International Journal of Pure and Applied Mathematics
Volume 120 No. 5 2018, 1385-1404
ISSN: 1314-3395 (on-line version)
url: http://www.acadpubl.eu/hub/
Special Issue

http://www.acadpubl.eu/hub/

Efficient and Provably Secure Certificateless Aggregate Signature Scheme from Bilinear Pairings

N. B. Gayathri[1], P Vasudeva Reddy[1]

[1] Department of Engineering Mathematics, Andhra University, Visakhapatnam, India.

gayatricrypto@gmail.com, vasucrypto@andhrauniversity.edu.in.

Abstract:

The most important contribution of modern cryptography is the invention of digital signatures. To deal with specific application scenarios, digital signature schemes have been evolved with different variants. One of such variant is aggregate signature scheme, which allows aggregation of different signatures by different users on different messages, to achieve computational and communication efficiency. Such schemes are useful in the design of Wireless Sensor Networks (WSN), Mobile Ad-hoc Networks (MANETS), and Vehicular Ad-hoc Networks (VANETS); where storage, bandwidth and computational complexity are major constraints. In order to improve the computational and communicational efficiency along with security, in this paper, we propose a novel Certificateless Aggregate Signature (CLAS) scheme and extended it to achieve full aggregation. The proposed CLAS scheme is using bilinear pairings over elliptic curves and is proven secure in Random Oracle Model under the assumption of Computational Diffie-Hellman Problem is hard. The security of the proposed CLAS scheme is proven without using Forking Lemma to achieve tight security. We compared our scheme with well known existing schemes. Efficiency analysis shows that our scheme is much efficient than existing schemes in terms of communication and computational costs.

Keywords: Public Key Cryptography, Certificateless Signature, Aggregate Signatures, Bilinear Pairing, ROM Security Model, CDH Problem.

1. Introduction

Digital signatures are the most important cryptographic primitive enabled by Public Key Cryptography (PKC) and are building blocks of many applications like, e-commerce, e-auction, e-voting, web browsing etc, by providing the authentication and integrity of data. Many signature schemes and their variants have been proposed in traditional and other cryptographic settings. The concept of PKC was proposed by Diffe and Hellman [1] in 1976, in which the authentication of public key relies on the certificate issued by Certificate Authority (CA). But certificate management leads to extra storage, large computation and communication costs. To overcome such difficulties in traditional PKC, the concept of Identity–based PKC (ID-PKC) was introduced by Shamir in 1984 [2]. In this system, public key of a user is directly derived from user's identity and the secret key is generated by a trusted third party called Private Key Generator / Key Generation Centre (PKG/KGC) using user's public key. Though this system abolishes the problems in certificate management, it suffers from inherent key escrow problem i.e. the trusted third party who knows user's secret key can impersonate the user. To overcome afore mentioned difficulties in PKC and ID-PKC, Al-Riyami [3] presented a new structure called Certificateless Public Key Cryptography (CL-PKC) in 2003. In this system, the full private key of a user is divided into two parts. The first part, called partial private key, is controlled by a key generator centre (KGC). The second part is chosen by the user himself and remains

secret to the KGC. Therefore, to discuss the security issues of CL-PKC, there are two types of attacks, depending on which part of the private key is compromised [4].

To deal with different scenarios, digital signature schemes have evolved into many different variants. One of such variants is aggregate signature. The aggregate signature scheme allows *n* signatures on *n* distinct messages from *n* distinct users to aggregate a single signature. The aggregate signature schemes could be used in many applications such as wireless sensor networks, secure routing protocols, banking transactions, traffic control, military applications etc., where storage, bandwidth and computational complexity are of major constraints. There are two types of aggregation. If a part of a signature is aggregated, particularly the part with secret key component is aggregated without aggregating the randomness part is called partial aggregation. If randomness part is also aggregated completely then the aggregation is called full aggregation. Combination of aggregate signature technique with certificateless setting integrates the advantages of both.

1.1. Related Work

After the invention of certificateless PKC by Al Riyami [3], many schemes were proposed in the state-of-the-art of certificateless cryptography. In 2003, Boneh et al. [5] introduced the concept of Aggregate signature and the first Certificateless Aggregate Signature (CLAS) scheme was presented by Castro et al. [6] in 2007. Since then, many CLAS schemes [7-20] have been proposed by different researchers. But most of these CLAS schemes [7],[11],[14],[16],[18] require relatively more number of pairing operations in verification process and these operations increases linearly with the number of signers in aggregation (verification) process and deviates from the goal of aggregate signatures. Later, researchers proposed CLAS scheme with fixed paring operations in aggregate verification [8], [9], [10], [12], [21], [15], [22], [17], [19]. In 2012, Xiong et al. [17] developed a CLAS scheme, in which the verification equation requires constant number of pairings (independent of signers). But, it is not secure due to the Type I and Type II attacks as presented in [8],[9], [22], [23] and coalition attacks (Coalition attack is the property that a group of signers containing KGC together can generate a valid aggregate certificateless signature.) presented in [18]. In [8], [9], [22] a new CLAS schemes were proposed to improve the security and efficiency. But the scheme proposed in [8] is insecure due to the attacks presented in [24], [25]. Recently, A. Fan et al. [26] and J. Li et al. [27] individually pointed out that CLAS scheme proposed by D. He et al. [23] is insecure against Type II adversary and proposed an improved CLAS scheme.

In 2014, M. Zhou et al. [16] proposed a compact CLAS scheme and proved its security in ROM under the CDH problem. This scheme achieves full aggregation. Unfortunately, Chen et al. [28] showed that the scheme proposed by M. Zhou et al. [16] is not secure against strong type-I adversary. In 2015, J. Deng et al. [10] showed that the scheme of Hou et al. [29] is vulnerable to type II adversary and presented an improved CLAS scheme with enhanced security. In the same year A. K. Malhi et al. [15] and S. J Horng et al. [12] have separately proposed two CLAS schemes for Vehicular Ad-hoc Networks (VANETS). But, scheme [12] was cryptanalysed by J li [30] in 2016 and improved their scheme. In 2016, B. Kang et al [14] has cryptanlyzed Lei Zhang et al. scheme [20] by coalition attack and proposed a CLAS scheme and claimed that their scheme is secure against coalition attack. In 2016,H. Nie et al. [31] proposed a novel and efficient CLAS scheme and proved its security in random oracle model. In the same year N. Pakniat et al. [32] proved that Nie et al. [31] scheme is insecure due to public key replacement attack by Type I adversary. In 2017, Kang et al. [33] presented a new CLAS scheme with security analysis. But this scheme is insecure due to Type II adversary. In 2017, P. Kumar et al. [34] presented a review on CLAS scheme. In 2017, P. Kumar et al. [35] presented a CLAS scheme for healthcare wireless sensor networks. But this scheme is also insecure due to Type II adversary.

1.2. Our Contribution

In this paper, to improve the efficiency along with security, we propose a secure and efficient CLAS scheme and extended it to achieve Full aggregation. These schemes are designed using bilinear pairings over elliptic curves. The CLAS scheme requires constant number of pairing operations in aggregate verification process, which improves the computational efficiency. The proposed CLAS scheme is secure in random oracle paradigm under the assumption that the CDH problem is hard. Moreover the security proofs are made without using forking lemma [36] to achieve tight security. Hence the proposed CLAS scheme is much efficient and more secure than all the existing CLAS schemes.

1.3. Organization

The remaining part of this paper is organized as follows. In Section 2 we presented some preliminaries. In Section 3 we presented the syntax and security model for our CLAS scheme. In Section 4 we presented our proposed CLAS scheme with its security analysis. Extension of CLAS scheme to Full aggregation is presented in Section 5. In Section 6 we presented the efficiency analysis of the proposed schemes. Finally, in Section 7 we presented conclusions.

2. Preliminaries

In this section we briefly describe the fundamental concepts on bilinear pairings and the complexity assumption, on which the proposed scheme is designed and achieves the desired security.

2.1 Bilinear Pairings

Let G_{Adt} and G_{Mlt} be additive and multiplicative cyclic groups respectively of same prime order q. Let P be the generator of G_{Adt}. A bilinear pairing is a map $e: G_{Adt} \times G_{Adt} \to G_{Mlt}$ which satisfies the following properties:

1. **Bilinearity:** The map $e: G_{Adt} \times G_{Adt} \to G_{Mlt}$ is bilinear if $e(A+B,C) = e(A,C)e(B,C)$, for all $A, B, C \in G_{Adt}$ and $u, v \in Z_q^*$. Also $e(uA, vB) = e(A,B)^{uv}$.
2. **Non-degeneracy:** If P is a generator of G_{Adt} then $e(P,P)$ is a generator of G_{Mlt}. i.e. $\exists A \in G_{Adt} \ni e(A,A) \neq 1$.
3. **Computability:** There exists an efficient algorithm to find $e(A,B) \forall A, B \in G_{Adt}$.

