Provable secure certificateless generalized signcryption scheme

Prashant Kushwah  
*Department of Mathematics and Statistics, Banasthali University, Rajasthan, India*  
*Email: pra.ibs@gmail.com*

Sunder Lal  
*Vice-Chancellor, V. B. S. Purvanchal University, Jaunpur, India*  
*Email: sunder_lal2@rediffmail.com*

Abstract

Generalized signcryption is a new cryptographic primitive, which works as a signcryption scheme, a signature scheme and an encryption scheme as per need. It is more suitable for some storage constrained environments, e.g. smart cards, WSN (Wireless sensor networks) etc. Recently, Ji et al. proposed a security model for certificateless generalized signcryption scheme and also proposed a scheme, which they claim to be secure under the proposed security model. In this paper we show that there security model is not complete and propose a modified security model. We also prove that Ji et al. scheme is not existentially unforgeable against Type-I adversary and propose a new efficient certificateless generalized signcryption scheme and prove the security of the proposed scheme under the new security model. Thus our proposed scheme is the first provable secure certificateless generalized signcryption scheme.

Keywords: signcryption, generalized signcryption, certificateless cryptography.

1. Introduction

Confidentiality and authenticity are two logically independent primitives of cryptography. An encryption scheme is used to achieve confidentiality and authenticity is achieved through a signature scheme. There are scenarios, where both the primitives are needed (for example secure e-mailing). We can achieve both primitives by “signature-then-encryption” approach but this approach has the high computational cost and communication overhead. However, in 1997, Zheng [30] proposed a novel cryptographic primitive “Signcryption” which achieve both confidentiality and authenticity in a single logical step with the cost significantly lower than the “signature-then-encryption” approach.

In 1984, Shamir [24] introduced the concept of identity based cryptography. In the identity based cryptosystems public keys of users are their identities (e.g. e-mail address, PAN number etc.) and secret keys of users are created by a trusted third party called private key generator (PKG). The first identity based signature scheme was given by Shamir [24] in 1984, but the first identity based encryption scheme was given by Boneh and Franklin [8] in 2001. In 2002, Malone-Lee [20] proposed the first identity based signcryption scheme along with the security model for signcryption in the identity based setting. Since then, many identity based signcryption schemes have been proposed in literature [5, 9, 10, 12, 19].

In 2003, Al-Riyami and Paterson [2] proposed certificateless public key cryptosystems, which avoids the key escrow problem of identity based cryptography and the need of certificates in public key cryptography. They proposed a certificateless encryption scheme and briefly outlined a signature scheme and extensions of other public-key primitives to the certificateless scenario [1]. In 2008, Barbosa and Farshim [4] introduced the notion of certificateless signcryption (CLSC). To improve efficiency, Aranha et al. [3], Wu et al. [26] and Xie et al. [27] separately gave three efficient CLSC schemes. Unfortunately in [22, 23], Selvi et al. showed that the schemes in [3, 26, 27] are not secure. Yu et al. [29] proposed an efficient CLSC scheme from the Weil pairing but they limit the capability of an adversary to avoid the attack given by Selvi et al. [23].

However, in some scenarios, we need both confidentiality and authenticity and sometimes, we need confidentiality or authentication separately. To achieve this, we can use three different cryptographic primitives: encryption, signature and signcryption. This
increases the implementation complexity. Also in the low bandwidth environment e.g. smartcards, ubiquitous computing and WSN (Wireless Sensor Networks), we cannot afford to use three different schemes to achieve confidentiality and authentication separately or simultaneously. Motivated by these scenarios, Han et al. [15] proposed the concept of generalized signcryption, which can work as an encryption scheme or a signature scheme or a signcryption scheme. Wang et al. [25] gave the security model for a generalized signcryption scheme and modified the scheme proposed in [15]. Han et al. proposed another generalized signcryption scheme based on bilinear pairing with the shortened ciphertext in [14]. The first identity based generalized signcryption (IBGSC) along with a security model was proposed by Lal and Kushwah [18] in 2008. However, Yu et al. [28] showed that the security model for IBGSC proposed in [18] is not complete. They modified the security model and proposed a concrete scheme which is secure in modified model. Recently, Kushwah and Lal [17] simplified the security notions for IBGSC and proposed an efficient IBGSC scheme. Meanwhile, Ji et al. [16] modelled security notions of generalized signcryption in the certificateless setting and proposed a concrete scheme which they claim to be secure in the proposed model. However, in this paper we show that Ji et al. [16] security model is not complete and moreover their scheme is not existentially unforgeable against Type-I adversaries. Further, we propose a new security model for CLGSC based on [17] and proposed an efficient CLGSC scheme. We also prove the security of the proposed scheme in the new security model.

This paper is organized as follows: In section 2, we give the formal definition of CLGSC and its security models. In section 3, we show that Ji et al.’s [16] scheme is not existentially unforgeable against Type-I adversaries [4]. Section 4 contains the preliminaries for the proposed scheme. Section 5 contains the new proposed CLGSC scheme followed by the formal security proofs of the proposed scheme in section 6. Finally, we conclude our paper in section 7.

2. Certificateless Generalized Signcryption (CLGSC)

A CLGSC scheme consists of the following algorithms:

1. Setup: This algorithm takes input a security parameter \( k \) and outputs the system parameter \( \text{params} \) and a master secret key.

2. Partial Private Key Generation: Given input \( \text{params} \), the master secret key and a user’s identity \( ID_U \), it outputs a partial private key \( D_U \) corresponding to \( ID_U \).

3. Set User Key: Given input \( ID_U \), partial private key \( D_U \) corresponding to \( ID_U \), it outputs a public key \( PK_U \) of the identity \( ID_U \) and a secret value \( x_U \). The secret key \( (SK_U) \) of the user is \( (x_U, D_U) \).

4. CLGSC (generalized signcryption): To send a message \( m \) from \( A \) to \( B \), this algorithm takes inputs \( (SK_A, m, ID_A, PK_A, ID_B, PK_B) \) and outputs \( \sigma = \text{CLGSC}(SK_A, m, ID_A, PK_A, ID_B, PK_B) \).

5. CLGUSC (generalized unsigncryption): This algorithm takes input \( (\sigma, ID_B, SK_B, PK_B, ID_A, PK_A) \) and outputs \( m \) and \( T \) if \( \sigma \) is valid otherwise return \( \perp \) if \( \sigma \) is not valid.

Note that there is no specific sender (receiver) when we only encrypt (only sign) a message \( m \) using CLGSC. We denote the absence of a sender (or receiver) by \( ID_\varnothing \). Thus we use \( ID_B = ID_\varnothing \) or \( ID_A = ID_\varnothing \) to only sign or encrypt a message \( m \) respectively. Therefore when \( ID_B = ID_\varnothing \), CLGSC becomes a certificateless signature scheme and output of CLGSC algorithm is a signature of sender \( ID_A \) on the message \( m \) and when \( ID_A = ID_\varnothing \), CLGSC becomes a certificateless encryption scheme and output of CLGSC algorithm is merely an encryption of message \( m \) for receiver \( ID_B \). If \( ID_A \neq ID_\varnothing \) and \( ID_B \neq ID_\varnothing \), then CLGSC works as a certificateless signcryption scheme and output of CLGSC is the signcryption of message \( m \) with the signature of sender \( ID_A \) to the receiver \( ID_B \). Thus CLGSC works in three modes namely signcryption mode, encryption-only mode and signature-only mode.

2.1 Security model for CLGSC

There are two different types of adversaries in the certificateless cryptosystems. A Type-I adversary \( A_I \) who is not allowed access to the master secret key but \( A_I \) may request public keys and replace them with values of her choice. The Type-II adversary \( A_{II} \) is allowed access to the master secret key but cannot replace any public keys.

2.1.1 Message Confidentiality

The notion of security with respect to confidentiality is indistinguishability of encryptions under adaptive chosen ciphertext attack (IND-CCA2). For CLGSC this notion is captured by the following game played between challenger \( C \) and adversary \( A \).
GAME 1 (IND-CCA2):

Initialization: On input a security parameter $k$, $C$ runs the setup algorithm and gives public parameters params to the adversary $A$. $C$ keeps the master key secret if $A$ is Type-I adversary else it also provides the master secret key to $A$.

Find Stage: The adversary $A$ makes the following queries adaptively.

