 of *Lecture Notes in Computer Science*, pages 24–34. Springer, 2009.
12. Joseph K Liu, Man Ho Au, and Willy Susilo. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model: extended abstract. In *ASIACCS 2007, Proceedings of the 2nd ACM symposium on Information, Computer and Communications Security*, pages 273–283. ACM, 2007.
13. Kevin S. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology*, Volume 1(Number 2):95–105, 1988.
14. Jong Hwan Park, Kyu Young Choi, Jung Yeon Hwang, and Dong Hoon Lee. Certificateless public key encryption in the selective-id security model (without random oracles). In *Pairing-Based Cryptography - Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 60–82. Springer, 2007.
15. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology, CRYPTO - 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.
16. Yijuan Shi and Jianhua Li. Provable efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/287, 2005. http://eprint.iacr.org/.
17. Z. Shmuely. Composite diffie-hellman public-key generating systems are hard to break. Technical Report No. 356, Computer Science Department, Technion-Israel Institute of Technology, February, 1985.
18. Yinxia Sun, Futai Zhang, and Joonsang Baek. Strongly secure certificateless public key encryption without pairing. In *Cryptology and Network Security - CANS 2007*, volume 4856 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2007.