2.2 Computational Diffie-Hellman (CDH) Problem: Given a random instance $P, uP, vP \in G_{Adt}$ for any $u, v \in Z_q^*$, it is to compute $uvP \in G_{Adt}$.

Notations and their meanings which we used throughout this paper are presented in the following Table 1.

Table 1: Notations and their meanings

Notation	Meaning
l, s	Security parameter & master secret key of the system generated by KGC.
τ	System Parameter.
z_q^*	The group with elements 1,2…q-1 under addition modulo q.
G_{Adt}, G_{Mlt}	Additive & Multiplicative cyclic groups of same prime order q.

$H_i, i = 1,2,3,4,5$	Cryptographic one way hash functions.
ID	Users Identity
$UPSK_{ID}, USK_{ID}, UPK_{ID}$	User partial secret key, User secret key &User public key of the identity respectively.
$\mathrm{ADV}_1, \mathrm{ADV}_2$	Type-I & Type-II adversaries respectively.
ξ	An algorithm to solve CDH problem by using adversaries
$e: G_{Adt} \times G_{Adt} \to G_{Mlt}$	An admissible bilinear map.
Ω	Signature on a message.

3 Syntax and Security Model

In this section, we present the syntax and security model for CLAS scheme.

3.1 Syntax of CLAS Scheme

A formal model of the proposed CLAS scheme consists of six components whose functionalities are described as follows.

- **Master Key Gen:** KGC runs this algorithm by taking $l \in Z^+$ as input and generates τ, s and master public key.
- **Partial Key Gen:** KGC runs this algorithm by taking *ID* as input and generates $UPSK_{ID}$.
- **User Key Gen:** User runs this algorithm by taking master public key, *ID* as input and generates USK_{ID}, UPK_{ID}.
- **Signature Generation:** Signer runs this algorithm by taking $USK_{ID}, UPSK_{ID}$ and message $m \in \{0,1\}^*$ as input and generates a signature Ω on a message $m \in \{0,1\}^*$.
- **Aggregate:** An aggregate signature generator (either 3rd party or one of the signers) runs this algorithm by taking various signatures $(\Omega_i)_{i=1\text{to}\,n}$ from different users $(\mathcal{U}_i)_{i=1\text{to}\,n}$ with identities $(ID_i)_{i=1\text{to}\,n}$ and their corresponding public keys $(UPK_{ID_i})_{i=1\text{to}\,n}$ and generate the aggregate signature Ω_{agg} for messages $(m_i)_{i=1\text{to}\,n}$.
- **Aggregate Verify:** By taking master public key, aggregate set of users $(\mathcal{U}_i)_{i=1\text{to}\,n}$ with identities $(ID_i)_{i=1\text{to}\,n}$ and corresponding $(UPK_{ID_i})_{i=1\text{to}\,n}$ and an aggregate signature Ω_{agg} on messages $(m_i)_{i=1\text{to}\,n}$, any verifier can run this algorithm to check the validity of aggregate signature. It out puts true if the signature is valid or $\perp$ otherwise.

3.2 Security Model of CLAS Scheme

As described in [4], based on the potential adversary behaviour, we consider the following types of adversaries.

1) Type I Adversary: Key Replacement Attack: The Adversary cannot access master secret key but can compromise user's secret value or capable to replace the public key of any user with a value of his choice.
2) Type II Adversary: Malicious KGC Attack: The Adversary can access master secret key but cannot replace the public key of any user.

The Existential unforgeability of a CLAS scheme can be defined by considering the following two games Game-I and Game-II against Type-I and Type-II adversaries.

Game-I: This game is executed between the challenger ξ and an adversary ADV_1 as follows.

- **Initialization Phase:** In this phase, challenger ξ runs Master Key Gen algorithm to get τ, s and master public key. The challenger then gives τ and master public key to the ADV_1 by keeping s secret.
- **Queries Phase:** In this phase, ADV_1 makes queries on the following oracles.

***Reveal Partial Secret Key Oracle*:** On receiving a query from ADV_1, the challenger ξ computes $UPSK_{ID}$ by taking ID as input and gives this to ADV_1.

***Create User Oracle*:** On receiving a query from ADV_1, the challenger ξ computes UPK_{ID} by taking ID as input and gives this to ADV_1.

***Reveal Secret Key Oracle*:** After receiving a query from ADV_1, the challenger ξ returns USK_{ID} by taking ID as input.

Replace Public Key Oracle: ADV_1 may replace current UPK_{ID} with the required UPK'_{ID} by giving ID and UPK'_{ID}.

Sign Oracle: On receiving a query from adversary ADV_1, signing oracle returns a valid signature Ω signed by current public/private key of the user ID, by taking ID, UPK_{ID} with message $m \in \{0,1\}^*$ as input.

- **Forgery Phase:** Finally ADV_1 out puts Ω^*_{agg} as forgery on messages $(m^*_i)_{i=1\text{to }n}$, under the identities $(ID^*_i)_{i=1\text{to }n}$ and the corresponding $(UPK^*_{ID_i})_{i=1\text{to }n}$ and wins the game if (i) Ω^*_{agg} is a valid signature.

(ii) Partial Secret Key Oracle, the Secret Key Oracle have never involved in this game for at least one of the $(ID^*_i)_{i=1\text{to }n}$, say (ID^*_1).

(iii) Sign Oracle has never been involved in this game for (ID^*_1, m^*_1).

Game-II: This game is executed between the challenger ξ and an adversary ADV_2 as follows.

- **Initialization Phase:** In this phase, challenger ξ runs Master Key Gen algorithm to get τ, s and master public key. The challenger then gives τ, s and master public key to the ADV_2.
- **Queries Phase:** In this phase, ADV_2 makes queries on the following oracles.

Create User Oracle: On receiving a query from ADV_2, the challenger ξ computes UPK_{ID} by taking ID as input and gives this to ADV_2.

Reveal Secret Key Oracle: After receiving a query from ADV_2, the challenger ξ returns USK_{ID} by taking ID as input.

Signing Oracle: On receiving a query from adversary ADV_2, signing oracle returns a valid signature Ω signed by current public/private key of the user ID, by taking ID, UPK_{ID} with message $m \in \{0,1\}^*$ as input.

- **Forgery Phase:** Finally ADV_2 out puts Ω^*_{agg} as forgery on message $(m^*_i)_{i=1\text{to }n}$, under the identities $(ID^*_i)_{i=1\text{to }n}$ and the corresponding $(UPK^*_{ID_i})_{i=1\text{to }n}$ and wins the game if (i) Ω^*_{agg} is a valid signature.

  (ii) Secret Key Oracle has never involved in this game for at least one of the $(ID^*_i)_{i=1\text{to }n}$, say (ID^*_1).

  (iii) Sign Oracle has never involved in this game for (ID^*_1, m^*_1).

Definition 1: A CLAS scheme is said to be existentially unforgeable under adaptive chosen message attack, if there exists no polynomial time adversary (Type-I and Type-II) with non-negligible advantage in the above games I and II respectively.

4. Proposed CLAS Scheme

In this section first we propose our efficient CLAS scheme and then we prove its security.

4.1 CLAS Scheme

As discussed in section 3.1, the proposed CLAS scheme consists of the following algorithms.

- **Master Key Gen:** KGC run this algorithm by taking security parameter $l \in Z^+$ as input and performs the following.

1. Choose additive and multiplicative cyclic groups as G_{Adt} and G_{Mlt} of same prime order q with a bilinear pairing $e: G_{Adt} \times G_{Adt} \to G_{Mlt}$; and $P \in G_{Adt}$ as a generator of G_{Adt}.
2. Select a random $s \in Z_q^*$ as the master secret key and sets master public key as $Q_{Pub} = sP$.
3. Choose five cryptographic hash functions $H_1 : \{0,1\}^* \to G_{Adt}$, $H_2, H_5 : \{0,1\}^* \to Z_q^*$ and $H_3, H_4 : \{0,1\}^t \to G_{Adt}$. KGC publishes the system parameters as $\tau = \{q, G_{Adt}, G_{Mlt}, e, P, Q_{Pub}, H_1, H_2, H_3, H_4, H_5\}$ and keeps s secretly.

- **Partial Key Gen:** KGC runs this algorithm by taking *ID* as input. KGC computes $K_{ID} = H_1(ID)$ and $UPSK_{ID} = sK_{ID}$ and sends $UPSK_{ID}$ to *ID* via secure channel.
- **User Key Gen:** User runs this algorithm by choosing $x_{ID} \in Z_q^*$ randomly and sets $USK_{ID} = x_{ID}$ and $UPK_{ID} = x_{ID}P$.
- **Signature Generation:** Signer runs this algorithm by taking $\tau, ID, UPK_{ID}, USK_{ID}, UPSK_{ID}$, message $m \in \{0,1\}^*$ as input and generates the signature Ω on a message $m \in \{0,1\}^*$ by performing the following.