- **Hash Queries**: $A$ can request the hash values of any input and challenger responds with appropriate hash values.
- **Partial Private Key Extraction**: $A$ submits an identity $ID_U$, $C$ computes the partial private key $D_U$ corresponding to $ID_U$ and returns it to $A$. Note that the adversary $A = A_U$ does not need this oracle because it has the master secret key and can compute the partial private key for any user.
- **Public Key Extraction**: $A$ submits an identity $ID_U$ for which she wants the public key, $C$ computes the corresponding public key $PK_U$ and sends it to $A$.
- **Private Key Extraction**: $A$ submits an identity $ID_U$, $C$ computes the corresponding private key $SK_U$ and sends it to $A$. Note that if $A$ is Type-I adversary then $A$ is not allowed to extract the full private key of any identity for which corresponding public key has been replaced. Because in this case challenger is not able to provide the full private key of that user.
- **Public Key Replacement**: If $A$ is Type-I adversary then $A$ has access to this oracle. For any identity $ID_U$, $A$ computes the new public key $PK_U$ by choosing a new secret value $x'_U$ of her choice and replacing $PK_U$. Note that if $A$ is Type-II adversary then $A$ cannot replace public key of any user.
- **CLGSC Queries**: $A$ submits two identities $ID_A$, $ID_B$ and a message $m$. Challenger $C$ runs the CLGSC algorithm with message $m$ and identities $ID_A$ and $ID_B$ and returns the output $\sigma$ to the adversary $A$. Note that if $A$ sets $ID_A = ID_\varphi$ then $C$ only encrypts the message $m$ for $ID_B$ and if $A$ sets $ID_B = ID_\varphi$, then $C$ only signs the message $m$ under $ID_A$.
- **CLGUSC Queries**: $A$ submits two identities $ID_A$, $ID_B$ along with $\sigma$ to the challenger $C$. $C$ runs the CLGUSC algorithm with input $\sigma$, $ID_A$ and $ID_B$ and returns the output $m$ and $T$ of CLGUSC. Note that if $A$ sets $ID_A = ID_\varphi$ then $\sigma$ is only the encryption for $ID_B$ and if $A$ sets $ID_B = ID_\varphi$ then $\sigma$ is only the signature of $ID_A$ on message $m$.

Note that it is possible that the public key $PK_A$ (or $PK_B$) corresponding to $ID_A$ (or $ID_B$) has been replaced earlier by $A$ (if $A$ is Type-I adversary) in CLGSC (or CLGUSC) queries. If so, $A$ has to submit the corresponding secret value to $C$ for the correctness of these oracles. Also we disallow queries to these oracles when $ID_A = ID_B$ for irreflexivity assumption.

Challenge: At the end of the find stage, $A$ submits two distinct messages $m_0$ and $m_1$ of equal length, a sender’s identity $ID_A^*$ and a receiver’s identity $ID_B^*$ on which he wishes to be challenged. The adversary $A$ must have made no private key extraction query (and partial private key extraction query if $A$ is Type-I adversary) on $ID_B^*$, also $ID_B^* \neq ID_\varphi$ for the confidentiality game. $C$ picks a bit $b \in \{0,1\}$ randomly and runs the CLGSC algorithm with message $m_b$ under $ID_A^*$ and $ID_B^*$ and returns the output $\sigma^*$ to the adversary $A$.

Guess stage: $A$ asks queries adaptively again as in the find stage. It is not allowed to extract the private key (and partial private key if $A$ is Type-I adversary) corresponding to $ID_B^*$ and it is not allow to make an CLGUSC query on $\sigma^*$ with sender $ID_A^*$ and receiver $ID_B^*$ unless the public key $PK_A^*$ of the sender or that of the receiver $PK_B^*$ has been replaced after the challenge if $A$ is Type-I adversary.

Eventually, $A$ outputs a bit $b'$ and wins the game if $b = b'$. $A$’s advantage is defined as

$$Adv_{\lambda}^{IND-CCA2} = 2Pr[b = b'] - 1.$$ 

Note that for the confidentiality in the encryption only mode, adversary $A$ will play the above game with $ID_A^* = ID_\varphi$. Also in the above game adversary $A$ cannot submit $\sigma^*$ to CLGUSC oracle.
strictly with sender $ID^*_A$ and receiver $ID^*_B$. However, if $\sigma^*$ is the signcrypted text then $A$ is allowed to transform $\sigma^*$ into a valid encrypted text and can query the CLGUSC oracle with the sender $ID_\phi$. Also if $\sigma^*$ is the encrypted text then $A$ is allowed to transform $\sigma^*$ into a valid signcrypted text and can query the CLGUSC oracle with sender $ID^*_A \neq ID_\phi$. These differences are not considered in [16].

**Definition 1:** A CLGSC scheme is said to be IND-CCA2 secure if no polynomially bounded adversary $A$ (Type-I or Type-II) has non-negligible advantage of winning the above game.

2.1.2 Signature unforgeability

The notion of security with respect to authenticity is existential unforgeability against chosen message attacks (EUF-CMA). For CLGSC this notion is captured by the following game played between challenger $C$ and adversary $A$.

**GAME 2 (EUF-CMA):**

**Initialization:** Same as in GAME 1.

**Queries:** The adversary $A$ asks a polynomially bounded number of queries adaptively as in the GAME 1.

**Forgery:** Finally, $A$ produces a triplet $(ID^*_A, ID^*_B, \sigma^*)$ that was not obtained from CLGSC query during the game and for which private key (and partial private key if $A$ is Type-I adversary) of $ID^*_A$ was not exposed, also $ID^*_A \neq ID_\phi$ for the signature unforgeability game.

The adversary $A$ wins the game if the result of $CLGSC(\sigma^*, ID^*_B, T^*_B, PK^*_B, ID^*_A, PK^*_A)$ is not the $\bot$ symbol.

The adversary $A$’s advantage is its probability of success.

Note that for the unforgeability in the signature only mode, adversary $A$ will play the above game with $ID^*_B = ID_\phi$. Also in the above game, $\sigma^*$ is not a valid forgery if it is the output of CLGSC query strictly with identities $ID^*_A$ and $ID^*_B$. But it can be the transformation of valid signcrypted text obtained from CLGSC query to the signature for the unforgeability in the signature only mode. Also it can be the transformation of valid signature obtained from CLGSC query to the signcrypted text for some receiver $ID^*_B$ for the unforgeability in the signcryption mode. These differences are not considered in [16].

**Definition 2:** A CLGSC scheme is said to be EUF-CMA secure if no polynomially bounded adversary $A$ (Type-I or Type-II) has non-negligible advantage of winning the above game.

3. Review of Ji et al. [16] Certificateless Generalized Signcryption

**Setup:** Given a security parameter $k$, the PKG chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order $p$, two generators $P, Q$ of $\mathbb{G}_1$ randomly, $P \neq Q$, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Computes $g = e(P, Q)$, defines five hash functions $H_i : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_i : \mathbb{G}_2 \times \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_i : \{0,1\}^n \times \mathbb{G}_2 \times \mathbb{Z}_p^* \times \{0,1\}^*$ $\rightarrow \mathbb{G}_2 \times \mathbb{G}_1 \times \{0,1\} \rightarrow \mathbb{G}_2^*$, $H_i : \{0,1\}^n \times \mathbb{G}_2 \times \mathbb{G}_2 \times \{0,1\} \rightarrow \{0,1\}^{k_1+k_2}$, where $k_1$ and $k_2$ denotes the number of bits to represent elements of $\mathbb{G}_1$ and $\mathbb{Z}_p^*$ respectively. PKG chooses a random $s \in \mathbb{Z}_p^*$ as the master secret key and sets $P_{pub} = sP$. PKG publishes $\langle \mathbb{G}_1, \mathbb{G}_1, p, P, Q, P_{pub}, e, g, H_1, H_2, H_3, H_4, H_5 \rangle$ as the system parameters.

**Extract Partial Private Key:** Given $ID_U$, the partial private key of the user with identity $ID_U$ is computed by PKG as $D_U = (qt_U + s)^{-1}Q$, where $qt_U = H_1(ID_U)$.

**Set User Key:** Given $D_U$, the user $U$ chooses random $x_U \in \mathbb{Z}_p^*$ and sets his private key as $SK_U = \{x_U, D_U\}$ and public key as $PK_U = \langle PK_{U1}, PK_{U2} \rangle = g^{x_U}x_U, T_U \rangle$, where $T_U = (qt_U + s)P$.

**CLGSC:** This algorithm has three scenarios: signcryption mode, signature only mode and encryption only mode.

**Signcryption:** given message $m$, sender’s identity $A$, receiver’s identity $ID_B$, $A$ operates as follows:

1. Chooses random $r, r' \in \mathbb{Z}_p^*$, computes $\alpha = g^{r'}$.
2. Computes $h = H_2(\alpha, mID_A, ID_B)$ and $h_3 = H_3(m, \alpha, h, ID_A, PK_{A1}, PK_{A2}, ID_B)$.
3. Computes $Z = (x_A + h_3)D_A$.
4. Computes \( c = H_4(h, ID_A, ID_B) \oplus m \parallel \alpha \); 
5. Computes \( h_5 = H_5(g^{r'}, (PK_{B1})^{r'}, PK_{B1}, ID_B) \); 
6. Computes \( d_1 = r'(q_{B} + s)P, \quad d_2 = h_5 \oplus h \parallel Z \); 
7. Returns ciphertext \( \sigma = (c, d_1, d_2, ID_B) \).

**Signature:** given message \( m \), the sender \( A \) operates as follows:

1. Chooses random \( r, r' \in Z_p^* \), computes \( \alpha = g^{r'} \); 
2. Computes \( h = H_2(\alpha, m, ID_A, 0) \) and \( h_3 = H_3(m, \alpha, h, ID_A, PK_{A1}, PK_{A2}, 0) \); 
3. Computes \( Z = r(x_A + h_2)D_A \); 
4. Computes \( c = m \parallel \alpha \); 
5. Computes \( h_5 = 0 \); 
6. Computes \( d_1 = 0 \) and \( d_2 = h_5 \oplus h \parallel Z = h \parallel Z \); 
7. Returns ciphertext \( \sigma = (c, d_1, d_2, 0) \).

**Encryption:** Given message \( m \) and receiver \( B \), someone operates as follows:

1. Chooses random \( r, r' \in Z_p^* \), computes \( \alpha = g^{r'} \); 
2. Computes \( h = H_2(\alpha, m, 0, ID_B) \) and \( h_2 = H_3(m, \alpha, h, 0, 0, 0, ID_B) \); 
3. Computes \( c = H_4(h, ID_B) \oplus m \parallel \alpha \); 
4. Computes \( h_5 = H_5(g^{r'}, (PK_{B1})^{r'}, PK_{B1}, ID_B) \); 
5. Computes \( d_1 = r'(q_{B} + s)P, \quad d_2 = h_5 \oplus h \parallel 0 \); 
6. Returns cipher text \( \sigma = (c, d_1, d_2, ID_B) \).

**CLGUSC:** Given \( \sigma \), the receiver \( B \) operates as follows:

1. Computes \( w' = e(d_1, DB) \) and \( (w')^yB \) (if the receiver \( B \) is not specified then \( DB = 0 \), and \( w' = 1 \)); 
2. Sets \( h'_5 = H_5(w', (w')^yB, PK_{B1}, ID_B) \); (if \( DB = 0 \), then \( h'_5 = 0 \)); 
3. Computes \( h' \parallel Z = d_2 \oplus h'_5 \); 
4. If \( ID_B \neq 0 \), computes \( m' \parallel \alpha' = c \oplus H_4(h', ID_A, ID_B) \); (if \( Z' = 0 \), then \( ID_B = 0 \)); 
5. If \( ID_B \neq 0 \), computes \( h_5 = H_3(m', \alpha', h', ID_A, PK_{A1}, PK_{A2}, ID_B) \); (if \( Z' = 0 \), then \( ID_B = 0 \) and \( PK_{A1} = PK_{A2} = 0 \)); 
6. If \( Z' = 0 \), then \( B \) accepts \( m' \) if and only if \( h' = H_2(\alpha', m', ID_A, ID_B) \) and \( e(Z', PK_{A2} + h'_5(q_A + s))P = \alpha' \) holds. Otherwise accept \( m' \) if and only if \( h' = H_2(\alpha', m', 0, ID_B) \).

### 3.1 Attack on unforgeability of Ji et al. [16] scheme by a Type-I adversary

Now, we model an attack on the unforgeability of Ji et al. scheme in signcryption mode by an insider Type-I adversary. Note that Type-I adversary \( A \) has the power to replace public keys of any user with his choice. To forge a signcrypted text, \( A \) does the following:

1. \( A \) replaces the public key \( PK_A = < PK_{A1}, PK_{A2} > = < g^{x_A}, x_A^BA > \) of \( A \) with \( PK'_A = < PK_{A1}, PK_{A2} > = < g^{x'_A}, x'_A^BA > \) by choosing \( x'_A \) randomly from \( Z_p^* \).
2. \( A \) submits a message \( m \), a sender’s identity \( A \), the receiver’s identity \( B \) to signcryption oracle. Note that \( A \) provides the secret value \( x'_A \) to the challenger \( C \) when public key of sender \( A \) is replaced.
3. \( C \) returns the ciphertext \( \sigma = (c, d_1, d_2, ID_B) \)
4. \( A \) extract \( Z \) and \( \alpha \) from \( \sigma \).

Now \( A \) can use \( Z = (r/(x'_A + h_3))D_A \) and \( \alpha = g^{r'} \) to signcrypt any message \( m' \) intended to any receiver \( B' \). For this \( A \)

1. Chooses randomly \( r' \in Z_p^* \).
2. Computes \( h' = H_2(\alpha, m', ID_A, ID_B) \) and \( h'_5 = H_5(g^{r'}, (PK_{B1})^{r'}, PK_{B1}, ID_B) \);
3. Computes \( c' = H_4(h', ID_A, ID_B) \oplus m' \parallel \alpha \);
4. Computes \( h'_5 = H_5(g^{r'}, (PK_{B1})^{r'}, PK_{B1}, ID_B) \);
5. Sets \( Z' = ((x'_A + h_3)/(x'_A + h'_5))Z \); 
6. Computes \( d'_1 = r'(q_{B} + s)P, \quad d'_2 = h'_5 \oplus h' \parallel Z' \); 
7. Returns the cipher text \( \sigma' = (c', d'_1, d'_2, ID_B) \).

Now we show that \( \sigma' \) is accepted as valid by \( B' \). For this \( B' \) does the following:

1. Computes 
   \[ w = e(d_1, DB) = e(r'(q_{B} + s)P, (q_{B} + s)^{-1}Q) = e(P, Q)^{r'} = g^{r'} \]
   and
   \[ (w)^{yB} = (g^{r'})^{yB} = (g^{yB})^{r'} = (PK_{B1})^{r'} \]; 
2. Sets \( h'_5 = H_5(w, (w)^{yB}, PK_{B1}, ID_B) \); 
3. Recovers \( h' \parallel Z = d'_2 \oplus h'_5 \); 
4. Recovers \( m' \parallel \alpha = c' \oplus H_4(h', ID_A, ID_B) \); 
5. Computes \( h'_5 = H_5(m', \alpha, h', ID_A, PK_{A1}, PK_{A2}, ID_B) \);
6. \(B'\) will accept the message \(m'\) because
\[ h' = H_2(\alpha, m', ID_A, ID_B) \quad \text{and} \quad e(Z', PK_{A2} + h_3'(Q_A + s)P) = \alpha \text{ holds.} \]

Verification:
\[
e(Z', PK_{A2} + h_3'(Q_A + s)P)
\]
\[= e\left(\frac{x_A^r + h_2}{x_A^r + h_3', h_3}, Q, (x_A + h_2'(Q_A + s)P)\right)
\]
\[= e(P, Q)^r = g^r = \alpha.\]

Thus we have shown that a Type-I adversary can forge the signcryption on any message for any receiver. The reason of such type of attack is that Ji et al. use different randomness for encryption and signature. A similar type of attack can be modelled in the signature mode.

4. Preliminaries

Let \(G_1\) be an additive group and \(G_2\) be a multiplicative group both of the same prime order \(p\). A function \(e: G_1 \times G_1 \rightarrow G_2\) is called a bilinear pairing if it satisfies the following properties:

1. \(\forall P, Q \in G_1, \forall a, b \in \mathbb{Z}_p^*\), \(e(aP, bP) = e(P, Q)^{ab}\)
2. For any \(O \neq P \in G_1\), there is \(Q \in G_1\), such that \(e(P, Q) \neq 1\).
3. There exists an efficient algorithm to compute \(e(P, Q), \forall P, Q \in G_1\).

Given \((P, aP, bP)\) for unknown \(a, b \in \mathbb{Z}_p^*\) to compute \(abP\) is known as Computational Diffie Hellman Problem (CDHP).

Given a \((q + 1)\) tuple \((P, aP, a^2P, \ldots, a^qP)\) to compute \(e(P, P)^{1/qa} \in G_2\) is known as \(q\)-Bilinear Diffie Hellman inversion problem \(q\)-BDH[7].

Given \(P, sP \in G_1, \lambda_1, \lambda_2, \ldots, \lambda_k \in \mathbb{Z}_p^*\) and \(\frac{1}{s + \lambda_1}, \frac{1}{s + \lambda_2}, \ldots, \frac{1}{s + \lambda_k} \in G_2\) where \(s \in \mathbb{Z}_p^*\) is unknown and \(k\) is integer, to compute \(\frac{1}{s + \lambda}P\) for some \(\lambda \notin \{\lambda_1, \lambda_2, \ldots, \lambda_k\}\) is known as \(k\)-CAA problem [21].

Given \(P, aP \in G_1\) where \(a\) is unknown to compute \(\frac{1}{a + \lambda}P\) for some \(\lambda \in \mathbb{Z}_p^*\) is known as Inversion

Computational Diffie Hellman Problem (Inversion CDHP) [21].

5. Proposed CLGSC Scheme

In this section we propose a new CLGSC scheme based on the certificateless encryption scheme proposed in [11].

Setup: Given a security parameter \(k\), the PKG chooses two groups \(G_1\) and \(G_2\) of prime order \(p\), a random generator \(P\) of \(G_1\), and a bilinear map \(e: G_1 \times G_1 \rightarrow G_2\). Let \(g = e(P, P)\), defines hash functions \(H_1: \{0,1\}^k \rightarrow \mathbb{Z}_p^*, \quad H_2: \{0,1\}^{n+k_2+2k_0} \rightarrow \mathbb{Z}_p^*, \quad H_3: \{0,1\}^{n+k_2+k_0} \rightarrow \mathbb{Z}_p^*, \quad H_4: \{0,1\}^{k_2+k_1 + \{0,1\}^{n+k_2+k_0}}, \quad k_1, k_2\) and \(k_0\) denote the number of bits to represent elements of \(G_1, G_2\) and identity respectively and \(n\) is the message bit length. PKG chooses random \(s \in \mathbb{Z}_p^*\) as the master secret key and sets \(P_{pub} = sP\). PKG publishes the system parameters as \(\{G_1, G_2, p, n, P, P_{pub}, e, g, H_1, H_2, H_3, H_4\}\).