1. The signer first chooses the state of information Δ and ∇. (Here we take some elements of the system parameters as Δ and ∇.)
2. The signer chooses $r \in Z_q^*$ and computes $R = rP$, $S = H_3(\Delta)$, $W = H_4(\nabla)$, $h_2 = H_2(m, ID, R, UPK_{ID})$ and $h_5 = H_5(\Delta, \nabla, m, ID, R, UPK_{ID}, S, W)$ where Δ and ∇ are arbitrary strings of length t.
3. The signer computes $T = h_2 UPSK_{ID} + S(h_5 x_{ID} + r) + h_2 x_{ID} W$.

Now $\Omega = (R, T)$ is a signature on a message *m*.

- **Aggregate:** An aggregate signature generator will run this algorithm for a set of *n* individual users who uses the same state of information Δ, ∇, by taking signatures $(\Omega_i)_{i=1 \text{to } n}$ from *n* different users $(\mathcal{U}_i)_{i=1 \text{to } n}$ with identities $(ID_i)_{i=1 \text{to } n}$ and corresponding public keys $(UPK_{ID_i})_{i=1 \text{to } n}$ on messages $(m_i)_{i=1 \text{to } n}$. The aggregate signature generator computes the aggregate signature $\Omega_{agg} = (R_1, R_2, R_3, ...R_n, T)$ by finding $T = \sum_{i=1}^{n} T_i$, for messages $(m_i)_{i=1 \text{to } n}$.

- **Aggregate Verify:** To verify an aggregate signature $\Omega_{agg} = (R_1, R_2, R_3, ...R_n, T)$ signed by aggregate set of users $(\mathcal{U}_i)_{i=1 \text{to } n}$ with identities $(ID_i)_{i=1 \text{to } n}$ and corresponding $(UPK_{ID_i})_{i=1 \text{to } n}$, on messages $(m_i)_{i=1 \text{to } n}$, with the same state of information Δ, ∇, the verifier performs the following.

Compute $K_{ID_i} = H_1(ID_i)$ and $S = H_3(\Delta)$, $W = H_4(\nabla)$ and

$h_{2i} = H_2(m_i, ID_i, UPK_{ID_i}, R_i),$

$h_{5i} = H_5(\Delta, \nabla, m_i, ID_i, UPK_{ID_i}, R_i, S, W)$ for $i = 1,2,3...n$

Verify whether

$$e(T,P) = e(\sum_{i=1}^{n} h_{2i} K_{ID_i}, Q_{Pub}) e(\sum_{i=1}^{n} (h_{5i} UPK_{ID_i} + R_i), S)$$

$$\times e(\sum_{i=1}^{n} h_{2i} UPK_{ID_i}, W)$$

holds or not. If it holds, accept the signature.

Remark: In an aggregating set, all the users must use the same (unique) state of information Δ and ∇ (according to the practical condition, Δ and ∇ can be empty strings) when signing. For such a Δ and ∇, one can choose the current time, some parts of the system parameters or other feasible information.

Proof of correctness of the proposed scheme:

The correctness of the scheme can be verified as follows.

$$e(T,P) = e(\sum_{i=1}^{n} T_i, P) = e(T_1, P).....e(T_n, P)$$

$$= e(h_{21} K_{ID_1}, Q_{Pub}) e(h_{51} UPK_{ID_1} + R_1, S) e(h_{21} UPK_{ID_1}, W).....$$
$$e(h_{2n} K_{ID_n}, Q_{Pub})\ e(h_{5n} UPK_{ID_n} + R_n, S) e(h_{2n} UPK_{ID_n}, W)$$

$$= e(\sum_{i=1}^{n} h_{2i} K_{ID_i}, Q_{Pub}) e(\sum_{i=1}^{n} (h_{5i} UPK_{ID_i} + R_i), S)$$

$$\times e(\sum_{i=1}^{n} h_{2i} UPK_{ID_i}, W).$$

4.2 Security of our CLAS Scheme

In the following, we prove the security of our CLAS scheme against Type I and Type II adversaries.

Theorem 1: *The proposed CLAS scheme is existentially unforgeable against adaptive chosen message attacks in the Random Oracle Model with the assumption that the CDH problem is hard.*

We prove this theorem with the help of the following lemma 1 and lemma 2.

Lemma 1: *In the random oracle model, if there exists a Type-I adversary* ADV_1 *who has an advantage* ε *in forging a valid aggregate signature of our CLAS scheme in an attack modelled by Game-I within a time span* t *for a security parameter* l*, after making at most* q_{H_i} *queries to random oracles* H_i *for i-1,2,3,4,5,* q_{Cuser} *queries to the* ***Create User*** *request oracle,* q_{Rpsk} *queries to the* ***Reveal Partial Secret Key*** *extraction oracle,* q_{Rsk} *queries to the* ***Reveal Secret Key*** *extraction oracle and* q_{Sign} *queries to the* ***Sign*** *oracle, then the CDH problem in* G_{Adt} *can be solved with in time*

$t+(q_{H_1}+q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5}+q_{Cuser}+q_{Rpsk}+q_{Rsk}+q_{Sign})T_E$ *and with probability* $\varepsilon' \geq \frac{1}{(q_{Rpsk}+n)\mathrm{e}}\varepsilon$, *where* T_E is the time required to compute the elliptic curve point multiplication in G_{Adt}, e *is base of natural logarithm and* n *is the size of the aggregating set.*

Proof: Let ξ be a CDHP challenger. Let $\mathrm{ADV_I}$ is a Type-I adversary who can forge a valid aggregate signature on a message by interacting with ξ by following Game-I. We show that using $\mathrm{ADV_I}$, ξ can solve the CDH problem. Challenger ξ is given $(A=uP, B=vP)$ as a random instance of the CDH problem in G_{Adt}.

- **Initialization Phase:** Algorithm ξ sets $Q_{Pub}=A=uP$ and runs **Master Key Gen** to generate τ. ξ then gives τ and master public key to $\mathrm{ADV_I}$ and keeps s secretly.
- **Queries Phase:** In this phase, $\mathrm{ADV_I}$ performs the oracle simulation and ξ responds to these oracles as follows.

Queries on oracle $H_1\left(H_1(ID_i)\right)$ *:* ξ maintains a list $\mathcal{L}_1$, which is initially empty. It contains the tuples of the form $(ID_i, l_{1i}, K_{ID_i}, c_i)$. After receiving a query on $H_1(ID_i)$, if there is a tuple $(ID_i, l_{1i}, K_{ID_i}, c_i)$ on $\mathcal{L}_1$, ξ returns K_{ID_i}. Otherwise, ξ first picks a random $l_{1i} \in Z_q^*$, then flips a coin $c_i \in \{0,1\}$ that yields 0 with probability δ and 1 with probability $(1-\delta)$. (δ will be determined later.) If $c_i=0$, ξ sets $K_{ID_i}=l_{1i}B=l_{1i}vP$, adds $(ID_i, \perp, K_{ID_i}, c_i)$ to $\mathcal{L}_1$, and returns K_{ID_i} as answer. Otherwise, sets $K_{ID_i}=l_{1i}P$, adds $(ID_i, l_{1i}, K_{ID_i}, c_i)$ to $\mathcal{L}_1$, and returns K_{ID_i} as answer.

Queries on oracle $H_2\left(H_2(m_i, ID_i, UPK_{ID_i}, R_i)\right)$*:* ξ maintains a list $\mathcal{L}_2$, which is initially empty. It contains the tuples of the form $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$. After receiving H_2 query on $(m_i, ID_i, UPK_{ID_i}, R_i)$, if a tuple $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ exists on $\mathcal{L}_2$, ξ returns l_{2i}. otherwise, ξ picks a random $l_{2i} \in Z_q^*$ and returns l_{2i}. ξ adds $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ to $\mathcal{L}_2$.

Queries on oracle $H_3\left(H_3(\Delta_i)\right)$**:** ξ maintains a list $\mathcal{L}_3$, which is initially empty. It contains the tuples of the form (Δ_i, S_i, l_{3i}). After receiving a query on $H_3(\Delta_i)$, ξ gives the same answer from $\mathcal{L}_3$, if the query has been made earlier. Otherwise, ξ picks a random $l_{3i} \in Z_q^*$, computes $S_i=l_{3i}P$ and returns S_i. ξ adds (Δ_i, S_i, l_{3i}) to $\mathcal{L}_3$.

Queries on oracle $H_4\left(H_4(\nabla_i)\right)$**:** ξ maintains a list $\mathcal{L}_4$, which is initially empty. It contains the tuples of the form (∇_i, W_i, l_{4i}). After receiving a query on $H_4(\nabla_i)$, ξ gives the same answer from $\mathcal{L}_4$, if the query has been made earlier. Otherwise, ξ picks a random $l_{4i} \in Z_q^*$, computes $W_i=l_{4i}P$ and returns W_i. ξ adds (∇_i, W_i, l_{4i}) to $\mathcal{L}_4$.