Let \(f\) be a function such that \(f(ID) = 0\) if \(ID = ID_\varphi\) otherwise \(f(ID) = 1\).

Extract Partial Private Key: Given a user \(U\) with identity \(ID_U\) the partial private key is computed by PKG as \(D_U = (q_U + s)^{-1}P\) where \(q_U = H_1(ID_U)\). For \(ID_\varphi\), we set \(D_\varphi = O\).

Set User Key: Given \(D_U\), the user \(U\) chooses random \(x_U \in \mathbb{Z}_p^*\) and set his private key \(SK_U = \langle x_U, D_U \rangle\) and public key \(PK_U = x_U T_U\), where \(T_U = (q_U + s)P\). For \(ID_\varphi\), we set \(T_\varphi = O\) and \(x_\varphi = 0\), Thus \(PK_\varphi = O\).

CLGSC: The sender \(A\) for the receiver \(B\)

1. Chooses \(r \in R \mathbb{Z}_p^*\)
2. Computes
   
   - \(\alpha = g^r\)
   - \(r' = H_2(m, \alpha, ID_A, ID_B)\)
   - \(X = r T_B\)
   - \(h_3 = H_3(m, \alpha, X, PK_A, PK_B, ID_A, ID_B)\)
   - \(Z = \frac{r + r'}{(x_A + h_3) D_A}\)
vi. $y = m \parallel \alpha \parallel Z \parallel ID_A \oplus \{ H_4(g \parallel r'PK_B) \}$

$f(ID_B) \}$

3. Returns $\sigma = (y, X)$.

**CLGUSC:** On receiving $\sigma$ from $A$, the user $B$

1. Recovers $m \parallel \alpha \parallel Z \parallel ID_A = y$ if $Z = O$, otherwise

2. Computes $\omega = e(X, DB)$ and recovers

$m \parallel \alpha \parallel Z \parallel ID_A = y \oplus \{ H_4(\omega \parallel x_B X) f(ID_B) \}$

3. Computes $r' = H_2(m, \alpha, ID_A, ID_B)$

4. If $Z = O$, accept the message iff $X = r'T_B$, otherwise

5. Computes $h_3 = H_3(m, \alpha, X, PK_A, PK_B, ID_A, ID_B)$

and accept the message iff

$e(Z, PK_A + h_3T_A) = \alpha g^{r'}$.

**Consistency:**

$\omega = e(X, DB) = e(r'T_B, DB)$

$= e(r'(q_B + s)P, (q_B + s)^{-1}P) = e(P, P)^{r'}$

$x_BX = x_Br'T_B = r'PK_B$

$e(Z, PK_A + h_3T_A)$

$= e\left( \frac{r + r'}{x_A + h_3} D_A \cdot x_A T_A + h_3T_A \right)$

$= e\left( \frac{r + r'}{x_A + h_3} (q_A + s)^{-1}P, (x_A + h_3)(q_A + s)P \right)$

$= e(P, P)^{r + r'} = g^{r + r'} = \alpha g^{r'}$

**Remarks:**

1. When we only sign a message, then the specific receiver $B$ does not exist. Therefore we use $ID_B = ID_\varnothing$ in the CLGSC algorithm. Thus the function $f(ID_\varnothing)$ becomes 0 which helps to give us the signature $y = m \parallel \alpha \parallel Z \parallel ID_A$, also the component $X$ of the output of CLGSC algorithm becomes $O$. This will reduce the extra computations in CLGUSC.

2. When we only encrypt a message then specific sender $A$ does not exist. Therefore we use $ID_A = ID_\varnothing$ in CLGSC algorithm. Thus in the computation of $Z$ we have $D_\varnothing = O$ which will again reduce the extra computations in CLGUSC by checking $X = r'T_B$, which will also provide chosen ciphertext security while we only encrypt a message.

3. The form of ciphertext is $(y, X)$ either we encrypt a message or signcrypt a message. The computations of $r'$ and $h_3$ involve both sender’s and receiver’s identity. This prevents an adversary to embed an encryption to valid signcryption or vice versa. Similarly an adversary cannot embed a signature of a message to valid signcryption or vice versa.

4. An important feature of the proposed IBGSC scheme is that there is no need to bind the information of sender and receiver to recognize that the ciphertext is the signcrypted text or the encrypted text or only the signature. Because UIBGSC algorithm itself distinguish these the modes.

5. Also note that we use the same randomness for encryption and signature i.e. we use $r'$ in the computation of $Z$, which avoids the unforgeability attack against proposed certificateless generalized signcryption, which we presented in section 6 on Ji et al. [16] scheme.

**6. Security Results**

**Theorem 1:** The proposed CLGSC scheme is IND-CCA2 secure under the $q$-BDHIP and CDHIP.

The theorem follows from the lemma 1 and 2.

**Lemma 1:** Assume that an IND-CCA2 Type-I adversary $A_j$ has an advantage $\varepsilon$ against the proposed CLGSC scheme, asking $q_{h_3}$ queries to random oracle $H_i(i = 1,2,3,4)$ and $q_{d_e}, q_{d_u}$ CLGSC queries, CLGUSC queries respectively. Then there is an algorithm $B$ to solve the $q$-BDHIP for $q = q_{h_3}$ with probability

$$\varepsilon' > \frac{\varepsilon}{q_{h_3}(q_{h_3} + q_{d_e})} \left(1 - \frac{q_{d_e}}{2^k}\right) \left(1 - \frac{q_{d_u}(2q_{d_e} + q_{h_3} + q_{h_3})}{2^k}\right)$$

**Proof of Lemma 1:** Let $A_j$ be a Type-I IND-CCA2 adversary against the proposed CLGSC scheme with advantage $\varepsilon$. We show how adversary $A_j$ is used to construct a simulator $B$ that extracts $e(P, P)^{1/a}$ on input $(P, aP, a^2 P, ..., a^d P)$.

We proceed as in [5]. In the preparation phase, first $B$ selects $\ell \in R \{1, ..., q_{h_3}\}$, elements $\lambda_\ell \in R Z_p^*$,

$\mu_1, \mu_2, ..., \mu_{\ell - 1}, \mu_{\ell + 1}, ..., \mu_{q_{d_e}} \in R Z_p^*$

and expand the polynomial $g(x) = \prod_{i=1}^{q_{d_e}}(x + \mu_i)$ to obtain the coefficient $c_1, c_2, ..., c_{q_{d_e} - 1} \in Z_p^*$ such that
\[ g(x) = \sum_{i=0}^{q-1} c_i x^i. \] 
\( \mathcal{B} \) also computes \( \lambda_i = \lambda_i - \mu_i \in \mathbb{Z}_p^* \) for \( i = 1, \ldots, \ell - 1, \ell + 1, \ldots, q. \) 

Now \( \mathcal{B} \) sets \( G = \sum_{i=1}^{q-1} c_i(a^i P) = g(a)P \) as a public generator of \( \mathbb{G}_1 \) and computes another element \( U \in \mathbb{G}_1 \) as \( U = \sum_{i=1}^{q} c_{i-1}(a^i P) = aP. \) Note that \( \mathcal{B} \) does not know \( a. \) Furthermore \( \mathcal{B} \) computes 
\[ g_i(x) = \frac{g(x)}{(x + \mu_i)} = \sum_{j=0}^{q-2} d_{ij} x^j \]
for \( i = 1, \ldots, \ell - 1, \ell + 1, \ldots, q \) such that 
\[ \frac{1}{(a + \mu_i)} G = \frac{g(a)}{(a + \mu_i)} P = g_i(a)P = \sum_{j=0}^{q-2} d_{ij} (a^i P) \]
Thus \( \mathcal{B} \) can compute \( q - 1 = q_{h_1} \) pairs \( (\mu_i, S_i) = (1/(a + \mu_i)G) \) by the last term of the above equation. The system wide public key \( P_{pub} \) is chosen as \( P_{pub} = -U - \lambda_x G = (-a - \lambda_\ell)G \) with (unknown) private key \( z = -a - \lambda_\ell \in \mathbb{Z}_p^*. \) For all \( i = 1, \ldots, \ell - 1, \ell + 1, \ldots, q, \) \( \mathcal{B} \) have \( (\lambda_i, -S_i) = (\lambda_i, 1/(z + \lambda_i)G). \)

Now simulator \( \mathcal{B} \) starts interacting with \( \mathcal{A}_i \) on input \( \langle \mathbb{G}_1, \mathbb{G}_2, p, n, G, P_{pub}, e, g, H_1, H_2, H_3, H_4 \rangle \) where \( g = e(G, G) \). \( \mathcal{A}_i \) asks queries to \( \mathcal{B} \) throughout the simulation. It is assumed that \( H_1 \) queries are distinct and any query involving the identity \( ID \) comes after a \( H_1 \) query on \( ID \). The target identity \( ID_R^* \) is submitted to \( H_1 \) at some point of simulation. Also to maintain consistency in queries, \( \mathcal{B} \) makes the lists \( L_i \) for random oracle \( H_i \) for \( i = 1, 2, 3, 4 \) and \( L_{pk} \) for public key extraction oracle. \( \mathcal{B} \) initializes a counter \( \eta \) to 1 and start answering \( \mathcal{A}_i \)'s queries as follow:

- **\( H_1 \) queries**: it takes input an identity \( ID_\eta \), \( \mathcal{B} \) answers \( \lambda_\eta \) to the \( \eta^{th} \) such query and increment \( \eta \). \( \mathcal{B} \) sets the identity \( ID \) as \( ID_\eta \).

- **\( H_2 \) queries**: it takes input \( (m, g^r, ID_z, ID_\eta) \). \( \mathcal{B} \) checks the list \( L_2 \), it returns a previous value if it exists. Otherwise it chooses a random \( h_2 \in \mathbb{Z}_p^* \) and returns this value as the answer. It also stores the tuple \( (m, g^r, ID_z^*, ID_\eta, r, h_2) \) to \( L_2 \) list.

- **\( H_3 \) queries**: it takes input \( (m, g^r, X, PK_\zeta, PK_\eta, ID_z, ID_\eta, r) \). \( \mathcal{B} \) checks the list \( L_3 \), it returns a previous value if it exists. Otherwise it chooses a random \( h_3 \in \mathbb{Z}_p \) and returns this value as the answer. It also stores the tuple \( (m, g^r, X, PK_\zeta, PK_\eta, ID_z, ID_\eta, r, h_3) \) in the \( L_3 \) list.

- **\( H_4 \) queries**: it takes input \( (g^r, r'PK_\eta) \). \( \mathcal{B} \) checks the list \( L_4 \), it returns a previous value if it exists. Otherwise it chooses a random \( h_4 \in \mathbb{Z}_p \) and returns this value as the answer. It also stores the tuple \( (g^r, r'PK_\eta, r, h_4) \) in the \( L_4 \) list.

- **Partial Private Key Extraction queries**: it takes input an identity \( ID_\eta \). \( \mathcal{B} \) fails if \( \eta = \ell \) otherwise it knows that \( H_1(ID_\eta) = \lambda_\eta \) and returns \( D_\eta = -S_\eta = \lambda_\eta + z \) \( \mathcal{B} \) replaces it by \( \lambda_\eta, z \) \( \mathcal{B} \) takes input a plaintext \( m \) and returns \( (x_\eta, D_\eta) = (x_\eta, -S_\eta) \) where \( -S_\eta = 1/(\lambda_\eta + z) \).

- **Public Key Extraction queries**: it takes input an identity \( ID_\eta \). \( \mathcal{B} \) checks the list \( L_{pk} \), if \( (ID_\eta, x_\eta, PK_\eta) \in L_{pk} \) then return \( PK_\eta \), otherwise, \( \mathcal{B} \) picks randomly \( x_\eta \in \mathbb{Z}_p^* \) sets \( PK_\eta = x_\eta (P_{pub} + H_1(ID_\eta)G) \), then returns \( PK_\eta \) as the answer. \( \mathcal{B} \) stores the tuple \( (ID_\eta, x_\eta, PK_\eta) \) in the \( L_{pk} \) list.

- **Private Key Extraction queries**: it takes input an identity \( ID_\eta \). \( \mathcal{B} \) fails if \( \eta = \ell \) otherwise it checks the list \( L_{pk} \) and returns \( (x_\eta, D_\eta) = (x_\eta, -S_\eta) \) where \( -S_\eta = 1/(\lambda_\eta + z) \).

- **Public Key Replacement**: it takes input \( (ID_\eta, PK_\eta) \). \( \mathcal{B} \) checks the \( L_{pk} \), for the entry \( (ID_\eta, x_\eta, PK_\eta) \), if it exists \( \mathcal{B} \) replaces it by \( (ID_\eta, x_\eta, PK_\eta) \). Otherwise \( \mathcal{B} \) stores \( (ID_\eta, x_\eta, PK_\eta) \) in the \( L_{pk} \) list.

- **CLGSC queries**: it takes input a plaintext \( m \) and identities \( (ID_\xi, ID_\eta) = (ID_\zeta, ID_\eta) \) where \( \zeta, \eta \in \{1, \ldots, q_{h_1}\} \). If \( \zeta \neq \ell \), \( \mathcal{B} \) knows the sender’s
private key \((x_\alpha, -S_\alpha)\) and can answer the query by following the specification of CLGSC algorithm. So we assume that \(\zeta = \ell\), then \(B\) does the following:

i. Chooses \(h_1 \in R \mathbb{Z}^*_p, h_2 \in R \mathbb{Z}^*_p\) and \(Z \in R \mathbb{G}_1\).

ii. Computes \(e(Z, PK_\ell + h_1 T_e) e(G, G)^{-h_2} = \alpha\) where \(T_e = H_1(ID_\ell)G + P_{pub}\).

iii. Set \(H_2(m, \alpha, ID_\ell, ID_\eta) = h_2\) and stores in the \(L_2\) list.

iv. Computes \(X = h_2 T_\eta\) where \(T_\eta = H_1(ID_\eta)G + P_{pub}\).

v. Sets \(H_3(m, \alpha, X, PK_\ell, PK_\eta, ID_\ell, ID_\eta) = h_3\) and stores in \(L_3\) list.

vi. Simulate \(H_4\) as \(H_4(e(G, G)^{h_2} || h_2 PK_\eta) = h_4\) and stores in \(L_4\) list.

vii. Computes \(y = m||\alpha||Z\parallel ID_\ell \oplus \{h_4 f(ID_\eta)\}\).

viii. Returns \(\sigma = (y, X)\).

Note that if \(ID_\eta \neq ID_\phi\), \(B\) answers the CLGSC query in the same way using \(ID_\phi\) in place of \(ID_\eta\) and returns the signature \((m||\alpha||Z\parallel ID_\ell\parallel 0)\). Also \(B\) fails if \(H_2\) and \(H_3\) are already defined but this happens with a probability smaller than \((2q_c + q_h + q_\eta) / 2^k\).

- **CLG USC queries:** it takes input a ciphertext \((y, X)\) and a receiver’s identity \(ID_\eta\). If \(ID_\eta \neq ID_\phi\) then \(B\) knows receiver’s private key \((x_\eta, -S_\eta)\) (Note that if public key of \(ID_\eta\) is replaced then in this case \(B\) seek \(A_\ell\) for the corresponding secret value). \(B\) runs the CLG USC algorithm normally and returns the output to \(A_\ell\). Also if \(ID_\eta = ID_\phi\) then \(B\) is able to give appropriate answer to \(A_\ell\). If \(ID_\eta = ID_\ell\), then \(B\) reject the ciphertext. Across the whole game an inappropriate rejection occurs with probability at most \(q_{a} / 2^k\).

At the end of find stage, \(A_\ell\) outputs two messages \(m_h, m_l\), a sender’s identity \(ID_S^*\) and a receiver’s identity \(ID_R^*\) such that \(A_\ell\) has not made any partial private key and private key extraction query on \(ID_R^*\). Note that \(ID_S^*\) can be \(ID_\phi\). If \(ID_R^* \neq ID_\ell\), \(B\) aborts the simulation. Otherwise it picks \(\xi \in R \mathbb{Z}^*_p, y \in R \{0,1\}^{n + k_1 + k_2 + k_3}\) to return the challenge \(\sigma^* = (y, X)\) where \(X = -\xi G \in \mathbb{G}_1\). If we define \(\delta = \xi / a\) and since \(z = -a - \lambda_\ell\), we can check that

\[
X = -\xi G = -a \delta G = (\lambda_\ell + z) \delta G = \delta \lambda_\ell G + \delta P_{pub}
\]

\(A_\ell\) cannot recognize that \(\sigma^*\) is not a valid ciphertext unless she queries \(H_2, H_3\) or \(H_4\) involving \((e(G, G)^{\delta}, \delta T_e)\) where \(T_e = H_1(ID_\ell)G + P_{pub}\). Also in the guess stage, her view is simulated as before and her eventual output is ignored. Standard arguments can show that a successful \(A_\ell\) is very likely to query \(H_4\) on the input \((e(G, G)^{\delta}, \delta \lambda_\ell G + \delta P_{pub})\) if the simulation is indistinguishable from a real attack environment.

To produce a result, \(B\) fetches a random record from \(L_4\) list. As \(L_4\) contains no more than \((q_{h_4} + q_{e})\) records by construction thus with probability \((1/ (q_{h_4} + q_{e}))\). \(B\) chooses the record which will contains the right element

\[
e(G, G)^{\delta} = e(P, P)e(a^2 \xi / a)
\]

where \(G = g(a)P\).