Queries on oracle $H_5\left(H_5(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i)\right)$*:* ξ maintains a list $\mathcal{L}_5$, which is initially empty. It contains the tuples of the form $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i, l_{5i})$. After receiving H_5 query on $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i)$, if a tuple $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i, l_{5i})$ exists on $\mathcal{L}_5$, ξ returns l_{5i}. otherwise, ξ picks a random $l_{5i} \in Z_q^*$ and returns l_{5i}. ξ adds $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i, l_{5i})$ to $\mathcal{L}_5$.

Reveal Partial Secret Key Oracle $(PSK(ID_i))$**:** ξ maintains a list $\mathcal{L}_{PSK}$, which is initially empty. It contains the tuples of the form $(ID_i, UPSK_{ID_i})$. After receiving a query on $PSK(ID_i)$, ξ gives $UPSK_{ID_i}$ if the request has been made earlier. Otherwise, ξ recovers the corresponding $(ID_i, l_{1i}, K_{ID_i}, c_i)$ from the list $\mathcal{L}_1$ and does as follows.

i) If $c_i = 0$, aborts.

ii) Else, it sets $UPSK_{ID_i} = l_{1i}Q_{Pub} = l_{1i}A$ and returns $UPSK_{ID_i}$ to ADV_1 and adds $(ID_i, UPSK_{ID_i})$ to $\mathcal{L}_{PSK}$.

Create User Oracle $(Cuser(ID_i))$**:** ξ maintains a list $\mathcal{L}_{Cuser}$, which is initially empty. It contains the tuples of the form $(ID_i, UPK_{ID_i}, USK_{ID_i})$. After receiving a query on $Cuser(ID_i)$, the current UPK_{ID_i} from the list $\mathcal{L}_{Cuser}$ will be given if the request has been made earlier. Otherwise, ξ will choose a random $w_i \in Z_q^*$ and sets $UPK_{ID_i} = w_i P$ and $USK_{ID_i} = w_i$. ξ gives UPK_{ID_i} and adds $(ID_i, UPK_{ID_i}, USK_{ID_i})$ to $\mathcal{L}_{Cuser}$.

Reveal Secret Key Oracle $(RSK(ID_i))$**:** When ADV_1 makes this query on $RSK(ID_i)$, if $c_i = 0$, ξ aborts. Otherwise, ξ finds the tuple $(ID_i, UPK_{ID_i}, USK_{ID_i})$ in a list $\mathcal{L}_{Cuser}$, and returns USK_{ID_i} to ADV_1. If there is no tuple in $\mathcal{L}_{Cuser}$, ξ makes a query on $Cuser(ID_i)$ to generate $UPK_{ID_i} = w_i P$, $USK_{ID_i} = w_i$. ξ saves these values in $\mathcal{L}_{Cuser}$, and returns $USK_{ID_i} = w_i$.

Replace Public Key Oracle $(RPK(ID_i))$**:** After receiving a query on $RPK(ID_i)$, ξ finds $(ID_i, UPK_{ID_i}, USK_{ID_i})$ in $\mathcal{L}_{Cuser}$. ξ replaces $UPK_{ID_i} = UPK'_{ID_i}$ and $USK_{ID_i} = \perp$.

Signing Oracle: When ADV_1 makes this query on $(ID_i, m_i, \Delta_i, \nabla_i)$, ξ first makes queries on oracles H_1, H_2 and recovers $(ID_i, l_{1i}, K_{ID_i}, c_i), (m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ from $\mathcal{L}_1, \mathcal{L}_2$ lists respectively and then generates signature as follows.

i) If $c_i = 0$, ξ chooses $r_i, l_{3i}, l_{5i} \in Z_q^*$, sets $S_i = l_{3i}l_{4i}Q_{Pub}$, $h_{5i} = l_{5i}$, $R_i = l_{3i}^{-1}(r_i P - l_{4i}^{-1}l_{2i}K_{ID_i}) - l_{5i}UPK_{ID_i}$ and adds (Δ_i, S_i, l_{3i}), $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i, l_{5i})$ to $\mathcal{L}_3, \mathcal{L}_5$ lists respectively. If any of these tuples exists in their respective $\mathcal{L}_3, \mathcal{L}_5$ lists, then ξ chooses another set of $r_i, l_{3i}, l_{5i} \in Z_q^*$ and tries again. Here ξ recovers (∇_i, W_i, l_{4i}) from $\mathcal{L}_4$ list where $W_i = l_{4i}P$. Now ξ computes $T_i = l_{4i}(r_i Q_{Pub} + l_{2i}UPK_{ID_i})$. Finally ξ responds to ADV_1 with $\Omega_i = (R_i, T_i)$. (R_i, T_i) is a valid signature on message m_i.

ii) If $c_i = 1$, ξ choose $r_i \in Z_q^*$ and computes $R_i = r_i P$, and $T_i = l_{2i}l_{1i}Q_{Pub} + (l_{5i}l_{3i} + l_{2i}l_{4i})UPK_{ID_i} + r_i S_i$. Here ξ recovers l_{3i}, S_i from $\mathcal{L}_3$ list, l_{4i} from $\mathcal{L}_4$ list and l_{5i} from $\mathcal{L}_5$ list. Finally ξ responds to ADV_1 with $\Omega_i = (R_i, T_i)$. (R_i, T_i) is a valid signature on message m_i.

Forgery: After forging a valid aggregate signature $\Omega^*_{agg} = (R_1^*, R_2^*, R_3^*, ...R_n^*, T^*)$ on messages $(m_i^*)_{i=1\text{to }n}$ under $(ID_i^*)_{i=1\text{to }n}$ and the corresponding $(UPK^*_{ID_i})_{i=1\text{to }n}$ of n users $(\mathcal{U}_i)_{i=1\text{to }n}$ with a state of information Δ^* and ∇^* by ADV_1, ξ outputs the value of uvP. It is required that there exists $I \in \{1, 2, 3, ...n\}$ such that ADV_1 has not asked the Partial Secret Key queries for ID_I^* and ADV_1 has not asked the Sign oracle query. Without loss of generality, we let I=1. In addition, the forged signature must satisfy

$$e(T^*,P)=e(\sum_{i=1}^{n}h_{2i}^*K_{ID_i}^*,Q_{Pub})e(\sum_{i=1}^{n}(h_{5i}^*UPK_{ID_i}^*+R_i^*),S^*)$$

$$\times e(\sum_{i=1}^{n}h_{2i}^*UPK_{ID_i}^*,W^*) \qquad (1)$$

where $K_{ID_i}^*=H_1(ID_i^*)$, $S^*=H_3(\Delta^*)$, $W^*=H_4(\nabla^*)$. ξ recovers the tuples $(ID_i^*l_{1i}^*,K_{ID_i}^*,c_i^*)$ from $\mathcal{L}_1$, $(m_i^*,ID_i^*,UPK_{ID_i}^*,R_i^*,l_{2i}^*)$ from $\mathcal{L}_2$, (Δ^*,S^*,l_{3i}^*) from $\mathcal{L}_3$, (∇^*,W^*,l_{4i}^*) from $\mathcal{L}_4$ and $(\Delta^*,\nabla^*,m_i^*,ID_i^*,R_i^*,UPK_{ID_i}^*,S^*,W^*,l_{5i}^*)$ from $\mathcal{L}_5$ lists for all $i,1\le i\le n$. ξ now proceeds only if $c_1^*=0$ and $c_i^*=1$ for all $2\le i\le n$. Otherwise, ξ aborts. Since the forged CLAS must satisfies equation (1), we have

$$\Rightarrow e(h_{21}^*K_{ID1}^*,Q_{Pub})=e(T^*,P)\left(e(\sum_{i=2}^{n}h_{2i}^*K_{ID_i}^*,Q_{Pub})\right)^{-1}$$

$$\times\left(e(\sum_{i=1}^{n}(h_{5i}^*UPK_{ID_i}^*+R_i^*),S^*)e(\sum_{i=1}^{n}h_{2i}^*UPK_{ID_i}^*,W^*)\right)^{-1}$$

By our setting $K_{ID_1}^*=l_{1i}^*vP, S^*=l_{3i}^*P, W^*=l_{4i}^*P,$ $R_i^*=r_i^*P,$ $UPK_{ID_i}^*=w_i^*P$ and $K_{ID_i}^*=l_{1i}^*P,$ ξ computes

$$\Rightarrow uvP=$$

$$\left\{T^*-\left(\begin{array}{l}\sum_{i=2}^{n}(h_{2i}^*l_{1i}^*)Q_{Pub}+\sum_{i=1}^{n}(h_{5i}^*UPK_{ID_i}^*+R_i^*)l_{3i}^*\\+\sum_{i=1}^{n}(h_{2i}^*UPK_{ID_i}^*)l_{4i}^*\end{array}\right)\right\}\times\left(h_{21}^*l_{1i}^*\right)^{-1}.$$

Finally ξ's success probability in solving the CDH problem is at least $\frac{1}{q_{Rpsk}+n}\left(1-\frac{1}{q_{Rpsk}+n}\right)^{(q_{Rpsk}+n-1)}\varepsilon$, and for large q_{Rpsk}, this probability turns to $\frac{1}{(q_{Rpsk}+n)\mathrm{e}}\varepsilon$. Hence, Given an instance $(P,A=uP,B=vP)$, ξ can solve the CDHP with non negligible probability $\frac{1}{(q_{Rpsk}+n)\mathrm{e}}\varepsilon$, which is a contradiction with CDH assumption. □

Lemma 2: *In the random oracle model, if there exists a Type-II adversary* ADV_2 *who has an advantage* ε *in forging a valid aggregate signature of our CLAS scheme in an attack modelled by Game-II within a time span* t *for a security parameter* l*, after making at most* q_{H_i} *queries to random oracles* H_i *for i-2,3,4,5,* q_{Cuser} *queries to the* ***Create User*** *request oracle,* q_{Rsk} *queries to the* ***Reveal Secret Key*** *extraction oracle and* q_{Sign} *queries to the* ***Sign*** *oracle, then the CDH problem in* G_{Adt} *can be solved with in time* $t+(q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5}+q_{Cuser}+q_{Rsk}+q_{Sign})T_E$ *and with probability* $\varepsilon'\ge\frac{1}{(q_{Rpsk}+n)\mathrm{e}}\varepsilon$, *where* T_E is the time required to compute the elliptic curve point multiplication in G_{Adt}, e *is base of natural logarithm and* n *is the size of the aggregating set.*

Proof: Let ξ be a CDHP challenger. Let ADV_2 is a Type-II adversary who can forge a valid aggregate signature on a message by interacting with ξ by following Game-II. We show that using ADV_2, ξ can solve the CDH problem. Challenger ξ is given $(A = uP, B = vP)$ as a random instance of the CDH problem in G_{Adt}.

- **Initialization Phase:** ADV_2 Chooses a random value $s \in Z_q^*$ as master secret key and sets $Q_{Pub} = sP$. ADV_2 runs **Master Key Gen** to generate τ and master public key and then gives s and master public key to the challenger ξ.
- **Queries Phase:** In this phase, ADV_2 performs the oracle simulation and ξ responds to these oracles as follows.

Create User Oracle $\left(Cuser(ID_i)\right)$**:** ξ maintains a list $\mathcal{L}_{Cuser}$, which is initially empty. It contains the tuples of the form $(ID_i, UPK_{ID_i}, USK_{ID_i})$. After receiving a query on $Cuser(ID_i)$, the current UPK_{ID_i} from the list $\mathcal{L}_{Cuser}$ will be given if the request has been made earlier. Otherwise, ξ first choose a random $l_{1i} \in Z_q^*$ then flips a coin $c_i \in \{0,1\}$ that yields 0 with probability δ and 1 with probability $(1-\delta)$. (δ will be determined later.) If $c_i = 0$, ξ sets $UPK_{ID_i} = l_{1i}B = l_{1i}vP$, adds $(ID_i, UPK_{ID_i}, \perp)$ to $\mathcal{L}_{Cuser}$. If $c_i = 1$, ξ sets $UPK_{ID_i} = l_{1i}P$, adds $(ID_i, UPK_{ID_i}, USK_{ID_i})$ to $\mathcal{L}_{Cuser}$. In this case, ξ sets $USK_{ID_i} = l_{1i}$.

Reveal Secret Key Oracle $\left(RSK(ID_i)\right)$***:***

When ADV_2 makes this query on $RSK(ID_i)$, if $c_i = 0$, ξ aborts. Otherwise (if $c_i = 1$), ξ finds the tuple $(ID_i, UPK_{ID_i}, USK_{ID_i})$ in a list $\mathcal{L}_{Cuser}$, and returns USK_{ID_i} to ADV_2. If there is no tuple in $\mathcal{L}_{Cuser}$, ξ makes a query on $Cuser(ID_i)$ to generate $UPK_{ID_i} = l_{1i}P, USK_{ID_i} = l_{1i}$. ξ saves these values in $\mathcal{L}_{Cuser}$, and returns $USK_{ID_i} = l_{1i}$.

Queries on oracle H_2 $\left(H_2(m_i, ID_i, UPK_{ID_i}, R_i)\right)$*:* ξ maintains a list $\mathcal{L}_2$, which is initially empty. It contains the tuples of the form $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$. After receiving H_2 query on $(m_i, ID_i, UPK_{ID_i}, R_i)$, if a tuple $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ exists on $\mathcal{L}_2$, ξ returns l_{2i}.

otherwise, ξ picks a random $l_{2i} \in Z_q^*$ and returns l_{2i}. ξ adds $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ to $\mathcal{L}_2$.

Queries on oracle H_3 $\left(H_3(\Delta_i)\right)$***:*** ξ maintains a list $\mathcal{L}_3$, which is initially empty. It contains the tuples of the form (Δ_i, S_i, l_{3i}). After receiving a query on $H_3(\Delta_i)$, ξ gives the same answer from $\mathcal{L}_3$, if the query has been made earlier. Otherwise, ξ picks a random $l_{3i} \in Z_q^*$, computes $S_i = l_{3i}P$ and returns S_i. ξ adds (Δ_i, S_i, l_{3i}) to $\mathcal{L}_3$.

Queries on oracle H_4 $\left(H_4(\nabla_i)\right)$***:*** ξ maintains a list $\mathcal{L}_4$, which is initially empty. It contains the tuples of the form (∇_i, W_i, l_{4i}). After receiving a query on $H_4(\nabla_i)$, ξ gives the same answer from $\mathcal{L}_4$, if the query has been made earlier. Otherwise, ξ picks a random $l_{4i} \in Z_q^*$, computes $W_i = l_{4i}uP$ and returns W_i. ξ adds (∇_i, W_i, l_{4i}) to $\mathcal{L}_4$.

Queries on oracle H_5 $\left(H_5(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i)\right)$*:* ξ maintains a list $\mathcal{L}_5$, which is initially empty. It contains the tuples of the form $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i, l_{5i})$. After receiving H_5 query on $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i)$, if a tuple $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i, l_{5i})$ exists on $\mathcal{L}_5$, ξ returns l_{5i}. Otherwise, ξ picks a random $l_{5i} \in Z_q^*$ and returns l_{5i}. ξ adds $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i, l_{5i})$ to $\mathcal{L}_5$.

Signing Oracle:

When ADV_2 makes this query on $(ID_i, m_i, \Delta_i, \nabla_i)$, ξ first makes queries on oracles H_2, H_4 and recovers $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ from list $\mathcal{L}_2$, (∇_i, W_i, l_{4i}) from $\mathcal{L}_4$ and then generates signature as follows.

i) If $c_i = 0$, choose $r_i, l_{3i}, l_{5i} \in Z_q^*$, set $S_i = l_{3i}uP$, $R_i = r_iP - (l_{3i}^{-1}l_{4i}l_{2i} + l_{5i})UPK_{ID_i}$ and add the tuple (Δ_i, S_i, l_{3i}) to list $\mathcal{L}_3$, $(\Delta_i, \nabla_i, m_i, ID_i, UPK_{ID_i}, R_i, S_i, W_i, l_{5i})$ to list $\mathcal{L}_5$. If any of these tuples exists in $\mathcal{L}_3, \mathcal{L}_5$ lists respectively, then ξ chooses another set of $r_i, l_{3i}, l_{5i} \in Z_q^*$ and tries again. Now ξ computes $T_i = l_{2i}UPSK_{ID_i} + r_iS_i$. Finally ξ responds to ADV_2 with $\Omega_i = (R_i, T_i)$. (R_i, T_i) is a valid signature on message m_i.

ii) If $c_i = 1$, ξ chooses $r_i \in Z_q^*$, computes $R_i = r_iP$ and

$T_i = l_{2i}UPSK_{ID_i} + l_{3i}(l_{5i}UPK_{ID_i} + R_i) + l_{2i}l_{1i}W_i$. Here ξ recovers l_{3i} from $\mathcal{L}_3$ list, W_i from $\mathcal{L}_4$ list and l_{5i} from $\mathcal{L}_5$ list. Finally ξ responds to ADV_2 with $\Omega_i = (R_i, T_i)$. (R_i, T_i) is a valid signature on message m_i.