The \(q\)-BDHIP solution can be extracted as follows, if \(\omega^* = e(P, P)^{1/a}\) then

\[
e(G, G)^{1/a}
= (\omega^*)^2 e \left( \sum_{i=0}^{q-2} c_{i+1} (a_i P), c_0 P \right)
= e(G, \sum_{j=0}^{q-2} c_{j+1} (a_j P))
\]

In the analysis of \(B\)’s advantage, following events will cause \(B\), to abort the simulation:

- \(E_1\): \(A_\ell\) does not choose to be challenged on \(ID_\ell\)
- \(E_2\): a partial private key extraction or private key extraction query is made on \(ID_\ell\)
- \(E_3\): \(B\) aborts in CLGSC query because of a collision on \(H_2\) and \(H_3\)
- \(E_4\): \(B\) rejects a valid ciphertext at some point of the game

We clearly have probability \(Pr[-E_1] = 1/ q_{h_4}\) and we know that \(-E_1\) implies \(-E_2\). Also \(Pr[E_3] \leq q_e (2q_c + q_{h_2} + q_{h_3}) / 2^k\) and \(Pr[E_4] \leq q_a / 2^k\). Thus we find that
\[ \Pr[\neg E_1 \land \neg E_2 \land \neg E_3] \geq \frac{1}{q_h} \left(1 - \frac{q_a}{2^k} \right) \left(1 - \frac{q_e (2q_e + q_{h_2} + q_{h_3})}{2^k} \right) \]

Also the probability that \( B \) select the correct record from \( L_4 \) list is \( \frac{1}{q(h_a + q_e)} \). Therefore advantage of \( B \) is
\[ e' > \frac{e}{q_h (q_{h_4} + q_e)} \left(1 - \frac{q_a}{2^k} \right) \left(1 - \frac{q_e (2q_e + q_{h_2} + q_{h_3})}{2^k} \right). \]

**Lemma 2:** Assume that an IND-CCA2 Type-II adversary \( A_{II} \) has an advantage \( \epsilon \) against the proposed CLGSC scheme, asking \( q_{h_0} \) queries to random oracle \( H_i (i = 1, 2, 3, 4) \) and \( q_e, q_u \) CLGSC queries, CLGUSC queries respectively. Then there is an algorithm \( B \) to solve the CDHP with probability \( e' > \frac{e}{q_h (q_{h_4} + q_e)} \left(1 - \frac{q_a}{2^k} \right) \left(1 - \frac{q_e (2q_e + q_{h_2} + q_{h_3})}{2^k} \right) \).

**Proof of Lemma 2:** Let \( A_{II} \) be a Type-II IND-CCA2 adversary against the proposed CLGSC scheme with advantage \( \epsilon \). We will show how adversary \( A_{II} \) is used to construct a simulator \( B \) that extract \( abP \) on input \( (P, aP, bP) \) for unknown \( a, b \in \mathbb{Z}_p^* \).

Simulator \( B \) chooses randomly \( s \in \mathbb{Z}_p^* \) as the master secret key and computes \( P_{pub} = sP \). Now \( B \) starts interacting with \( A_{II} \) on input \( \{\mathcal{G}_1, \mathcal{G}_2, p, n, P, P_{pub}, e, g, H_1, H_2, H_3, H_4\} \) where \( g = e(P, P) \). \( A_{II} \) asks queries to \( B \) throughout the simulation. It is assumed that \( H_1 \) queries are distinct and any query involving the identity \( ID \) comes after a \( H_1 \) query on \( ID \). The target identity \( ID_{\mathcal{R}} \) is submitted to \( H_1 \) at some point of simulation. Also to maintain consistency in queries, \( B \) makes the lists \( L_i \) for random oracle \( H_i \) where \( i = 1, 2, 3, 4 \) and \( L_{pk} \) as in Lemma 1. \( B \) randomly picks \( \ell \in \mathbb{Z}_p^* \) and start answering \( A_{II} \)'s queries as follows:

- **H1 queries:** it takes input an identity \( ID \). To the \( \eta \)th one such query \( B \) randomly chooses \( \lambda_\eta \in \mathbb{Z}_p^\ast \). \( B \) sets the identity \( ID \) as \( ID_\eta \) and stores \( (ID_{\eta}, \lambda_\eta) \) to \( L_\eta \) list.
- **H2, H3, H4 queries:** \( H_2, H_3, H_4 \) random oracles are simulated as in Lemma 1.
- **Public Key Extraction queries:** it takes input an identity \( ID_\eta \). \( B \) does the following
  i. If the entry \( (ID_\eta, x_\eta, PK_\eta) \in L_{pk} \) then return \( PK_\eta \).
  ii. Else, if \( \eta \in \ell \), \( B \) returns \( PK_\eta = (s + \lambda_\eta)aP \) and adds \( (ID_\ell, \perp, PK_\ell) \) to \( L_{pk} \) list.
- **Private Key Extraction queries:** it takes input an identity \( ID_\eta \). \( B \) fails if \( \eta \in \ell \) otherwise it finds \( (ID_\eta, x_\eta, PK_\eta) \) in \( L_{pk} \) list and returns \( (x_\eta, D_\eta) \) where \( D_\eta = 1 - \lambda_\eta + sP \).
- **CLGSC queries:** it takes input a plaintext \( m \) and identities \( (ID_3, ID_4) = (ID_\zeta, ID_\eta) \) where \( \zeta, \eta \in \{1, \ldots, q_{h_4}\} \). If \( \zeta \neq \ell \), \( B \) knows the sender’s private key \( (x_\zeta, D_\zeta) \) and can answer the query by following the specification of CLGSC algorithm. So we assume that \( \zeta = \ell \), then \( B \) does the following:
  i. Chooses \( h_3 \in \mathbb{Z}_p^\ast, h_2 \in \mathbb{Z}_p^\ast \) and \( Z \in \mathbb{G}_1 \)
  ii. Computes \( e(Z, PK_\ell + h_3 T_\ell) e(P, P)^{h_2} = \alpha \) where \( T_\ell = H_1(ID_\ell) G + P_{pub} \)
  iii. Sets \( H_2(m, a, ID_\ell, ID_\eta) = h_2 \) and store it in the \( L_2 \) list.
  iv. Computes \( X = h_3 T_\eta \) where \( T_\eta = H_1(ID_\eta) P + P_{pub} \)
  v. Sets \( H_3(m, a, X, PK_\ell, PK_\eta, ID_\ell, ID_\eta) = h_3 \) and store it in \( L_3 \) list.
  vi. Simulates \( H_4 \) as \( H_4(e(P, P)^{h_2} \parallel h_2 PK_\eta) = h_4 \)
  vii. Computes \( y = m \parallel a \parallel Z \parallel ID_\ell \oplus \{h_4 f(ID_\eta)\} \)
  viii. Returns \( \sigma = (y, X) \)

Note that if \( ID_\eta = ID_\varphi \), then \( B \) answers the CLGSC query in the same way using \( ID_\varphi \) in place of \( ID_\eta \) and returns the signature \( (m \parallel a \parallel Z \parallel ID_\ell, \varnothing) \). Also \( B \) fails if \( H_2 \) and \( H_3 \) are already defined but this
happens with a probability smaller than 
\((2q_e + q_h + q_{b_{1}})/2^k\).

- **CLGSC queries**: it takes input a ciphertext 
\((y, X)\) and a receiver’s identity \(ID_{\eta}\). If 
\(ID_{\eta} \neq ID_{\ell}\) then \(\mathcal{B}\) knows receiver’s private key 
\((x_{\eta}, D_{\eta})\). \(\mathcal{B}\) runs the UCLGSC algorithm normally and returns the output to \(A_{II}\). Also if \(ID_{\eta} = ID_{\phi}\) then \(\mathcal{B}\) is able to give appropriate answer to \(A_{II}\). If \(ID_{\eta} = ID_{\ell}\), then \(\mathcal{B}\) reject the ciphertext. Across the whole game an inappropriate rejection occurs with probability at most \(q_{u}/2^k\).

At the end of find stage, \(A_{II}\) outputs two messages \(m_{0}, m_{1}\), a sender’s identity \(ID_{S}^{\ast}\) and a receiver’s identity \(ID_{R}^{\ast}\) such that \(A_{II}\) has not made any private key extraction query on \(ID_{R}^{\ast}\). Note that \(ID_{S}^{\ast}\) can be \(ID_{0}\). If \(ID_{R}^{\ast} \neq ID_{\ell}\), \(\mathcal{B}\) aborts the simulation. Otherwise it picks, \(y \in_R \{0, 1\}^{n+k_1+k_2+k_3}\) and sets \(X = (s + H_1(ID_{\ell}))bP\) to return the challenge \(\sigma^{\ast} = (y, X)\).

Note that \(A_{II}\) cannot recognize that \(\sigma^{\ast}\) is not a valid ciphertext unless \(A_{II}\) queries \(H_2, H_3\) or \(H_4\) involving \((e(X, D_{\ell}), x_{\ell}, X)\). Also in the guess stage, \(A_{II}\)’s view is simulated as before and her eventual output is ignored. Standard arguments can show that a successful \(A_{II}\) is very likely to query \(H_4\) on the input \((e(X, D_{\ell}), x_{\ell}, X)\) if the simulation is indistinguishable from a real attack environment.

To produce a result, \(\mathcal{B}\) fetches a random record from \(L_{4}\) list. As \(L_{4}\) contains no more than \((q_{h_{4}} + q_{e})\) records by construction thus with probability \(1/(q_{h_{4}} + q_{e})\), \(\mathcal{B}\) chooses the record which will contains the right element \(Q = x_{\ell}X = (s + H_1(ID_{\ell}))abP\). Then \(\mathcal{B}\) returns

\[
abP = (s + H_1(ID_{\ell}))^{-1}Q
\]

as the solution of CDHP.

In the analysis of \(\mathcal{B}\)’s advantage, following events will cause \(\mathcal{B}\), to abort the simulation:

- \(E_1\): \(A_{II}\) does not choose to be challenged on \(ID_{\ell}\)
- \(E_2\): a private key extraction query is made on \(ID_{\ell}\)
- \(E_3\): \(\mathcal{B}\) aborts in CLGSC query because of a collision on \(H_2\) or \(H_3\)
- \(E_4\): \(\mathcal{B}\) rejects a valid ciphertext at some point of the game

We clearly have probability \(Pr[\neg E_1] = 1/q_{h_{4}}\) and we know that \(\neg E_1\) implies \(\neg E_2\). Also \(Pr[E_3] \leq q_e(2q_e + q_{h_{2}} + q_{h_{3}})/2^k\) and \(Pr[E_4] \leq q_{u}/2^k\). Thus we find that

\[
Pr[\neg E_1 \land \neg E_3 \land \neg E_4] \geq \frac{1}{q_{h_{4}}}(1 - \frac{q_{u}}{2^k})\left(1 - \frac{q_e(2q_e + q_{h_{2}} + q_{h_{3}})}{2^k}\right)
\]

Also the probability that \(\mathcal{B}\) select the correct record from \(L_{4}\) list is \(1/(q_{h_{4}} + q_{e})\). Therefore advantage of \(\mathcal{B}\) is

\[
e^{\ast} > \frac{e^{\ast}}{q_{h_{4}}(q_{h_{4}} + q_{e})}\left(1 - \frac{q_{u}}{2^k}\right)\left(1 - \frac{q_e(2q_e + q_{h_{2}} + q_{h_{3}})}{2^k}\right)
\]

**Theorem 2**: The proposed CLGSC scheme is EUF-CMA secure under the k-CAA and Inv-CDHP.

To prove this theorem we are going to use the technique of \([6, 13]\) which does not require the use of forking lemma. The theorem follows from the lemma 3 and 4.

**Lemma 3**: Assume that an EUF-CMA Type-I adversary \(A_{I}\) has an advantage \(e\) against the proposed CLGSC scheme, asking \(q_{h_{4}}\) queries to random oracle \(H_i\) \((i = 1, 2, 3, 4)\) and \(q_{e}\), \(q_{u}\) CLGSC queries, CLGSC queries respectively. Then there is an algorithm \(\mathcal{B}\) to solve the k-CAA problem in \(G_1\) with probability

\[
e^{\ast} > \left(e - \frac{1}{2^k}\right)\frac{1}{q_{h_{4}}}\left(1 - \frac{q_{u}}{2^k}\right)\left(1 - \frac{q_e(2q_e + q_{h_{2}} + q_{h_{3}})}{2^k}\right)
\]

**Proof of Lemma 3**: Let \(A_I\) be the Type-I EUF-CMA adversary against the proposed CLGSC scheme with advantage \(e\). We will show how adversary \(A_I\) is used to construct a simulator \(\mathcal{B}\) that solve the k-CCA problem given input \(P, R(\approx sp) \in G_1\), \(\lambda_1, \lambda_2, \ldots, \lambda_q \in \mathbb{Z}_p^\ast\) and \(\frac{1}{s+\lambda_{1}}, \frac{1}{s+\lambda_{2}}, \frac{1}{s+\lambda_{3}}, \ldots, \frac{1}{s+\lambda_{q}}\) i.e. \(\mathcal{B}\) outputs a pair \((\lambda, \frac{1}{s+\lambda})\) for \(\lambda \notin \{\lambda_1, \lambda_2, \ldots, \lambda_q\}\) where \(q = q_{h_{4}}-1\).
Simulator \( \mathcal{B} \) starts interacting with \( \mathcal{A}_f \) on input \( \{G_1, G_2, p, n, P, P_{pub} = R(= s P), e, g, H_1, H_2, H_3, H_4 \} \) where \( g = e(P, P) \). \( \mathcal{A}_f \) asks queries throughout the simulation. It is assumed that \( H_1 \) queries are distinct and any query involve the identity \( ID \) comes after a \( H_1 \) query on \( ID \). Also to maintain consistency in queries, \( \mathcal{B} \) makes the lists \( L_4 \) for random oracle \( H_1 \) where \( i = 1, 2, 3, 4 \) and \( L_{pk} \) as in Lemma 1. \( \mathcal{B} \) randomly picks \( \ell \in R \{1,\ldots,q_{\eta}\} \) and start answering \( \mathcal{A}_f \)'s queries as follows:

- **\( H_1 \) queries**: it takes input an identity \( ID \). If \( \eta = \ell \), \( \mathcal{B} \) returns the value \( \lambda \not\in \{\lambda_1, \lambda_2, \ldots, \lambda_q\} \). \( \mathcal{B} \) sets the identity \( ID \) as \( ID_\ell \) and store (\( ID_\ell, \lambda, \downarrow \)) to \( L_4 \) list. Otherwise \( \mathcal{B} \) answers \( \lambda_\eta \in \{\lambda_1, \lambda_2, \ldots, \lambda_q\} \) to the \( \eta^{th} \) one such query and increment \( \eta \). \( \mathcal{B} \) sets the identity \( ID \) as \( ID_\eta \) and store (\( ID_\eta, \lambda_\eta, \frac{1}{s + \lambda_\eta} P \)) to \( L_4 \) list.

- **\( H_2, H_3, H_4 \) queries**: \( H_2, H_3, H_4 \) random oracles are simulated as in Lemma 1.

- **Partial Private Key Extraction queries**: it takes input an identity \( ID_\eta \). \( \mathcal{B} \) fails if \( \eta = \ell \) and abort the simulation. Otherwise \( \mathcal{B} \) check the list \( L_4 \) to find the entry (\( ID_\eta, \lambda_\eta, \frac{1}{s + \lambda_\eta} P \)) and returns \( \frac{1}{s + \lambda_\eta} P \) as the corresponding private key \( D_\eta \).

- **Public Key Extraction queries**: same as in lemma 1.

- **Private Key Extraction queries**: it takes input an identity \( ID_\eta \). \( \mathcal{B} \) fails if \( \eta = \ell \) otherwise it checks \( L_4 \) and \( L_{pk} \) for the tuples (\( ID_\eta, \lambda_\eta, \frac{1}{s + \lambda_\eta} P \)) and (\( ID_\eta, x_\eta, PK_\eta \)) , \( \mathcal{B} \) returns
\[
SK_\eta = (x_\eta, D_\eta) = (x_\eta, \frac{1}{s + \lambda_\eta} P).
\]

- **Public Key Replacement**: same as in lemma 1.

- **CLGSC queries**: it takes input a plaintext \( m \) and identities (\( ID_\zeta, ID_\gamma \)) where \( \zeta, \eta \in \{1,\ldots,q_{\eta}\} \). If \( \zeta = \ell \), \( \mathcal{B} \) knows the sender’s private key \( (x_\zeta, D_\zeta) \) and can answer the query by following specification of CLGSC algorithm. So we assume that \( \zeta = \ell \), then \( \mathcal{B} \) does the same as in CLGSC queries of Lemma 1.

Note that if \( ID_\eta = ID_\phi \), \( \mathcal{B} \) answers the CLGSC query in the same way using \( ID_\phi \) in place of \( ID_\eta \) and returns the signature \((m || \alpha || Z || ID_\ell, \mathcal{O}) \).

Also \( \mathcal{B} \) fails if \( H_2 \) and \( H_3 \) are already defined but this happens with a probability smaller than \((2q_\ell + q_{\phi} + q_{\eta})/2^k\).

- **CLGUSC queries**: it takes input a ciphertext \((y, X)\) and a receiver’s identity \( ID_\eta \). If \( ID_\eta \neq ID_\ell \), then \( \mathcal{B} \) knows receiver’s private key \((x_\eta, D_\eta)\) (Note that if public key of \( ID_\eta \) is replaced then in this case \( \mathcal{B} \) ask \( \mathcal{A}_f \) for the corresponding secret value). \( \mathcal{B} \) runs the CLGUSC algorithm normally and returns the output to \( \mathcal{A}_f \). Also if \( ID_\eta = ID_\phi \) then \( \mathcal{B} \) is able to give appropriate answer to \( \mathcal{A}_f \). If \( ID_\eta = ID_\ell \), then \( \mathcal{B} \) reject the ciphertext. Across the whole game an inappropriate rejection occurs with probability at most \( q_{\eta}/2^k \).

Finally \( \mathcal{A}_f \) stops and outputs \((y^*, X^*, ID_R^*)\) such that \((y^*, X^*)\) is not the output of CLGSC algorithm under the receiver’s identity \( ID_R^* \). If \( X^* \neq \mathcal{O} \), \( ID_R^* \neq ID_\phi \) and because of irreflexivity assumption \( ID_R^* \) cannot be \( ID_\ell \). Thus \( \mathcal{B} \) can extract the message signature pair \((m^*, \alpha^*, Z^*, ID_S^*)\) from \((y^*, X^*)\). If \( X^* = \mathcal{O} \) then \( ID_R^* = ID_\phi \), also in this case \( \mathcal{B} \) have message signature pair \((m^*, \alpha^*, Z^*, ID_S^*)\).