- **Forgery:** After forging a valid aggregate signature $\Omega^*_{agg} = (R_1^*, R_2^*, R_3^*, ...R_n^*, T^*)$ on messages $(m_i^*)_{i=1\text{to }n}$ under $(ID_i^*)_{i=1\text{to }n}$ and the corresponding $(UPK^*_{ID_i})_{i=1\text{to }n}$ of n users $(\mathcal{U}_i)_{i=1\text{to }n}$ with a state of information Δ^* and ∇^* by ADV_2, ξ outputs the value of uvP. It is required that there exists $I \in \{1, 2, 3, ...n\}$ such that ADV_2 has not asked the Reveal Secret Key queries for ID_I^* and ADV_2 has not asked the Sign oracle query. Without loss of generality, we let I=1. In addition, the forged signature must satisfy equation (1). Here $K^*_{ID_i} = H_1(ID_i^*)$, $S^* = H_3(\Delta^*)$, $W^* = H_4(\nabla^*)$. ξ recovers the tuples $(ID_i^*, l_{1i}^*, K^*_{ID_i}, c_i^*)$ from $\mathcal{L}_1$, $(m_i^*, ID_i^*, UPK^*_{ID_i}, R_i^*, l_{2i}^*)$ from $\mathcal{L}_2$, $(\Delta^*, S^*, l_{3i}^*)$ from $\mathcal{L}_3$, $(\nabla^*, W^*, l_{4i}^*)$ from $\mathcal{L}_4$ and $(\Delta^*, \nabla^*, m_i^*, ID_i^*, R_i^*, UPK^*_{ID_i}, S^*, W^*, l_{5i}^*)$ from $\mathcal{L}_5$ lists for all $i, 1 \le i \le n$. ξ now proceeds only if $c_1^* = 0$ and $c_i^* = 1$ for all $2 \le i \le n$. Otherwise, ξ aborts. Since the forged CLAS must satisfies equation (1), we have

$$e(h_{21}^*UPK^*_{ID_1}, W^*) = e(T^*, P)$$
$$\times\left(e(\sum_{i=2}^{n} h_{2i}^*UPK^*_{ID_i}, W^*)e(h_{51}^*UPK^*_{ID_1} + R_1^*, S^*)\right)$$
$$\times\left(e(\sum_{i=1}^{n} h_{2i}^*K^*_{ID_i}, Q_{Pub})e(\sum_{i=2}^{n}(h_{5i}^*UPK^*_{ID_i} + R_i^*), S^*)\right)^{-1}$$

By our setting $UPK^*_{ID_1} = l_{1i}^*vP, UPK^*_{ID_i} = l_{1i}^*P, W^* = l_{4i}^*uP$, $R_i^* = r_i^*P$ and $Q_{Pub} = sP$, ξ computes

$$uvP = \left\{\begin{array}{l} T^* - \left(\sum_{i=2}^{n}(h_{2i}^*l_{1i}^*l_{4i}^*A)\right) - l_{31}^*(h_{51}^*l_{11}^*A + R_1^*) - \\ \sum_{i=1}^{n}(h_{2i}^*UPSK^*_{ID_i}) - \sum_{i=2}^{n} l_{3i}^*(h_{5i}^*l_{1i}^*P + R_i^*) \end{array}\right\}\left(h_{21}^*l_{11}^*l_{41}^*\right)^{-1}.$$

Finally ξ's success probability in solving the CDH problem is at least $\frac{1}{q_{Rsk}+n}\left(1-\frac{1}{q_{Rsk}+n}\right)^{(q_{Rsk}+n-1)}\varepsilon$, and for large $q_{Rsk,}$ this probability turns to $\frac{1}{(q_{Rsk}+n)\mathrm{e}}\varepsilon$. Hence, Given an instance $(P, A=uP, B=vP)$, ξ can solve the CDHP with non negligible probability $\frac{1}{(q_{Rsk}+n)\mathrm{e}}\varepsilon$, which is a contradiction with CDH assumption. □

5. Extension of CLAS Scheme to achieve Full aggregation

In this section we give a brief idea about how to extend our CLAS scheme to achieve full aggregation.

5.1 Extension to Full Aggregation

- **Master Key Gen:** KGC runs this algorithm by taking security parameter $l \in Z^+$ as input and performs as follows:
  1. Choose additive and multiplicative cyclic groups G_{Adt} and G_{Mlt} respectively of same prime order q with a bilinear pairing $e: G_{Adt} \times G_{Adt} \to G_{Mlt}$; and $P \in G_{Adt}$ as a generator of G_{Adt}.
  2. Select a random $s \in Z_q^*$ as the master secret key and sets $P_{Pub} = sP$.
  3. Choose four cryptographic hash functions $H_1: \{0,1\}^* \to G_{Adt}$, $H_2, H_5: \{0,1\}^* \to Z_q^*$ and $H_3, H_4: \{0,1\}^t \to G_{Adt}$. KGC publishes the system parameters as $\tau = \{q, G_{Adt}, G_{Mlt}, e, P, P_{Pub}, H_1, H_2, H_3, H_4, H_5\}$ and keeps s secretly.
- **Partial Key Gen:** Let $(\mathcal{U}_i)_{i=1\text{to }n}$ denote all the users who join in signing. The identity of $\mathcal{U}_i$ is denoted as ID_i. KGC runs this algorithm by taking ID_i as input. KGC computes $K_{ID_i} = H_1(ID_i)$ and $UPSK_{ID_i} = sK_{ID_i}$ and sends $UPSK_{ID_i}$ to ID_i via secure channel.
- **User Key Gen:** User runs this algorithm by choosing $x_{ID_i} \in Z_q^*$ randomly and sets $USK_{ID_i} = x_{ID_i}$ and $UPK_{ID_i} = x_{ID_i}P$.
- **Individual Signature Generation:** Given n different messages $(m_i)_{i=1\text{to }n}$, without loss of generality, we assume that $\mathcal{U}_i$ signs message m_i. Signer runs this algorithm by

taking $\tau, ID_i, UPK_{ID_i}, USK_{ID_i}, UPSK_{ID_i}$, message $m_i \in \{0,1\}^*$ as input and generates the signature Ω_i on a message $m_i \in \{0,1\}^*$ by performing the following.

1. The signer first chooses the one-time-use state of information $\Delta \& \nabla$. (Here we take some system parameters)
2. The signer chooses $r_i \in Z_q^*$ and computes and broadcasts $R_i = r_iP$. Let $R = \sum_{i=1}^{n} r_iP$, compute $S = H_3(\Delta)$, $W = H_4(\nabla)$, $h_{2i} = H_2(m_i, ID_i, R, UPK_{ID_i})$ and $h_{5i} = H_5(\Delta, \nabla, m_i, ID_i, R, UPK_{ID_i}, S, W)$ where Δ and ∇ is an arbitrary string of length t.
3. The signer computes $T_i = h_{2i}UPSK_{ID_i} + S(h_{5i}x_{ID_i} + r_i) + h_{2i}x_{ID_i}W$.

Now $\Omega_i = (R_i, T_i)$ is a signature on a message m_i.

- **Aggregate Signature Generation:** Any one (one of the signers) can be designated to aggregate all these individual/single signatures that use the same string Δ and ∇. The designated player (DP) first verifies the validity of each single signature

$\Omega_i (1 \le i \le n)$ that use the same string Δ and ∇. We assume that the single signatures of same string Δ and ∇ are all valid. DP will run this algorithm for an aggregate set of n individual users, by taking signatures $(\Omega_i)_{i=1 \text{to } n}$ from n different users $(\mathcal{U}_i)_{i=1 \text{to } n}$ with identities $(ID_i)_{i=1 \text{to } n}$ and corresponding public keys $(UPK_{ID_i})_{i=1 \text{to } n}$ on messages $(m_i)_{i=1 \text{to } n}$ and computes the aggregate signature $\Omega_{agg} = (R, T)$ by finding $T = \sum_{i=1}^{n} T_i$, for messages $(m_i)_{i=1 \text{to } n}$.

- **Aggregate Signature Verification:** To verify an aggregate signature $\Omega_{agg} = (R, T)$ signed by aggregate set of users $(\mathcal{U}_i)_{i=1 \text{to } n}$ with identities $(ID_i)_{i=1 \text{to } n}$ and corresponding $(UPK_{ID_i})_{i=1 \text{to } n}$, on messages $(m_i)_{i=1 \text{to } n}$, the verifier performs the following.

  Compute $K_{ID_i} = H_1(ID_i)$, $S = H_3(\Delta)$, $W = H_4(\nabla)$, and

  $h_{2i} = H_2(m_i, ID_i, UPK_{ID_i}, R)$,

  $h_{5i} = H_5(\Delta, \nabla, m_i, ID_i, R, UPK_{ID_i}, S, W)$ for $i = 1, 2, 3...n$.

  Verify whether the following equation holds or not.

$$e(T, P) = e(\sum_{i=1}^{n} h_{2i} K_{ID_i}, P_{Pub}) e(\sum_{i=1}^{n} (h_{5i} UPK_{ID_i}) + R, S)$$

$$\times e(\sum_{i=1}^{n} h_{2i} UPK_{ID_i}, W).$$ If it is holds, accept the signature.

5.2 Security Analysis

As discussed in section 4.2, we can prove the security of the extended CLAS scheme against Type I and Type II adversaries.

6 Efficiency Analysis

In this section we present the performance analysis of our CLAS scheme. We compare our scheme with other related schemes. We consider the experimental results [37],[38],[39],[40], to achieve the comparable security with1024-bit RSA key, where the bilinear pairing (Tate pairing) is defined over the super singular elliptic curve $E / F_p : y^2 = x^3 + x$ with embedding degree 2 and the 160-bit Solinas prime number $q = 2^{159} + 2^{17} + 1$ with 512-bit prime number p satisfying $p + 1 = 12qr$. The running time is calculated for different cryptographic operations in [37],[38],[40] using MIRACL [39], a standard cryptographic library and implemented on a hardware platform PIV (Pentium-4) 3GHZ processor with 512-MB memory and a windows XP operating system. From the results [37],[38],[39],[40], various cryptographic operations and their conversions are presented in Table 2. Also, we compared our proposed scheme with all the CLAS schemes presented in the literature [7-23] in terms of computational cost, verification cost, signature length and security point of view. We also compared the scheme in terms of partial aggregation or full aggregation. The detailed comparison is presented in Table 3.

Table 2: Notations of various cryptographic operations and their conversions

Notations	Descriptions
T_M	Time required to compute modular multiplication operation
T_E	Time required to compute the elliptic curve point multiplication G_{Adt} : $T_E = 29T_M$
T_P	Time required to compute the bilinear pairing in G_{Mlt} : $T_P = 87T_M$
T_X	Time required to compute the pairing–based exponentiation in G_{Mlt} : $T_X = 43.5T_M$
T_I	Time required to compute modular inversion operation in Z_q^* : $T_I = 11.6T_M$
T_H	Time required to compute a map to point hash function : $1T_H = 1T_E = 29T_M$
T_A	Time required to compute the elliptic curve point addition in G_{Adt} : $T_A = 0.12T_M$

From Table 3, it is clear that most of the signature schemes [8],[12],[[21],[16],[17],[20],[23], [31] were proven insecure. Though the remaining schemes [7],[9],[10],[11],[13],[14], [15], [18],[19],[22] are secure, but majority of the schemes [7],[9],[10],[11],[15] are not achieving tight security due to use of forking lemma [36] in their proof of security. Hence, the schemes presented in [14],[18],[19] are the only schemes proven secure without using forking lemma [36] for tightness in security. Also our scheme is secure against coalition of a signer with malicious KGC attack presented in [18].

Table 3: Comparison of the proposed CLAS scheme with the related scheme

Scheme Ref. No[ ]		Signing Cost (CLAS)	Aggregate Verification Cost(CLAS)	Sign. Length CLAS	Const. Pairing	Secure	Without Forking Lemma	Partial/ Full Agg.
[11]	i	$2T_E+1T_A+1T_H$	$(2n+1)T_P+2nT_A$	$2\lvert G_{Adt}\rvert$	No	Yes	No	Partial
	ii	$3T_E+2T_A+2T_H$	$(n+2)T_P+nT_E+nT_A$	$2\lvert G_{Adt}\rvert$	No	Yes	No	Full
[23]		$3T_E+2T_A$	$3T_P+2nT_E+3nT_A$	$2\lvert G_{Adt}\rvert$	Yes	No [26,27]	No	Partial
[17]		$3T_E+2T_A$	$3T_P+2nT_E$	$(n+1)\lvert G_{Adt}\rvert$	Yes	No [8,9,23]	No	Partial
[7]		$2T_E+1T_A+1T_H$	$(n+2)T_P+nT_E+nT_A+nT_H$	$(n+1)\lvert G_{Adt}\rvert$	No	Yes	No	Partial
[9]		$4T_E+3T_A+1T_H$	$3T_P+2nT_E+2nT_A+1T_H$	$(n+1)\lvert G_{Adt}\rvert$	Yes	Yes	No	Partial
[10]		$4T_E+3T_A+1T_H$	$3T_P+3nT_E+2nT_A$	$(n+1)\lvert G_{Adt}\rvert$	Yes	Yes	No	Partial
[15]		$4T_E+2T_A$	$3T_P+3nT_E+2nT_A$	$(n+1)\lvert G_{Adt}\rvert$	Yes	Yes	No	Partial
[12]		$2T_E+1T_A$	$3T_P+nT_E+2nT_A$	$(n+1)\lvert G_{Adt}\rvert$	Yes	No [30]	No	Partial
[20]		$3T_E+2T_A+2T_H$	$(n+3)T_P+2nT_A+(n+1)T_H$	$(n+1)\lvert G_{Adt}\rvert$	No	No [14]	Yes	Partial
[19]		$5T_E+4T_A+3T_H$	$5T_P+2nT_E+(4n+1)T_A+3T_H$	$2\lvert G_{Adt}\rvert$	Yes	Yes	Yes	Full
[21]		$3T_E+2T_A$	$3T_P+2nT_E+2nT_A$	$(n+1)\lvert G_{Adt}\rvert$	Yes	No [41]	Yes	Partial
[18]		$2T_E+1T_A+2T_H$	$2nT_P+2nT_E+nT_A+2nT_H$	$\lvert G_{Adt}\rvert$	No	Yes	Yes	Partial
[22]		$3T_E+2T_A+2T_H$	$4T_P+2nT_E+2nT_A+1T_H$	$(n+1)\lvert G_{Adt}\rvert$	Yes	Yes	No	Partial
[8]		$4T_E+3T_A+2T_H$	$4T_P+2nT_E+2nT_A+2T_H$	$2\lvert G_{Adt}\rvert$	Yes	No [24,25]	Yes	Full
[16]		$3T_E+2T_A+2T_H$	$(n+3)T_P+nT_A+(n+1)T_H$	$2\lvert G_{Adt}\rvert$	No	No [28]	Yes	Full
[28]		$4T_E+2T_A+2T_H$	$(n+3)T_P+2nT_A+nT_E+(n+1)T_H$	$(n+1)\lvert G_{Adt}\rvert$	No	Yes	Yes	Partial
[13]		$4T_E+3T_A+2T_H$	$(n+3)T_P+2nT_E+(3n+1)T_A+nT_H$	$\lvert G_{Adt}\rvert$	No	Yes	Yes	Full
[31]		$3T_E+2T_A+2T_H$	$4T_P+2nT_E+(3n-2)T_A+2T_H$	$2\lvert G_{Adt}\rvert$	Yes	No [32]	Yes	Full
Our Scheme	i	$4T_E+2T_A+2T_H$	$4T_P+3nT_E+(4n-3)T_A+2T_H$	$(n+1)\lvert G_{Adt}\rvert$	Yes	Yes	Yes	Partial
	ii	$4T_E+2T_A+2T_H$	$4T_P+3nT_E+(3n-2)T_A+2T_H$	$2\lvert G_{Adt}\rvert$	Yes	Yes	Yes	Full

Coming to computational efficiency, the schemes with constant pairing operations are more efficient than the schemes with more number of pairing operations in verification process. From Table 3, the schemes presented in [7],[11],[13],[14],[16],[18],[20] are not having constant pairings in aggregation verification, which leads to high computation cost. Hence the schemes presented in [8],[9],[10],[12],[21],[15],[22],[17],[19] are computationally efficient.

The other metric to compare the efficiency is signature length. From Table 3, it is clear that most of the signature schemes [7],[9],[10],[12],[14],[21],[15],[22],[17],[20] are of signature length $(n+1)|G_{Adt}|$ and [8],[11],[13],[16], [18],[19] are the only schemes that have signature length $2|G_{Adt}|$ or $|G_{Adt}|$. In literature, most of the CLAS schemes [7],[9], [10],[11(i)],[12],[14],[15],[17],[20],[22], provide only partial aggregation, which increases the signature length unanimously. There are only four signature schemes [8], [11(ii)], [16], [19] which achieve full aggregation and reduce communication cost. We have extended our scheme to achieve full aggregation by aggregating the randomness part in the signature generation itself. Comparing our full aggregation scheme with these schemes, our scheme is much efficient and secure than these [8], [11(ii)], [16], [19] schemes.

Now comparing our schemes with all other schemes in terms of security, computational efficiency, signature length and about partial aggregation or full aggregation, there are only two schemes [22],[19] to compare. Comparing our scheme with [19] scheme, it is clear that our proposed CLAS scheme is much efficient than [19] as it requires 5 paring operations in verification process. Comparing our scheme with [22] scheme, though the computational cost is almost same as our proposed scheme but this scheme [22] uses forking lemma for proving its security. Also the scheme [22] did not mention about full aggregation. Thus the proposed CLAS scheme is efficient and achieves tight security.

7. Conclusions

In this paper, we have presented a novel and efficient CLAS scheme and extended this to achieve full aggregation using bilinear pairings over elliptic curves. The proposed scheme is unforgeable against chosen message and identity attack in random oracle model under the hardness of CDH problem and does not use Forking lemma to achieve tight security. The efficiency analysis shows that our CLAS scheme is computationally more efficient and secure than the well-known existing schemes. Our CLAS scheme requires a constant number of pairing operations for aggregation verification. Thus, the proposed CLAS scheme can be applied in the environments where low bandwidth, less storage and low computability are of great concern.

Acknowledgement

This work is supported by WOS-A, DST, Govt. of India under the grant No. SR/WOS-A/PM-1033/2014(G), WOS-A, DST.

References

[1] Diffe, W., Hellman, M.E.: 'New directions in cryptography', IEEE Transactions in Information Theory, 1976, 22, pp. 644-654.

[2] Shamir, A.: 'Identity-based Cryptosystems and Signature Schemes', LNCS, 1984, 196, pp.47-53.

[3] Al-Riyami, S. S., Paterson, K.G.: 'Certificateless Public key Cryptography', LNCS, 2003, 2894, pp. 452-473.

[4] Shim, K. A.: 'Security models for certificateless signature schemes revisted', Information Science, 2015, 296, pp. 315-321.

[5] Boneh, D., Gentry, C., Lynn, B., Shacham, H.: 'Aggregate and verifiably encrypted signatures from bilinear maps' Proc. of EUROCRYPT2003, LNCS2656, 2003, pp.416-432.

[6] Castro, R., Dahab, R.: 'Efficient Certificateless Signatures suitable for aggregation', Cryptology eprint archive, Report 2007/454, 2007.

[7] Chen, Y.C., Horng, G. ,Chao-Liang Liu, Yuan-Yu and Chan C. S.: 'Efficient Certificateless Aggregate Signature Scheme', Journal Electronic Science and Technology, 2012, 10, (3), pp. 209-214.

[8] Chen, Y.C., Tso, R., Mambo, M., Huang, K., Horng, G.: 'Certificateless aggregate signature with efficient verification', Security and communication networks,2015, 18, (13), pp. 2232-2243.

[9] Cheng, L., Wen, Q., Jin, Z., Zhang, H., Zhou, L.: 'Cryptanalysis and improvement of a Certificateless aggregate signature scheme', Information Science, 2014, 295, pp. 337-346.

[10] Deng, J., Xu, C., Wu, H., Dong, L.: 'A New Certificateless signature with enhanced security and aggregation version', Concurrency and computation: Practice and Experience, 2016, 28, (4), pp. 1124-1133.

[11] Gong, Z., Long, Y., Hong, X., Chen, K.: 'Practical Certificateless aggregate signatures from bilinear maps', Journal of Information science and Engineering, 2010, 26, pp. 2093-2106.

[12] Horng S. J., Tzeng, S. F., Huang, P. H., Wang, X., Li, T., and Khan, M. K.: 'An Efficient Certificateless Aggregete Signature with Conditional privacy-preserving for Vehicular Sensor Networks, Information Sciences, 2015, 317, pp. 48-66.

[13] Jayaprakash kar, Certificateless Aggregate Short Signature Scheme, e-print,IACR, 2016. Available at https://eprint.iacr.org/2016/305.pdf

[14] Kang B. and Xu D.: 'A Secure Certificateless Aggregate Signature Scheme', International Journal of Security and its Applications, 2016, 10, (3), pp. 55-68.

[15] Malhi A. K. and Shalini B.: 'An Efficient Certificateless Aggregate Signature Scheme for Vehicular Ad-Hoc Networks', Discrete Mathematics and Theoretical Computer Science, DMTCS, 2015, 17, (1), pp. 317–338.

[16] Zho, M., Zhang, M., Wang, C., and Yang, B.: 'CCLAS: A Practical and Compact Certificateless Aggregate Signature with Share Extraction', International Journal of Network Security, 2014,16, (3), pp.174-181.

[17] Xiong, H., Guan, Z. , Chen, Z., Li, F.: 'An efficient certificateless aggregate signature with constant pairing computations', Information Science, 2013, 219, pp. 225-235.

[18] Zhang, F.,Shen, L., Wu, G.: 'Notes on Security of Certificateless aggregate signature schemes', Information Science, 2014, 287, pp. 32-37.

[19] Zhang, L., Qin, B., Wu, Q., Zhang, F.: 'Efficient many-to-one authentication with certificateless aggregate signatures', Computer Networks, 2010, 54, (14), pp. 2482-2491.

[20] Zhang, L., Zhang, F.: 'A new certificateless aggregate signature scheme', Computation and communication, 2009, 32, (6), pp. 1079-1085.

[21] Liu, H.,Wang, S., Liang, M., Chen, Y.: 'New Construction of Efficient Certificateless aggregate signatures', International Journal of Security and its applications, 2014, 8, (1), pp. 411-422.

[22] Tu, H., He, D., Huang, B.: 'Reattack of a Certificateless aggregate signature scheme with constant pairing computations', The scientific world journal, 2014, article ID343715.

[23] He, D., Tian, M., Chen, J.: 'Insecurity of an efficient Certificateless aggregate signature scheme with constant pairing computations', Information Sciences, 2014, 268, pp.458-462.

[24] Zhang, J., Zhao, X., and Mao, J.: 'Attack on Chen et al's certificateless aggregate signature scheme', Security and communication networks, 2016,9, pp.54-59.

[25] Wang, L., Chen, K., Long, Y., and Wang, H.: 'Cryptanalysis of a certificateless aggregate signature scheme', Security and communication networks, 2016, 9, (11), pp.1353-1358.

[26] Fan, A., Wang, Q.: 'Security analysis and improvement of the certificateless aggregate signature schemes', AMSE journals-AMSE IIETA publication, 2017, 60, (1), pp. 174-188.

[27] Li, J., Yuan, H., Zhang, Y.: 'Cryptanalysis and improvement for certificateless aggregate signature', Fundamenta Informaticae, 2018, 157, pp. 111-123.

[28] Chen, C. C., Chien, H., and Horng, G.: 'Cryptanalysis of a Compact Certificateless Aggregate Signature Scheme', International Journal of Network Security, 2016, 18, (4), pp.793-797.

[29] Hou, H., Zhang, X., Dong, X.: 'Improved certificateless aggregate signature scheme', Journal of Shandong University (Natural Science), 2013, 48,(9), pp. 29–34.

[30] Li, J., Hong, Y., and Zhang, Y.: 'Cryptanalysis and improvement of Certificateless Aggregate Signature with Conditional privacy-preserving for Vehicular Sensor Networks', e-prnt, IACR, 2016.

[31] Nie, H., Li, Y., Chen, W.,Ding, Y., NCLAS: a novel and efficient certificateless aggregate signature scheme, Security and Communication Networks, 2016, 9, pp. 3141-3151.

[32] Pakniat, N., Noroozi, M.: 'Cryptanalysis of a certificateless aggregate signature scheme', Proc. of 9th Conference on Command, Control, Communications and Computer Intelligence, 2016.

[33] Kang, B., Wang, M., Jing, D.: 'An efficient certificateless aggregate signature scheme", Wuhan university journal of natural sciences, 2017, 22, (2), pp. 165-170.

[34] Kumar, P., Sharma, V., Sharma, G.: 'Certificateless aggregate signature schemes: A review', Proc. of International Conference on Computing, Communication and Automation (ICCCA 2016), 2016.

[35] Kumar, P., Kumari, S., Sharma, V., Sangaiah, A. K., Wei, J., Li, X.: 'A certificateless aggregate signature scheme for healthcare wireless sensor network', Sustainable computing: Informatics and systems, Article in press.

[36] Pointcheval D., Stern J.: 'Security arguments for digital signatures and blind signatures', Journal of Cryptology, 2000, 13, (3), pp.361-369.

[37] Barreto, P., Kim, H. Y., Lynn, B., Scott, M.: 'Efficient Algorithms for Pairing based Cryptosystems', LNCS 2442, 2002, pp. 354-368.

[38] Cao, X., Kou, W., Du, X.: 'A Pairing –free Identity Based Authenticated Key Agreement Protocol with Minimal Message Exchanges', Information Sciences, 2010, 180, (15), pp. 2895-2903.

[39] MIRACL Library. Available at http://certivox.org/display/EXT/MIRACL.

[40] Tan, S.H., Heng, S.H., Goi, B.M.: 'Java Implementation for Pairing-based Cryptosystems' LNCS, 2010, 6019, pp. 188-198.

[41] Zhang, Y., Wang, C.: 'Comment on New construction of efficient certificateless aggregate signatures', International Journal of Security and its Applications, 2015, 9, (1), pp. 147-154.