In both cases \((m^*, \alpha^*, Z^*, ID_S^*)\) must satisfy the verification equation. If \( ID_S^* \neq ID_\ell \), \( \mathcal{B} \) abort the simulation. Otherwise \( \mathcal{B} \) recovers the tuple \((ID_\ell, \lambda, PK_\ell, x_\ell)\) from the \( L_{pk} \) list and the tuple \((m^*, g^*, X^*, PK_\ell, PK_R^*, ID_\ell, ID_R^*, r^*, b_3^*)\) form the \( L_3 \) list. Then we have
\[
e(Z^*, PK_\ell + h_3^* T_\ell) = \alpha^* g^{r'}
\]
where, \( b_3^* = H_3(m^*, \alpha^*, X^*, PK_\ell, PK_R^*, ID_\ell, ID_R^*) \) and
\( r' = H_2(m^*, \alpha^*, ID_\ell, ID_R^*) \).
\[
e(Z^*, PK_\ell + h_3^* T_\ell) = (e(P, P)^{y^* + r'})
\]
\[
e(Z^*, (x_\ell + h_3^*) (s + \lambda) P) = (e(P, P)^{y^* + r'})
\]
\[
e \left( \frac{(x_i + h_3^*) (s + \lambda)}{r^* + r'} \right) Z^* \right) = e(P, P)\
\]

Hence \( \mathcal{B} \) can successfully compute \( \frac{1}{(s + \lambda)} P = (x_i + h_3^*) Z^* \) and output the pair \( \left( \lambda, \frac{1}{(s + \lambda)} P \right) \) for \( \lambda \not\in \{ \lambda_1, \lambda_2, ..., \lambda_q \} \) as a solution of the \( k \)-CAA problem in \( \mathbb{G}_1 \).

In the analysis of \( \mathcal{B} \)'s advantage, following events will cause \( \mathcal{B} \) to abort the simulation:

- \( E_1 \): If \( ID^*_5 \not= ID^*_1 \), in the forged ciphertext \( E_2 \): a partial private key or private key extraction query is made on \( ID^*_i \).
- \( E_3 \): \( \mathcal{B} \) aborts in CLGSC query because of a collision on \( H_2 \) or \( H_3 \).
- \( E_4 \): \( \mathcal{B} \) rejects a valid ciphertext at some point of the game.

We clearly have probability \( \Pr[-E_1] = 1/q_{h_3} \) and we know that \( -E_1 \) implies \( -E_2 \). Also \( \Pr[E_3] \leq q_u (2q_e + q_{h_2} + q_{h_3}) / 2^k \) and \( \Pr[E_4] \leq q_u / 2^k \).

Thus we find that
\[
\Pr[-E_1 \land -E_3 \land -E_4] \geq \frac{1}{q_{h_3}} \left( 1 - \frac{q_u}{2^k} \right) \left( 1 - \frac{q_e (2q_e + q_{h_2} + q_{h_3})}{2^k} \right).
\]

It is remains to calculate the probability for which the simulation is not perfect. The only event where it can happen is that \( \mathcal{A}_H \) forges a valid signature of \( ID^*_i \) without making \( H_3 \) query with input \((m^*, a^*, X^*, PK_i, PK^*_R, ID^*_1, ID^*_R)\). Clearly the probability to generate the valid signature without asking \( H_3 \) query is at most \( 1/2^k \). Therefore the advantage of \( \mathcal{B} \) is
\[
\varepsilon' \left( e - \frac{1}{2^k} \right) \frac{1}{q_{h_3}} \left( 1 - \frac{q_u}{2^k} \right) \left( 1 - \frac{q_e (2q_e + q_{h_2} + q_{h_3})}{2^k} \right).
\]

**Lemma 4**: Assume that an EUF-CMA Type-II adversary \( \mathcal{A}_H \) has an advantage \( \varepsilon \) against the proposed CLGSC scheme, asking \( q_{h_3} \) queries to random oracle \( H_i (i = 1, 2, 3, 4) \) and \( q_e, q_{h_2} \) CLGSC queries, CLGUSC queries respectively. Then there is an algorithm \( \mathcal{B} \) to solve the Inv-CDH problem in \( \mathbb{G}_1 \) with probability
\[
\varepsilon' \left( e - \frac{1}{2^k} \right) \frac{1}{q_{h_3}} \left( 1 - \frac{q_u}{2^k} \right) \left( 1 - \frac{q_e (2q_e + q_{h_2} + q_{h_3})}{2^k} \right).
\]

**Proof of Lemma 4**: Let \( \mathcal{A}_H \) be the Type-II EUF-CMA adversary against the proposed CLGSC scheme with advantage \( \varepsilon \). We will show how adversary \( \mathcal{A}_H \) is used to construct a simulator \( \mathcal{B} \) that solve Inv-CDH problem with given input \( P, aP \in \mathbb{G}_1 \) where \( a \) is unknown i.e. \( \mathcal{B} \) outputs a pair \( (h^*_3, \frac{1}{a + h^*_3} P) \) for some \( h^*_3 \in \mathbb{Z}_p^* \).

Simulator \( \mathcal{B} \) chooses randomly \( s \in_R \mathbb{Z}_p^* \) as the master secret key and computes \( P_{pub} = sP \). Now \( \mathcal{B} \) start interaction with \( \mathcal{A}_H \) on input \( \langle \mathbb{G}_1, \mathbb{G}_2, p, n, P_{pub}, e, g, H_1, H_2, H_3, H_4 \rangle \) where \( g = e(P, P) \). \( \mathcal{A}_H \) asks queries to \( \mathcal{B} \) throughout the simulation. It is assumed that \( H_1 \) queries are distinct and any query involving the identity \( ID \) comes after a \( H_1 \) query on \( ID \). Also to maintain consistency of queries, \( \mathcal{B} \) makes the lists \( L_i \) for random oracle \( H_i \) where \( i = 1, 2, 3, 4 \) and \( L_{pk} \) as in Lemma 1. \( \mathcal{B} \) randomly picks \( \ell \in_R \{ 1, ..., q_{h_3} \} \) and start answering \( \mathcal{A}_H \)'s queries. All \( H_1, H_2, H_3, H_4 \) hash oracles queries, Public and Private Key Extraction queries, CLGSC queries and CLGUSC queries are answered as in the proof of Lemma 2.

Finally \( \mathcal{A}_H \) stops and outputs \( (y^*, X^*, ID^*_R) \) such that \( (y^*, X^*) \) is not the output of CLGSC algorithm under the receiver’s identity \( ID^*_R \). If \( X^* \neq O \), \( ID^*_R \neq ID^*_\phi \) and because of irreflexivity assumption \( ID^*_R \) cannot be \( ID^*_i \). Thus \( \mathcal{B} \) can extract the message signature pair \( (m^*, a^*, Z^*, ID^*_3) \) from \( (y^*, X^*) \). If \( X^* = O \) then \( ID^*_R = ID^*_\phi \), also in this case \( \mathcal{B} \) have message signature pair \( (m^*, a^*, Z^*, ID^*_S) \). In both cases \( (m^*, a^*, Z^*, ID^*_S) \) must satisfy the verification equation. \( \mathcal{B} \) aborts the simulation if \( ID^*_S \neq ID^*_i \). Otherwise \( \mathcal{B} \) recovers the tuple.
(IDr, λ, PKr, ⊥) from Lpk list and the tuple (m∗, g∗, X∗, PKr, PKRr, IDr, IDRr, r∗, h∗) form L2 list. Then we have
\[ e(Z^*, PKr + h∗3T1) = α^* g^* \]
where, \[ h∗3 = H3(m∗, α∗, X∗, PKr, PKRr, IDr, IDRr) \] and \[ r∗ = H2(m∗, α∗, IDr, IDRr). \]
\[ e(Z^*, PKr + h∗3T1) = e(P, P)^{r∗ + r'} \]
\[ e(Z^*, (a + h∗3)(s + λr))P = e(P, P)^{r∗ + r'} \]
\[ e \left( \frac{(a + h∗3)(s + λr)}{r + r'} \right) Z^* P = e(P, P) \]
Hence \( B \) can successfully compute \[ \frac{1}{(a + h3)} P = (s + λr)Z^* \] where \( h3 \in \mathbb{Z}_p \) as a solution of Inv-CDH problem in \( \mathbb{G}_1 \).
Analysis of \( B \)'s advantage is similar to the proof of Lemma 3.

7. Conclusion
In this paper, we first gave the simplified security notions for the CLGSC schemes and showed that Ji et al. [16] CLGSC scheme is not existentially unforgeable against insider Type-I adversary. We also proposed a new and efficient CLGSC scheme and proved its security in the random oracle model. As Ji et al. [16] CLGSC scheme is not secure, thus proposed scheme is the first provable secure certificateless generalized signcryption scheme.

References:


