Certificateless Aggregate Short Signature Scheme

Jayaprakash Kar

Information Security Research Group
Department of Information Systems
Faculty of Computing & Information Technology
King Abdulaziz University,
Kingdom of Saudi Arabia, Jeddah-21589
jgopabandhu@kau.edu.sa

Abstract. An aggregate signature scheme is the aggregation of multiple signatures into a single compact signature of short string that can convince to any arbitrary verifier participating in the scheme. The aggregate signature scheme is very useful for real-world cryptographic applications such as secure routing, database outsourcing etc where the signatures on several distinct messages generated by many distinct users requires to be compact. In this paper, we presented an aggregate signature scheme using Certificateless Public Key Cryptography(CL-PKC). The scheme is provably secure with strongest security and shortest length. We have proven the scheme is existentially unforgeable under adaptive chosen message attack, assuming the hardness of computational Diffie-Hellman(CDH) Problem.

Keywords: Bilinear map, Aggregation, Digital signature, Certificateless signature

1 Introduction

In conventional public key cryptography (PKC), there exist two keys public and private for each user. Public key is publicly known to everyone, whereas a private key is to be kept secret. In order to combine the user's identity and public key, the conventional PKC involves public key infrastructure (PKI) where a trusted third party known as certificate authorities (CA) issues certificate to combine the two parameters user's identity and the corresponding public key. In this cryptosystem, prior to using the public key of the user, the participant must verify the certificate of the corresponding user. Significantly, this results huge storage space and computing cost for managing the certificates.

In order to simplify the certificate management process of PKI, Shmir [1] introduced identity-based public key cryptography(ID-PKC) in 1984 where the user's public key is taken as his unique identity such as telephone number, IP address or mail address etc. However, in order to generate the private key of the user, a private Key Generator (PKG) is involved in the system. The private key of the user is combined with his own public key and PKG is the owner of master secret. Therefore ID-PKC faces the key-escrow problem. Consequently, to resolve this problem, Al-Riyami and Paterson [3] introduced a novel paradigm known as certificateless public key cryptography (CL-PKC). CL-PKC also involves Key Generation Center (KGC) to construct the private key of the user as in ID-PKC. But the KGC does not allow to access the complete private key of the user. So the user generates his full private key by choosing a secret information and combines with the partial private key constructed by the KGC. The corresponding public key of the user is computed by using the system parameters published by KGC and the user's secret information chosen by himself.

Authentication is a very important security goal for many cryptographic applications. In order to improve the performance of structuring blocks, authentication is very crucial which is achieved by digital signature. For constrained low processor devices such as cell phone, PDA, tablet, RFID chips, sensors time complexity, communication overhead and storage space are very crucial. In all these cases, battery life is frequently more of a restricting bottleneck than the processor speed. The communication overhead of a single bit of data, consumption of power is more than the execution of 32-bit basic arithmetic instructions [2]. Therefore, it is the greatest challenge to the research community to limit the communication requirement constructing a short size of signatures. The solution to serve this purpose, is to develop an aggregate signature. In an aggregate signature, a multiple number of signatures generated by multiple user on multiple documents can be compressed into a single signature.

This is very useful for many real-world cryptographic applications such as to develop a secure routing protocol, construct a combined aggregate certificate, etc. In PKI of depth n, the chain of certificate consists of n signature issued by n different CAs with n public keys. The series of certificates are compressed to an aggregate certificate.

In a secure routing protocol such as Boarder Gateway Protocol where, the segment of paths in the network are being signed by each individual router and the collection of signatures for all the paths are to be forwarded to the next router. This results in raising of communication overhead, which can be reduced by using aggregate signatures. Apart from the compression, aggregate signature can be applied in the dynamic content distribution [4] and database out-souring [5].

1.1 Motivation and Contribution

One of the most important and well-known cryptographic primitive in public key cryptography is digital signature that provides the security goals authenticity, integrity and non-repudiation. In real-word cryptographic application, it is required to achieve these security goals. Further for low processor constrained devices such as cell phones, PDA, smart card, etc, it is desirable to construct an efficient signature scheme of short size with high level of security. It is a challenging task for the research community to develop such signature schemes. In many applications such as wireless sensor network, RFID, etc, the communication overhead or bandwidth is very low, have lower storage space and computability. In such environments, the conventional signature is not suited to implement. It needs to construct a signature that can be implemented in such constraint situations. Therefore, aggregate signature is most suitable to implement on such constraint scenarios. Aggregate signature has two characteristics. These allow to integrate multiple signatures signed by multiple signers into a single signature. That results to reduce the length in size. Further the computational cost for the aggregate signature is less than that the computational cost of individual verifier. Due to these two most important characteristics, it is suitable for many applications.

In this paper, we have presented the formal adversary model and analyzed the security of CL-AS scheme. We investigate the scheme proposed by Choi *et* al. [20] and construct a provably secure CL-ASS scheme in random oracle model. Security of the scheme relies on computational Diffie-Hellman problem over groups with bilinear maps. We have proven, the scheme is secure against existentially unforgeable under adaptive chosen message attack.

1.2 Related Work

The notion of aggregate signature was first introduced by Boneh, Gentry, Lynn and Shacham [21]. Subsequently, Lysyanskaya *et* al. [22] presented a certified trapdoor permutation to release an aggregate signature under the weaker assumption of security model. However, the method requires sequential aggregation. Cheon *et* al.[25] introduced the first Identity-based aggregate signature (IDAS) scheme. Subsequently Cheng *et* al. [8], Xu *et* al. [9] and Gentry and Ramzan [10] proposed improved IDAS scheme. Later on Gentry and Ramzan [10] presented IDAS scheme based on bilinear pairing. The notion of security of certificateless encryption scheme was first introduced by Al-Riyami and Paterson [3] and defines a formal security proof. The adversary model of certificateless signature scheme(CLS) was given by Huang *et* al. [11] and proposed a provable secure CLS. Subsequently a generic construction of CLS was proposed by Yumand Lee [12]. Afterward, Hu *et* al. [13] proved that the scheme is not secure under the adversary model and proposed an improved CLS scheme. Huang *et* al. [14] re-framed the adversary models of certificateless signatures and presented two novel constructions of the scheme. Choi *et* al. [15] proposed two efficient CLS schemes and have proven the security under weak security model [14]. A short CLS scheme has been proposed by Du and Wen. However, the proof of security is not correct. Later on an aggregate signature scheme was proposed in certificateless public key setting [16]. The author claimed, their proposed scheme is secure under the security model. However the security model was wrong. Also the computational time and cost is more due to pairing operation in the verification process.

The remaining sections of the paper is organized as, we introduce the mathematical assumptions in section-2. We presented the framework of CL-AS scheme in section-3. Section-4 defines the adversary model. The scheme is proposed in section-5. Section-5 presents the proof of correctness and security of the proposed scheme. Finally, we conclude in section-7.

2 Preliminaries

2.1 Bilinear Pairings

Let $\mathbb{G}_1$ be a cyclic additive group of prime order q and $\mathbb{G}_2$ be a cyclic multiplicative group of the same prime order q. Let $\hat{e}$ be a bilinear map which is non-degenerated and computable called admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ if it satisfies the following properties:

- `Bilinearity`: Let $a, b \in \mathbb{Z}_q^*$ and $P, Q \in \mathbb{G}_1$
  1. $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $a, b \in \mathbb{Z}_q^*$
  2. $\hat{e}(P + Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$, for $P, Q, R \in \mathbb{G}_1$.
- `Non-degenerate`: There exists $P \in \mathbb{G}_1$ such that $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$
- `Computable`: There exist an efficient algorithm to compute $\hat{e}(P, Q)$ or all $P, Q \in \mathbb{G}_1$.

2.2 Computational Assumptions

In this section, we outline the mathematical hard problems on which the scheme relies.

Definition 1. Computational Diffie-Hellman(CDH)Problem:*Let $\mathbb{G}$ be an additive cyclic group with generator P. Given $P, aP, bP \in \mathbb{G}$, for any random numbers $a, b \in \mathbb{Z}_q^*$, compute abP.*

There is an probabilistic polynomial time(PPT) solvable algorithm $\mathcal{A}$ has negligible advantage ϵ in solving CDH problem in $\mathbb{G}$ if the $Pr[\mathcal{A}(P, aP, bP) = abP] \leq \epsilon$, where ϵ is a small positive integer and the probability is over the selection of $P \in \mathbb{G}$, random numbers $a, b\mathbb{Z}_q^*$ and the security parameter 1^μ. This can be formally presented by the following definition.

Definition 2. Computational Diffie-Hellman(CDH)Assumption: *The assumption (t, ϵ)-CDH holds in $\mathbb{G}$ if there does not exist any PPT algorithm with running time t has advantage ϵ in solving CDH problem.*

Table 1. Notations used and meanings

Notation	Meaning
CL-AS:	Certificateless Aggregate Signature
CL-ASS:	Certificateless Aggregate Short Signature
$\mathcal{A}_I/\mathcal{A}_{II}$:	`Type-I` and II adversary
ID_i:	Identity of any arbitrary user
x_i:	Secret value chosen by the user with identity ID_i
$<D_i, D_i^{'}>$:	Partial private key of the user with identity ID_i
pk_i:	Public key of the user with identity ID_i
μ:	A security parameter
$\mathbb{Z}_q$:	Set of elements $\{0, 1 \ldots q-1\}$ of an additive group and $\mathbb{Z}_q^* = \mathbb{Z}_q \diagdown 0$
m_i:	An arbitrary message belongs to the message space $\mathcal{M}$
$H_0, H_0^{'}, H_1, H_2, H_2^{'}$ and H_3:	Cryptographic hash functions
$\perp$:	Null value.

3 Certificateless Aggregate Signature Scheme

In this section, we define the components of Certificateless Aggregate Signature(CL-AS) Scheme.

3.1 Framework

A CL-AS scheme is composed of six polynomial time algorithms. This is associated with a KGC and a set of n users $\mathcal{U}_1 \ldots \mathcal{U}_n$ known as aggregating set $\mathbb{U}$ participating in the scheme.

- `Setup`: $(s, params) \leftarrow \texttt{Setup}(1^\mu)$. This algorithm is run by KGC. It takes a security parameter μ as input and generates master secret key s and system parameters $params$.
- `Partial-Private-Key`: $D_i \leftarrow \texttt{Partial} - \texttt{Private} - \texttt{Key}(ID_i, s, params)$. This algorithm is also run by KGC to generate partial private key of the participating user. It takes the user's identity ID_i, master secret key s and system parameters $params$ as input and returns the partial private key D_i of the corresponding user.
- `User-KeyGen`:$x_i \leftarrow \texttt{User} - \texttt{KeyGen}(ID_i)$, $pk_i \leftarrow \texttt{User} - \texttt{KeyGen}(ID_i, x_i)$. This algorithm is performed by the user. It takes the user's identity ID_i as input, picks a random $x_i \in \mathbb{Z}_q^*$ and returns the secret value x_i and public key $pk_i = \ x_i P$ of the corresponding user.
- `Sign`: $\sigma_i \leftarrow \texttt{Sign}(\triangle, ID_i x_i, D_i, pk_i, m_i)$. This algorithm is run by any arbitrary user $\mathcal{U}_i$ from the aggregating set $\mathbb{U}$ as signer. He chooses some state information $\triangle$ from public system parameters. The input of the algorithm are the state information $\triangle$, the message $m_i \in \mathbb{M}$ on which the signature is to be generated, signer's identity ID_i, public key pk_i and the signing key (x_i, D_i).
- `Aggregate`: $\sigma \leftarrow \texttt{Aggregate}(\triangle, \sigma_i, ID_i, pk_i)$ for all $1 \leq i \leq n$. The algorithm compressed the signatures $\sigma_1 \ldots \sigma_n$ generated by the corresponding users $\mathcal{U}_1 \ldots \mathcal{U}_n$ from the aggregating set $\mathbb{U}$. It takes σ_i, messages m_i, the state information $\triangle$ and public key pk_i (for all $1 \leq i \leq n$) as input. It returns the aggregate signature σ.
- `Verify`: "`TRUE/FALSE`" $\leftarrow Verify(\triangle, \sigma, ID_i, pk_i)$. This algorithm is use to verify the correctness of the aggregate signature. It takes the state information $\triangle$, identities $ID_1 \ldots ID_n$ from the aggregating set $\mathbb{U}$ of n the users $\mathcal{U}_1 \ldots \mathcal{U}_n$, public key $pk_1 \ldots pk_n$ of the corresponding user, aggregate signature σ generated on the messages $m_1 \ldots m_n$ as input. It pass through the verification equation. If it holds returns "`TRUE`", otherwise "`FALSE`".

4 Adversary Model and Security

Following are the two type of adversaries are involved.

- `Type-I` adversary ($\mathcal{A}_I$): The adversary behaves as a common dishonest user in the model. It does not have access of the master secret key of KGC and has the capability to replace a value of his choice with the public key.
- `Type-II` adversary ($\mathcal{A}_{II}$): The adversary behaves as honest user but is inquisitive to generates the user's partial private key and is allowed to access master secret key, but is not capable replace public key of the targeted user.

Security of CLS scheme is defined through two games between the adversary $\mathcal{A}_I/\mathcal{A}_{II}$ and challenger $\mathcal{C}$. The two games are defined as:

Game-I

`Setup`: $\mathcal{C}$ performs the `Setup` algorithm taking the security parameter μ as input, generates the system parameter and master secret key as output. $\mathcal{C}$ provides the system parameter to $\mathcal{A}_I$ and keeps secret the master key.

Attack

The adversary $\mathcal{A}_I$ submits polynomially bounded number of queries to the following oracles in an adaptive manner.

- `Partial-Private-Key queries` $\mathcal{PPK}(ID_i)$: $\mathcal{A}_I$ submits the request for partial private key for any arbitrary user ID_i. $\mathcal{C}$ obtains the partial private key D_i for the corresponding user as output.
- `Public-Key queries` $\mathcal{PK}(ID_i)$: $\mathcal{A}_I$ can submit the request of public key for any arbitrary user ID_i. $\mathcal{C}$ returns pk_i for the corresponding user as output.
- `Secret-Value queries` $\mathcal{SV}(ID_i)$: $\mathcal{A}_I$ can submit the request of secret value x_i for the user ID_i. $\mathcal{C}$ returns the output the secret value x_i.

– `Replace-Public-Key queries` $\mathcal{RPK}(ID_i, pk_i^{'})$: $\mathcal{A}_I$ submits the replacement queries choosing a new public key $pk_i^{'}$. Subsequently $\mathcal{A}_I$ sets the new public key $pk_i^{'}$ and $\mathcal{C}$ keeps the record of all replacements.
– `CL-Sign queries` $\mathcal{SGN}(\triangle_i, m_i, ID_i, pk_i)$: $\mathcal{A}_I$ can submit a request of signature of the user's identity ID_i with the state information $\triangle_i$, message m_i and public key pk_i. After submitting the query $\mathcal{SGN}(\triangle_i, m_i, ID_i, pk_i)$, $\mathcal{C}$ returns a valid signature σ_i.

Forgery: $\mathcal{A}_I$ obtains an aggregating set of n user $\mathbb{U}^* = \{\mathcal{U}_1^* \dots \mathcal{U}_n^*\}$ with identities from the aggregating set $\mathbb{X}_{ID}^* = \{ID_1^* \dots ID_n^*\}$, corresponding public key set $\mathbb{X}_{pk}^* = \{pk_1^* \dots pk_n^*\}$ the set of n messages $\mathbb{X}_m^* = \{m_1^* \dots m_n^*\}$ along with a state information $\triangle^*$, the aggregate signature σ^*. The adversary $\mathcal{A}_I$ successes or wins the `Game-I` if and only if the following holds:

- The signature generated is a valid aggregate signature σ^* on messages $\mathbb{X}_m^* = \{m_1^* \dots m_n^*\}$ along with a state information $\triangle^*$ of the identities $\mathbb{X}_{ID}^* = \{ID_1^* \dots ID_n^*\}$ and the corresponding public key set $\mathbb{X}_{pk}^* = \{pk_1^* \dots pk_n^*\}$.
- The identity $ID_i^* \in \mathbb{X}_{ID}^*$ of any arbitrary user has been queried to $\mathcal{PPK}(ID_i)$ and the same $\mathcal{SGN}(\triangle_i^*, m_i^*, ID_i^*, pk_i^*)$ has not been submitted before.

Game-II

`Setup`: The algorithm is run by the challenger $\mathcal{C}$. It takes as input the security parameter μ, returns the system parameter *params* and master secret key s. $\mathcal{C}$ provides the two parameters *params* and s to $\mathcal{A}_{II}$.
Attack The adversary $\mathcal{A}_{II}$ submits polynomially bounded number of queries to the following oracles in an adaptive manner.

– `Public-Key queries` $\mathcal{PK}(ID_i)$: $\mathcal{A}_{II}$ can submit the request of public key for any arbitrary user ID_i. $\mathcal{C}$ returns pk_i for the corresponding user as output.
– `Secret-Value queries` $\mathcal{SV}(ID_i)$: $\mathcal{A}_I$ can submit the request of secret value x_i for the user ID_i. $\mathcal{C}$ returns as output the secret value x_i.
– `CL-Sign queries` $\mathcal{SGN}(\triangle_i, m_i, ID_i, pk_i)$: $\mathcal{A}_{II}$ can submit a request of signature of the user's identity ID_i with the state information $\triangle_i$, on message m_i. After submitting the query $\mathcal{SGN}(\triangle_i, m_i, ID_i, pk_i)$, $\mathcal{C}$ returns a valid signature σ_i on message m_i with state information $\triangle_i$ of the corresponding user with identity ID_i and public key pk_i.

Forgery: $\mathcal{A}_I$ obtains an aggregating set of n user $\mathbb{U}^* = \{\mathcal{U}_1^* \dots \mathcal{U}_n^*\}$ with identities the set $\mathbb{X}_{ID}^* = \{ID_1^* \dots ID_n^*\}$ and corresponding public keys from the set $\mathbb{X}_{pk}^* = \{pk_1^* \dots pk_n^*\}$, the set of n messages $\mathbb{X}_m^* = \{m_1^* \dots m_n^*\}$ along with a state information $\triangle^*$ and the aggregate signature σ^*. The adversary $\mathcal{A}_I$ successes or wins the `Game-II` if and only if the following holds:

- The signature generated is a valid aggregate signature σ^* on messages from $\mathbb{X}_m^* = \{m_1^* \dots m_n^*\}$ along with a state information $\triangle^*$ of the identities $\mathbb{X}_{ID}^* = \{ID_1^* \dots ID_n^*\}$ and the corresponding public key set $\mathbb{X}_{pk}^* = \{pk_1^* \dots pk_n^*\}$.
- The identity $ID_i^* \in \mathbb{X}_{ID}^*$ of any arbitrary user has been queried to $\mathcal{PPK}(ID_i)$ queries and the same $\mathcal{SGN}(\triangle_i^*, m_i^*, ID_i^*, pk_i^*)$ has not been submitted before.

Definition 3. *A CL-AS signature scheme is said to be existentially unforgeable against adaptive chosen message attacks, if the probability of success of polynomially bound attackers $\mathcal{A}_I$ and $\mathcal{A}_{II}$ in the above two games are negligible.*

5 Certificateless Aggregate Short Signature(CL-ASS) Scheme

The scheme consists of the following six algorithms.

– `Setup`: This algorithm is run by KGC. Follows the steps
  1. It takes a security parameter μ, chooses an cyclic additive group $\mathbb{G}_1$ of prime order q has generator P, a cyclic multiplicative group $\mathbb{G}_2$ of same order q and admissible bilinear map $e : \mathbb{G}_1 \times G_1 \to \mathbb{G}_2$.
  2. Picks a random scalar $s \in \mathbb{Z}_q^*$ and computes $P_{pub} = sP$.

3. Considers three cryptographic hash functions $H_0, H_0^{'}, H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2, H_2^{'} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$.

- `Partial-Private-Key-Extract`: The algorithm takes $params$, master secret key s and user's identity $ID_i \in \{0,1\}^*$. It performs the following computations and generates the partial private key of the corresponding user.
  1. $Q_i = H_0(ID_i), Q_i^{'} = H_0^{'}(ID_i)$
  2. $D_i = s \cdot Q_i, D_i^{'} = s \cdot Q_i^{'}$
- `Set-Secret-Value`: The algorithm is used to set a secret value of an arbitrary user taking the security parameter μ, picks a random scalar $x_i \in \mathbb{Z}_q^*$ as input and returns x_i as secret value of the corresponding user.
- `Set-Public-Key`: This algorithm takes the user's secret values x_i and computes the public key of the corresponding user ID_i as $pk_i = x_i \cdot P$.
- `Sign`: This algorithm generates the signature on any arbitrary message $m_i \in \mathcal{M}$. It takes the signing key $(x_i, <D_i, D_i^{'}>)$ (of the signer with identity ID_i) and the public key pk_i and perform the following computation choosing the state information $\triangle$(It can take some elements from the public system parameter)
  1. $W = H_1(\triangle)$, $T_i = H_2(m_i \| pk_i \| \triangle \| ID_i)$
  2. $h_i = H_3(m_i \| pk_i \| ID_i)$
  3. $h_i^{'} = H_3^{'}(m_i \| pk_i \| ID_i)$
  4. $V_i = x_i T_i + h_i D_i + h_i^{'} D_i^{'} + x_i W$
- `Aggregate`: This algorithm is performed by any of the user who can aggregate a collection of individual signature that use some state information $\triangle$. Let $\mathbb{U} = \{\mathcal{U}_1 \ldots \mathcal{U}_n\}$ be an aggregate set with the corresponding identities $ID_i, ID_2 \ldots ID_n$ and the corresponding public keys are $pk_1, pk_2 \ldots pk_n$. The message signature pairs are (m_i, σ_i) for $1 \leq i \leq n$. The algorithm aggregates and generates the aggregate signature as $\sigma = \sum_{i=1}^{n} \sigma_i$.
- `Aggregate-Verify`: The aggregate signature σ signed by n users $U_1, U_2 \ldots U_n$ with identities $ID_1, ID_2 \ldots ID_n$ is verified. The verifier takes the corresponding user's identities $ID_1, ID_2 \ldots ID_n$ and the public keys $pk_1, pk_2 \ldots pk_n$ on messages $m_1, m_2 \ldots m_n$ with the same state of information $\triangle$ as input and computes the following for all i, $1 \leq i \leq n$.
  1. $W = H_1(\triangle)$, $Q_i = H_0(ID_i), Q_i^{'} = H_0^{'}(ID_i)$.
  2. $T_i = H_2(m_i \| pk_i \| \triangle \| ID_i)$
  3. $h_i = H_3(m_i \| pk_i \| ID_i)$
  4. $h_i^{'} = H_3^{'}(m_i \| pk_i \| ID_i)$

  After the above computations, the verifier checks the validity of signature with the following equation.

$$e(\sigma, P) = \prod_{i=1}^{n} e(T_i, pk_i) e\left(\sum_{i=1}^{n} h_i Q_i + h_i^{'} Q_i^{'}, P_{pub}\right) e\left(W, \sum_{i=1}^{n} pk_i\right) \tag{1}$$

  The aggregate signature is accepted if and only if the equation holds and returns $\perp$.

6 Analysis of the Scheme

This section analyzes the consistency and performance. Also presented the security of the proposed scheme.

6.1 Consistency

$$\begin{aligned}
e(\sigma, P) &= e(\textstyle\sum_{i=1}^{n} V_i, P) = e(\sum_{i=1}^{n}(x_i T_i + h_i D_i + h_i^{'} D_i^{'} + x_i W), P) \\
&= e(\textstyle\sum_{i=1}^{n} x_i T_i, P) e(\sum_{i=1}^{n}(h_i s Q_i + h_i^{'} s Q_i^{'}), P) e(\sum_{i=1}^{n} x_i W, P) \\
&= \textstyle\prod_{i=1}^{n} e(T_i, x_i P) \prod_{i=1}^{n} e((h_i Q_i + h_i^{'} Q_i^{'}), sP) \prod_{i=1}^{n} e(W, x_i P) \\
&= \textstyle\prod_{i=1}^{n} e(T_i, pk_i) \prod_{i=1}^{n} e((h_i Q_i + h_i^{'} Q_i^{'}), P_{pub}) \prod_{i=1}^{n} e(W, pk_i) \\
&= \textstyle\prod_{i=1}^{n} e(T_i, pk_i) e(\sum_{i=1}^{n}(h_i Q_i + h_i^{'} Q_i^{'}), P_{pub}) e(W, \sum_{i=1}^{n} pk_i)
\end{aligned}$$

6.2 Security

In this section, we have proven the security of the proposed scheme with the assumption of CDH problem is hard *i.e* computationally infeasible to solve.

Theorem 1. *In the random oracle model, if the adversary $\mathcal{A}_{II}$ of* `Type-I` *has a non-negligible advantage ϵ against the EUF-CMA security of the proposed scheme in duration of time t for a security parameter μ, and performing at most q_{H_0}, $q_{H_0^{'}}$ queries to oracles H_0, q_{H_1} queries to H_1 oracle, q_{H_2} queries to H_2 oracle, q_{H_3}, $q_{H_3^{'}}$ queries to oracles H_3, q_{ppk} to partial private key oracle, q_{pk} queries to public key oracle and q_s signing queries to signing oracle, then there exists an algorithm $\mathcal{B}$ that can solve CDH problem in $\mathbb{G}_1$ with time $t + \mathcal{O}(q_{H_0} + q_{H_1} + q_{H_2} + q_{H_3} + q_{ppk} + q_{pk} + q_s)\mathcal{T}_{\mathbb{G}_1}$ with probability $\epsilon^* \geq \frac{\epsilon}{(q_{H_0}+n)}$, where $\mathcal{T}_{\mathbb{G}_1}$ is the computational time for scalar multiplication in $\mathbb{G}_1$ and n is the size of the aggregating set.*

Proof. Let us assume that, there exists a super `Type-I` adversary $\mathcal{A}_I$ which has an advantage in attacking the proposed CL-ASS scheme. Let us construct an algorithm $\mathcal{B}$ that applies $\mathcal{A}_I$ to solve CDH problem *i.*e the algorithm takes the CDH instance (P, aP, bP) for randomly picking the scalar $a, b \in \mathbb{Z}_q^*$ and P be an element in $\mathbb{G}_1$. The goal of $\mathcal{A}_I$ is to compute abP. $\mathcal{B}$ runs $\mathcal{A}_I$ as subroutine and simulates the adversary model defined in Game-I. $\mathcal{B}$ initializes $P_{pub} = aP$, where a is the master secret key and $\mathcal{B}$ does not know the value of a and provides the system parameters to $\mathcal{A}_I$. $\mathcal{B}$ maintains lists L_0, L_1, L_2 and L_3 to simulate the hash oracles H_0, H_1, H_2 and H_3 respectively. Further to store all answers for partial private key, public key and signing queries $\mathcal{B}$ maintains the lists. $\mathcal{A}_I$ performs the following queries adaptively.

- $H_0, H_0^{'}$ queries: Assume that, $\mathcal{A}_I$ submits at most q_{H_0} queries to the hash oracle $H_0H_0^{'}$. The list L_{H_0} stores the tuples $(ID_i, Q_i, Q_i^{'}, \alpha_i, \beta_i, b_i)$. At the beginning of the simulation, list is empty. $\mathcal{B}$ selects $j \in [1, q_{H_0}]$ at random. When $\mathcal{A}_I$ submits H_0 and $H_0^{'}$ query on any arbitrary ID_i for $1 \leq i \leq q_{H_0}$, it returns the same answers from L_{H_0}, if the request has been asked before.
  Otherwise, $\mathcal{B}$ picks two random $\alpha_i, \beta_i \in \mathbb{Z}_q^*$ and flips a coin $b_i = [0,1]$ that yields 0 with probability ξ and 1 with probability $1-\xi$. If $b_i = 0$, then $\mathcal{B}$ sets $Q_i = bP, Q_i^{'} = \alpha_i(\beta_i P - bP)$ and adds $<ID_i, Q_i, Q_i^{'}, \alpha_i, \beta_i, b_i>$ to the list L_{H_0}.
  Otherwise $\mathcal{B}$ picks $\gamma_i, \gamma_i^{'} \in \mathbb{Z}_q^*$ at random, returns $Q_i = \gamma_i P, Q^{'} = \gamma_i^{'} P$ and adds $<ID_i, Q_i, Q_i^{'}, \gamma_i, \gamma_i^{'}, b_i>$ to L_{H_0}.
- H_1 queries:$\mathcal{B}$ maintains a list as L_{H_1} which is null at the beginning of the simulation. It keeps the tuples $<\triangle_i, W_i, \theta_i>$. When $\mathcal{A}_I$ submits the query $H_1(\triangle_i)$. This returns the same answer from the list L_{H_1} if it has been requested before. Otherwise $\mathcal{B}$ picks a random $\theta_i \in \mathbb{Z}_q^*$, returns $W_i = \theta_i P$, adds $<\triangle_i, W_i, \theta_i>$ and returns W_i as answer.
- H_2 queries: $\mathcal{B}$ maintains a list as L_{H_2}. At the beginning of the simulation, the list is empty. It keeps all the tuples $<m_i, pk_i, ID_i, \triangle_i, \psi_i, T_i>$. $\mathcal{A}_I$ submits a query on $(m_i \| pk_i \| \triangle_i \| ID_i)$, if the list L_{H_2} contains the tuples $<m_i, pk_i, ID_i, \triangle_i, \psi_i, T_i>$, $\mathcal{B}$ returns $T_i(= \psi_i P)$. Otherwise $\mathcal{B}$ picks a random $\psi_i \in \mathbb{Z}_q^*$ and returns $T_i = \psi_i P$ and add $<m_i, pk_i, ID_i, \triangle_i, \psi_i, T_i>$ to L_{H_2}.
- $H_3, H_3^{'}$ queries: $\mathcal{B}$ submits this query on $(m_i \| pk_i \| ID_i$. It maintains a list L_{H_3}. At the beginning of the simulation, the list is empty. It keeps the tuples $<m_i, pk_i, ID_i, h_i, h_i^{'}>$. If this list contains this entry, it returns $(h_i, h_i^{'})$ as answer. Otherwise $\mathcal{B}$ chooses $h_i, h_i^{'} \in \mathbb{Z}_q^*$ at random, returns $(h_i, h_i^{'})$ and $<m_i, pk_i, ID_i, h_i, h_i^{'}>$ to L_{H_3}.
- `Partial-Private-Key` queries: $\mathcal{A}_I$ submits a query on ID_i to oracle $\mathcal{PPK}(ID_i)$. $\mathcal{B}$ searches the entry $<ID_i, Q_i, Q_i^{'}, \theta_i, \theta_i^{'}>$ in the list L_{H_0}. It returns the same answer from the list L_{H_0} if the request has been submitted before. Otherwise it performs the following steps:
  1. If $b_i = 0$ abort the simulation.
  2. Otherwise computes $D_i = \gamma_i aP$ and $D_i^{'} = \gamma_i^{'} aP$
- `Secret-value` queries: $\mathcal{B}$ maintains a list L_{sv} to keep the tuples $<ID_i, pk_i, x_i>$. At the beginning of the simulation, the list is empty. $\mathcal{A}_I$ submits the secret value query $\mathcal{SV}(ID_i)$ on ID_i, $\mathcal{B}$ chooses a random $x_i \in \mathbb{Z}_q^*$, computes $pk_i = x_i P$, returns x_i and add $<ID_i, pk_i, x_i>$ to L_{sv}.

- `Public-Key` queries: $\mathcal{B}$ submits the public key query $\mathcal{PK}(ID_i)$, it returns the same answer from the list L_{sv}, if the request has been submitted before. Otherwise, $\mathcal{B}$ performs the following:
  - If entry $<ID_i, pk_i, x_i>$ is in L_{sv}, the public key pk_i of the user with identity ID_i is $\perp$. $\mathcal{B}$ chooses $x_i^{'} \in \mathbb{Z}_q^*$, computes $pk_i^{'} = x_i^{'}P$ and returns the answer is $pk_i^{'}$. Then upgrade the entry $<ID_i, pk_i, x_i>$ to $<ID_i, pk_i^{'}, x_i^{'}>$ in the list L_{sv}.
  - Otherwise, chooses $x_i \in \mathbb{Z}_q^*$, computes $pk_i = x_iP$ and the answer return is pk_i. Includes pk_i to L_{sv}.
- `Replace-Public-Key` queries: $\mathcal{A}_I$ selects a new public key for the user with identity ID_i and submits a Replace-Public-Key query $\mathcal{RPK}(ID_i, pk_i^{'})$, $\mathcal{B}$ search the entry $<ID_i, pk_i, x_i>$ on the list L_{sv}, if found on the list L_{sv}, set $pk_i = pk_i^{'}$ and $x_i = \perp$. Otherwise $\mathcal{B}$ runs $\mathcal{SV}(ID_i)$ query, updates pk_i to $pk_i^{'}$ and sets $x_i = \perp$.
- `Sign` queries: $\mathcal{A}_I$ submits sign query $\mathcal{SGN}(ID_i, m_i, \triangle_i, pk_i)$, $\mathcal{B}$ searches the entry $<ID_i, Q_i, Q_i^{'}, \gamma_i, \gamma_i^{'}>$ from the list L_{H_0}, $<ID_i, pk_i, x_i>$ from L_{sv} list and $<\triangle_i, W_i, \theta_i>$ from L_{H_1} list. Then $\mathcal{B}$ does as follows:
  - If $b_i = 0$, chooses $\psi_i, \theta_i, h_i \in \mathbb{Z}_q^*$ at random and computes $h_i^{'} = h_i\alpha_i^{-1}$, $\sigma_i = \psi_i pk_i + \theta_i pk_i + ah_i\beta_i P$. Returns σ_i and add the tuple $<m_i, pk_i, ID_i, \triangle_i, \psi_i, T_i>$ to L_{H_2}, $<m_i, pk_i, ID_i, h_i, h_i^{'}>$ to L_{H_3} and $<\triangle_i, W_i, \theta_i>$ to L_{H_1} list.
  - Otherwise for $b_i = 1$, $\mathcal{B}$ selects four random $\psi_i, \theta_i, h_i, h_i^{'} \in \mathbb{Z}_q^*$ and computes $\sigma_i = \psi_i pk_i + \theta_i pk_i + h_i\gamma_i aP + h_i^{'}\gamma_i^{'}aP$.

`Forgery`

Eventually $\mathcal{B}$ returns a valid CL-ASS σ^* with a set $\mathbb{U}$ of n users $U_1 \ldots U_n$. The corresponding identities are from the aggregating sets $\mathbb{L}_{ID^*} = \{ID_1^* \ldots ID_n^*\}$ and the set of public key $\mathbb{L}_{pk} = \{pk_1^* \ldots pk_n^*\}$ of the corresponding users and a state information $\triangle^*$. There are two cases in the simulations.

1. If $b_i = 0$, $\mathcal{B}$ aborts the simulation and returns ``fail"
2. Else for $b_i = 1$, $\mathcal{B}$ recovers the tuples $<m_i^*, pk_i^*, ID_i^*, \triangle_i^*, \psi_i^*, T_i^*>$, $<m_i^*, pk_i^*, ID_i^*, h_i^*, h_i^{'*}>$ and $<\triangle_i^*, W_i^*, \theta_i^*>$ from the list L_{H_2}, L_{H_3} and L_{H_1} respectively. $\mathcal{A}$ replaces the public key pk_i. Since it successes to generate valid CL-ASS, the following equation holds.

$$e(\sigma^*, P) = \prod_{i=1}^{n} e(T_i^*, pk_i^*) e\Big(\sum_{i=1}^{n} h_i^* Q_i^* + h_i^{'*} Q_{i^*}^{'}, P_{pub}\Big) e\Big(W^*, \sum_{i=1}^{n} pk_i^*\Big)$$

$$\begin{aligned}
&\prod_{i=1}^{n} e(T_i^*, pk_i^*) e\Big(\sum_{i=1}^{n} h_i^* Q_i^* + h_i^{'*} Q_{i^*}^{'}, P_{pub}\Big) e\Big(W^*, \sum_{i=1}^{n} pk_i^*\Big) \\
&= \prod_{i=1}^{n} e(\psi_i^* P, pk_i^*) e\Big(\sum_{i=1}^{n} h_i^* bP + h_i^{'}\alpha_i(\beta_i P - bP), aP\Big) e\Big(\theta_i P, \sum_{i=1}^{n} pk_i^*\Big) \\
&= \prod_{i=1}^{n} e(\psi_i^* P, pk_i^*) e\Big(\sum_{i=1}^{n} h_i^* bP + h_i^{*'}\alpha_i(\beta_i P - bP), aP\Big) e\Big(\theta_i^* P, \sum_{i=1}^{n} pk_i^*\Big) \\
&= \prod_{i=1}^{n} e(\psi_i^* pk_i^*, P) e\Big(\sum_{i=1}^{n} h_i^* abP + h_i^{*'}\alpha_i a(\beta_i P - bP), P\Big) e\Big(\sum_{i=1}^{n} pk_i^* \theta_i^*, P\Big) \\
&= \prod_{i=1}^{n} e(\psi_i^* pk_i^*, P) e\Big(\sum_{i=1}^{n} h_i abP + h_i^{*'}\alpha_i a(\beta_i P - bP), P\Big) e\Big(\sum_{i=1}^{n} pk_i^* \theta_i^*, P\Big) \\
&= e\Big(\sum_{i=1}^{n} (\psi_i^* pk_i^*, P)\Big) e\Big(\sum_{i=1}^{n} h_i^* abP + h_i^{*'}\alpha_i a(\beta_i P - bP), P\Big) e\Big(\sum_{i=1}^{n} pk_i^* \theta_i^*, P\Big) \\
&= e\Big(\sum_{i=1}^{n} (\psi_i^* pk_i^* + h_i^* abP + h_i^{*'}\alpha_i a(\beta_i P - bP) + pk_i^* \theta_i^*, P)\Big)
\end{aligned}$$

Hence $\sigma = \sum_{i=1}^{n} (\psi_i^* pk_i^* + h_i abP + h^{*'}{}_i \alpha_i a(\beta_i P - bP) + pk_i^* \theta_i^*$

$$\begin{aligned}
&= \sum_{i=1}^{n} (\psi_i^* pk_i^* + h_i^* abP + h_i^{*'}\alpha_i\beta_i aP - h_i^{'}\alpha_i\beta_i abP) + pk_i^* \theta_i \\
&= \sum_{i=1}^{n} (\psi_i^* pk_i^* + pk_i^* \theta_i^* + abP(h_i^* - h_i^{*'}\alpha_i) + h_i^{*'}\alpha_i\beta_i aP \\
&= \sum_{i=1}^{n} (\psi_i^* pk_i^* + pk_i^* \theta_i^*) + abP \sum_{i=1}^{n} (h_i^* - h_i^{*'}\alpha_i) + \sum_{i=1}^{n} h_i^{*'}\alpha_i\beta_i aP
\end{aligned}$$

$$\Rightarrow abP\sum_{i=1}^{n}(h_i^* - h_i^{*'}\alpha_i) = \sigma - \sum_{i=1}^{n}(\psi_i^* pk_i^* + pk_i^*\theta_i^*) - \sum_{i=1}^{n} h^{*'}_i \alpha_i\beta_i aP$$

$$\Rightarrow abP = \frac{\sigma - \sum_{i=1}^{n}(\psi_i^* pk_i^* + pk_i^*\theta_i^*) - aP\sum_{i=1}^{n} h_i^{*'}\alpha_i\beta_i}{\sum_{i=1}^{n}(h_i^* - h_i^{*'}\alpha_i)}$$

`Probability of success`

We compute probability of success of $\mathcal{B}$. $\mathcal{B}$ solves CDH problem with probability $\epsilon^* \geq \frac{\epsilon}{(q_{H_0}+n)\mu}$. During the simulation, there exists the following events for success of $\mathcal{B}$.

- $\mathcal{E}_1$: $\mathcal{B}$ does not abort the simulation during the $\mathcal{A}_I$'s partial private key queries.
- $\mathcal{E}_2$: $\mathcal{A}_I$ can generate a valid and nontrivial forged aggregate signature.
- $\mathcal{E}_3$: Event $\mathcal{E}_2$ happen, $b_i^* = 0$ for one of i, $[i = 1 \dots n]$ and $b_i^* = 1$ for other value of indices of i.

If all the three events happen, then $\mathcal{B}$ succeeds.The advantage of $\mathcal{B}$ is

$$Adv_{\mathcal{B}}^{CDH} = Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3]$$

$Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] = Pr[\mathcal{E}_1]Pr[\mathcal{E}_2 \mid \mathcal{E}_1]Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2]$

Lemma 1. *The probability of $\mathcal{B}$ does not abort the simulation as long as the $\mathcal{A}_I$'s partial private key queries continues is at least $(1-\xi)^{q_{H_0}}$. So $Pr[\mathcal{E}_1] \geq (1-\xi)^{q_{H_0}}$*

Proof. The probability that $\mathcal{B}$ does not abort the simulation for a query of partial private key extraction is $(1-\xi)$. The maximum number of queries submitted to the partial private key oracles $\mathcal{PPK}(ID_i)$ is q_{H_0}. Hence the probability of $\mathcal{B}$ does not abort the simulation as a result of $\mathcal{PPK}(ID_i)$ is at least $(1-\xi)^{q_{H_0}}$.

Lemma 2. *The probability of $\mathcal{B}$ does not abort the simulation over the extraction of partial private key and signing queries of $\mathcal{A}_I$ is at least ϵ (where ϵ is small positive integer).*

Proof. Let $\mathcal{B}$ does not abort the simulation over the extraction of partial private key queries of $\mathcal{A}_I$. Under this circumstances, $\mathcal{A}_I$ generates a valid and nontrivial forged aggregate signature. The probability of observing event $\mathcal{E}_2$, given that $\mathcal{E}_1$ is true is $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1]$. Then the algorithm view of $\mathcal{A}_I$ is identical to its view in the actual attack. Hence $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq \epsilon$.

Lemma 3. *The probability that $\mathcal{B}$ does not abort the simulation after $\mathcal{A}_I$ returning a valid and nontrivial forged aggregate signature is at least $\xi(1-\xi)^{n-1}$. So $Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2] \geq \xi(1-\xi)^{n-1}$.*

Proof. Let both the events $\mathcal{E}_1$ and $\mathcal{E}_2$ happen at a time and $\mathcal{A}_I$ returns a valid and nontrivial forgery $(ID_1^* \dots ID_n^*; pk_1^* \dots pk_n^*; m_1^* \dots m_n^*; \sigma^*)$. $\mathcal{B}$ aborts the simulation as long as $\mathcal{A}_I$ is not generating a forgery such that $b_j = 0$ for some $j \in \{1 \dots n\}$ and $b_i = 1$, for all other $i \in \{1 \dots n\}/\{j\}$. Hence $\Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2] \geq \xi(1-\xi)^{n-1}$.

$Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] \geq (1-\xi)^{q_{H_0}}\epsilon\xi(1-\xi)^{n-1} = \epsilon\xi(1-\xi)^{q_{H_0}+n-1}$.
Let $\xi = \frac{1}{q_{H_0}+n}$. Hence $\epsilon\xi(1-\xi)^{q_{H_0}+n-1} = \epsilon\frac{1}{q_{H_0}+n}\left(1 - \frac{1}{q_{H_0}+n}\right)^{q_{H_0}+n-1} \geq \epsilon\frac{1}{q_{H_0}+n}$.

Theorem 2. *In the random oracle model, if the adversary $\mathcal{A}_{II}$ of* `Type-II` *has a non-negligible advantage ϵ against the EUF-CMA security of the proposed CL-ASS scheme in the adversary model of* `Game-II` *in time span t for a security parameter μ, and performing at most q_{pk} number of public key queries, then there exists an algorithm $\mathcal{B}$ that can solve CDH problem in $\mathbb{G}_1$ with time $t + \mathcal{O}(q_{H_1} + q_{H_2} + q_{H_3} + q_{sv})\mathcal{T}_{\mathbb{G}_1}$ with probability $\epsilon^* \geq \frac{\epsilon}{(q_{pk}+n)}$, where $\mathcal{T}_{\mathbb{G}_1}$ is the computational time for scalar multiplication in $\mathbb{G}_1$ and n is the size of the aggregating set.*

Proof. Let us assume that, there exists a super `Type-II` adversary $\mathcal{A}_{II}$ which has an advantage in attacking the proposed CL-ASS scheme in the adversary model of `Game-II`. Let to construct an algorithm $\mathcal{B}$ that applies $\mathcal{A}_{II}$ to solve CDH problem *i*.e the algorithm takes the CDH instance (P, aP, bP) for randomly picking the scalar $a, b \in \mathbb{Z}_q^*$ and P be an element in $\mathbb{G}_1$. The goal of $\mathcal{A}_{II}$ is to compute abP. $\mathcal{B}$ runs $\mathcal{A}_{II}$ as subroutine and simulates the adversary model of `Game-II`. $\mathcal{B}$ initializes $P_{pub} = sP$, where s is the master secret key. $\mathcal{B}$ provides the system parameters along with the master secret key to $\mathcal{A}_{II}$. Since $\mathcal{A}_{II}$ is allowed to access the master secret key, he can perform `Partial-Private-Key` extraction query.

$\mathcal{B}$ maintains four lists $L_{H_0}, L_{H_1}, L_{H_2}, L_{H_3}$ and L_{sv} to simulate the hash oracles $<H_0, H_0^{'}>, H_1, H_2, <H_3, H_3^{'}>$ and secret value oracle respectively. $\mathcal{A}_{II}$ performs the following queries in an adaptive manner.

- $H_0, H_0^{'}$ queries: Assume that, $\mathcal{A}_{II}$ submits at most q_{H_0} queries to the hash oracle $H_0 H_0^{'}$. $\mathcal{B}$ maintains a list L_{H_0} to store the tuples $(ID_i, Q_i, Q_i^{'})$. At the beginning of the simulation, list is empty. $\mathcal{B}$ selects Q_i and $Q_i^{'}$ at random, returns $<Q_i, Q_i^{'}>$ and add to L_{H_0}.
- H_1 queries:$\mathcal{B}$ maintains a list as L_{H_1}. At the beginning of the simulation, the list is empty. It keeps the tuples $<\triangle_i, W_i, \theta_i>$. When $\mathcal{A}_{II}$ submits the query $H_1(\triangle_i)$. This returns the same answer from the list L_{H_1} if it has been requested before. Otherwise $\mathcal{B}$ picks a random $\theta_i \in \mathbb{Z}_q^*$, returns $W_i = \theta_i P$, adds $<\triangle_i, W_i, \theta_i>$ to L_{H_1} and returns W_i as answer.
- H_2 queries: $\mathcal{B}$ maintains a list as L_{H_2}. At the beginning of the simulation, the list is empty. It keeps the tuples $<m_i, pk_i, ID_i, \triangle_i, \psi_i, T_i>$. $\mathcal{A}_I$ submits a query on $(m_i \| pk_i \| \triangle_i \| ID_i)$, if the list L_{H_2} contains the tuples $<m_i, pk_i, ID_i, \triangle_i, \psi_i, T_i>$, $\mathcal{B}$ returns T_i. Otherwise $\mathcal{B}$ picks a random $\psi_i \in \mathbb{Z}_q^*$, returns $T_i = \psi_i aP$ as answer and add $<m_i, pk_i, ID_i, \triangle_i, \psi_i, T_i>$ to L_{H_2}.
- $H_3, H_3^{'}$ queries: $\mathcal{B}$ submits this query on $(m_i \| pk_i \| ID_i$. At the beginning of the simulation, the list L_{H_3} is empty. It keeps the tuples $<m_i, pk_i, ID_i, h_i, h_i^{'}>$. If this list contains this entry, it returns $(h_i, h_i^{'})$ as answer. Otherwise $\mathcal{B}$ chooses $h_i, h_i^{'} \in \mathbb{Z}_q^*$ at random, returns $(h_i, h_i^{'})$ as answer and adds $<m_i, pk_i, ID_i, h_i, h_i^{'}>$ to L_{H_3}.
- `Public-Key` queries: $\mathcal{B}$ maintains a list L_{sv} containing the tuples $<ID_i, pk_i, x_i, b_i>$. At the beginning of the simulation, the list is empty. $\mathcal{B}$ submits at most q_{pk} public key query $\mathcal{PK}(ID_i)$ on ID_i, $1 \leq i \leq q_{H_0}$. $\mathcal{B}$ selects $x_i \in \mathbb{Z}_q^*$, then tosses a coin $b_i \in \{0,1\}$ that come up 0 with probability ξ and 1 with probability $1-\xi$. If $b_i = 0$, $\mathcal{B}$ returns $pk_i = bP$ and add $<ID_i, pk_i, x_i = \perp, b_i>$ to L_{sv}, otherwise, $\mathcal{B}$ computes $pk_i = x_i P$ and adds $<ID_i, pk_i, x_i, b_i>$. The answer return is pk_i.
- `Secret-value` queries: $\mathcal{B}$ maintains a list L_{sv} to keep the tuples $<ID_i, pk_i, x_i>$. At the beginning of the simulation the list is empty. $\mathcal{A}_I$ submits the query on ID_i, $\mathcal{B}$ chooses a random $x_i \in \mathbb{Z}_q^*$, computes $pk_i = x_i P$, returns x_i and add $<ID_i, pk_i, x_i>$ to L_{sv}.
- `Public-Key-Request`: Let $\mathcal{A}_{II}$ submits at most q_{pk} times Public-Key-Request queries. $\mathcal{B}$ picks $j \in [1, q_{pk}]$ at random. When $\mathcal{A}_{II}$ submits `Public-Key-Request` query on ID_j to oracle $\mathcal{RPK}(ID_j)$, it returns the same answer from the list L_{sv} if it has been requested before, otherwise $\mathcal{B}$ chooses $x_i \in \mathbb{Z}_q^*$ and tosses a coin $b_i = [0,1]$ come up 0 with probability ξ and 1 with probability $1-\xi$.
  - If $b_i = 0$, it returns $pk_i = bP$ and adds $<ID_i, pk_i, x_i, b_i>$ to L_{sv}.
  - Otherwise, $b_i = 1$, $\mathcal{B}$ computes $pk_i = x_i P$, adds the entry $<ID_i, pk_i, x_i, b_i>$ to L_{sv} and returns the output pk_i.
- `Sign` queries: $\mathcal{A}_I$ submits sign query $\mathcal{SGN}(ID_i, m_i, \triangle_i, pk_i)$, $\mathcal{B}$ searches the entry $<ID_i, Q_i, Q_i^{'}, \gamma_i, \gamma_i^{'}>$ from the list L_{H_0}, $<ID_i, pk_i, x_i>$ from L_{sv} list and $<\triangle_i, W_i, \theta_i>$ from L_{H_1} list. $\mathcal{B}$ generates the signature as follows:
  - If $b_i = 0$, chooses $\psi_i, \theta_i, h_i \in \mathbb{Z}_q^*$ at random and sets $\sigma_i = \psi_i bP + h_i D_i + h_i^{'} D_i^{'} + \theta_i pk_i$. Returns σ_i and add the tuple $<m_i, pk_i, ID_i, \triangle_i, \psi_i, T_i>$ to L_{H_2}, $<m_i, pk_i, ID_i, h_i, h_i^{'}>$ to L_{H_3} and $<\triangle_i, W_i, \theta_i>$ to L_{H_1} list.
  - Otherwise, $b_i = 1$, $\mathcal{B}$ selects four random $\psi_i, \theta_i, h_i, h_i^{'} \in \mathbb{Z}_q^*$ and computes $\sigma_i = \psi_i pk_i + h_i s Q_i + h_i^{'} s Q_i^{'} + \theta_i pk_i$.

<u>`Forgery`</u>

Eventually $\mathcal{B}$ returns a valid CL-ASS σ^* with a set $\mathbb{U}$ of n users $U_1 \ldots U_n$. The corresponding identities are from the aggregating sets $\mathbb{L}_{ID^*} = \{ID_1^* \ldots ID_n^*\}$ and the set of public key $\mathbb{L}_{pk} = \{pk_1^* \ldots pk_n^*\}$ of the corresponding users and a state information $\triangle^*$.

1. If $b_i = 0$, $\mathcal{B}$ aborts the simulation and returns "`fail`".

2. $\mathcal{B}$ searches the entry $<m_i^*, pk_i^*, ID_i^*, \triangle_i^*, \psi_i^*, T_i^*>$, $<m_i^*, pk_i^*, ID_i^*, h_i^*, h_i^{*'}>$ and $<\triangle_i^*, W_i^*, \theta_i^*>$ from the list L_{H_2}, L_{H_3} and L_{H_1} respectively. $\mathcal{A}_{II}$ replaces the public key pk_i^*. Since it returns a valid CL-ASS, the following equation holds.

$$e(\sigma, P) = \prod_{i=1}^{n} e(T_i^*, pk_i^*) e\Big(\sum_{i=1}^{n} h_i^* Q_i^* + h_i^{*'} Q_i^{*'}, P_{pub}\Big) e\Big(W^*, \sum_{i=1}^{n} pk_i^*\Big)$$

$$\prod_{i=1}^{n} e(T_i^*, pk_i^*) e\Big(\sum_{i=1}^{n} h_i^* Q_i^* + h_i^{*'} Q_i^{*'}, P_{pub}\Big) e\Big(W^*, \sum_{i=1}^{n} pk_i^*\Big)$$

$$= \prod_{i=1}^{n} e(\psi_i^* aP, pk_i^*) e\Big(\sum_{i=1}^{n} h_i^* Q_i^* + h_i^{*'} Q_i^{*'}, P_{pub}\Big) e(W^*, \sum_{i=1}^{n} pk_i^*\Big)$$

$$= \prod_{i=1}^{n} e(\psi_i^* aP, pk_i^*) e\Big(\sum_{i=1}^{n} h_i^* Q_i^* + h_i^{*'} Q_i^{*'}, sP)\Big) e\Big(\theta_i^* P, \sum_{i=1}^{n} pk_i^*\Big)$$

$$= \prod_{i=1}^{n} e(abP\psi_i^*, P) e\Big(s\sum_{i=1}^{n} h_i^* Q_i^* + h_i^{*'} Q_i^{*'}, P)\Big) e\Big(\sum_{i=1}^{n} pk_i^* \theta_i^*, P\Big)$$

$$= e\Big(\sum_{i=1}^{n} (abP\psi_i^*, P)\Big) e\Big(s\sum_{i=1}^{n} h_i^* Q_i^* + h_i^{*'} Q_i^{*'}, P)\Big) e\Big(\sum_{i=1}^{n} pk_i^* \theta_i^*, P\Big)$$

$$= e\Big(\sum_{i=1}^{n} (abP\psi_i^* + sh_i^* Q_i^* + sh_i^{*'} Q_i^{*'} + \theta_i^* pk_i^*, P)\Big)$$

Hence $\sigma = \sum_{i=1}^{n} (abP\psi_i^* + sh_i^* Q_i^* + sh_i^{*'} Q_i^{*'} + \theta_i^* pk_i^*$

$$\Rightarrow abP \sum_{i=1}^{n} \psi_i^* = \sigma - \sum_{i=1}^{n} (sh_i^* Q_i^* + sh_i^{*'} Q_i^{*'} + \theta_i^* pk_i^*)$$

$$\Rightarrow abP = \frac{\sigma - \sum_{i=1}^{n} (sh_i^* Q_i^* + sh_i^{*'} Q_i^{'} + \theta_i^* pk_i^*)}{\sum_{i=1}^{n} \psi_i^*}$$

Probability of success

We compute probability of success for $\mathcal{B}$ to solve the given instances of CDH problem. The probability is $\epsilon^* \geq \frac{\epsilon}{(q_{H_0}+n)\mu}$. We consider the following events for success of $\mathcal{B}$.

- $\mathcal{E}_1$: $\mathcal{B}$ does not abort the simulation during the $\mathcal{A}_{II}$'s secret value queries.
- $\mathcal{E}_2$: $\mathcal{A}_I$ can generate a valid and nontrivial forged aggregate signature.
- $\mathcal{E}_3$: Event $\mathcal{E}_2$ happen, $b_i^* = 0$ for one of i, $[i = 1 \ldots n]$ and $b_i^* = 1$ for other value of indices of i.

If all the three events happen, then $\mathcal{B}$ succeeds. The advantage of $\mathcal{B}$ is

$$Adv_{\mathcal{B}}^{CDH} = Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3]$$

$Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] = Pr[\mathcal{E}_1] Pr[\mathcal{E}_2 \mid \mathcal{E}_1] Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2]$, $\epsilon^* = Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3]$

Lemma 4. *The probability of $\mathcal{B}$ does not abort the simulation as long as the $\mathcal{A}_{II}$'s secret value key queries continues is at least $(1-\xi)^{q_{pk}}$. So $Pr[\mathcal{E}_1] \geq (1-\xi)^{q_{pk}}$*

Proof. The probability that $\mathcal{B}$ does not abort the simulation for a query of secret value is $(1-\xi)$. The maximum number of queries submitted to $\mathcal{PK}(ID_i)$ is q_{pk}. Hence the probability of $\mathcal{B}$ does not abort the simulation as a result of $\mathcal{PK}(ID_i)$ is at least $(1-\xi)^{q_{pk}}$.

Lemma 5. *The probability of $\mathcal{B}$ does not abort the simulation over the extraction of public key and signing queries of $\mathcal{A}_{II}$ is at least ϵ (where ϵ is small positive integer).*

Proof. Let $\mathcal{B}$ does not abort the simulation over the extraction of partial private key queries of $\mathcal{A}_{II}$. Under this circumstances, $\mathcal{A}_{II}$ generates a valid and nontrivial forged aggregate signature. The probability of observing event $\mathcal{E}_2$, given that $\mathcal{E}_1$ is true is $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1]$. Then the algorithm view of $\mathcal{A}_{II}$ is identical to its view in the actual attack. Hence $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq \epsilon$.

Lemma 6. *The probability that $\mathcal{B}$ does not abort the simulation after $\mathcal{A}_{II}$ returning a valid and nontrivial forged aggregate signature is at least $\xi(1-\xi)^{n-1}$. So $Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2] \geq \xi(1-\xi)^{n-1}$.*

Proof. Let both the events $\mathcal{E}_1$ and $\mathcal{E}_2$ happen at a time and $\mathcal{A}_I$ returns a valid and nontrivial forgery $(ID_1^* \ldots ID_n^*; pk_1^* \ldots pk_n^*; m_1^* \ldots m_n^*; \sigma^*)$. $\mathcal{B}$ aborts the simulation as long as $\mathcal{A}_I$ is not generating a forgery such that $b_j = 0$ for some $j \in \{1 \ldots n\}$ and $b_i = 1$, for all other $i \in \{1 \ldots n\}/\{j\}$. Hence $\Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \wedge \mathcal{E}_2] \geq \xi(1-\xi)^{n-1}$.

$Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] \geq (1-\xi)^{q_{pk}} \epsilon\xi(1-\xi)^{n-1} = \epsilon\xi(1-\xi)^{q_{pk}} + n - 1.$

Let $\xi = \frac{1}{q_{pk}+n}$. Hence $\epsilon\xi(1-\xi)^{q_{pk}+n-1} = \epsilon\frac{1}{q_{pk}+n}\left(1 - \frac{1}{q_{H_{pk}}+n}\right)^{q_{H_{pk}}+n-1} \geq \epsilon\frac{1}{q_{pk}+n}$.

7 Conclusion

In this paper, we have proposed an efficient and provably secure certificateless aggregate signature scheme of short length. Our scheme adopted CL-PKC that guarantees the validity of the public key without certificate signed by TTP. Also it greatly reduces the computational cost and communication overhead. Our scheme is proven to be existentially unforgeable under the adversary model with the assumption of hardness of solving CDH problem. Our scheme can be implemented on constrained hand held devices such as cell phone, smart card, PDA etc.

References

1. A. Shamir "Identity-based cryptosystems and signature schemes", *in Advances in Cryptology, vol. 196 of Lecture Notes in Computer Science*, pp. 47–53, Springer, Berlin, Germany, 1984.
2. K.C. Barr and K. Asanovic Energy aware lossless data compression. In Proceeding of. of Mobisys 2005
3. S. S. Al-Riyami and K. G. Paterson "Certificateless public key cryptography",*in Advances in Cryptology-ASIACRYPT 2003,vol. 2894 of Lecture Notes in Computer Science*, pp. 452–473, Springer, Berlin, Germany, 2003.
4. T. Suzuki, Z. Ramzan, H. Fujimoto, C. Gentry, T. Nakayama, and R. Jain. A system for end-to-end authentication of adaptive multimedia content. In Proceeding of Conference on Communications and Multimedia Security, 2004.
5. E. Mykletun, M. Narasimha, and G. Tsudik Signature bouquets: Immutability for aggregated/condensed signatures. In Proceeding of ESORICS 2004, pages 160-176, 2004.
6. K. Y. Choi, J. H. Park, and D. H. Lee "A new provably secure certificateless short signature scheme",*Computers and Mathematics with Applications*, vol. 61,no.7, pp. 1760–1768, 2011.
7. J. Cheon, Y. Kim, H. Yoon A new id-based signature with batch verification, Cryptology ePrint Archive, Report 2004/131.
8. X. Cheng, J. Liu, L. Guo, X. Wang Identity-based multisignature and aggregate signature schemes from m-torsion groups, Journal of Electronics (China) 23 (2006) 569-573.
9. J. Xu, Z. Zhang, D. Feng ID-based aggregate signatures from bilinear pairings, in: Y.G. Desmedt et al. (Eds.), CANS 2005, LNCS 3810, Springer-Verlag, Shenzhen, China, 2005, pp. 110-119.
10. C. Gentry, Z. Ramzan Identity-based aggregate signatures, in:M. Yung et al. (Eds.), PKC 2006, LNCS 3958, Springer-Verlag, New York, USA, 2006, pp. 257-273.
11. X. Huang, W. Susilo, Y. Mu, and F. Zhang "On the security of certificateless signature schemes from ASIACRYPT 2003", *in Cryptology and Network Security, vol. 3810 of Lecture Notes in Computer Science*, pp. 13–25, Springer, Berlin, Germany, 2005.
12. D. Yum, P. Lee Generic construction of certificateless signature, in: H. Wang et al. (Eds.), ACISP 2004, LNCS 3108, Springer-Verlag, Sydney, Australia, 2004, pp. 200-211.
13. B. Hu, D. Wong, Z. Zhang, X. Deng Key replacement attack against a generic construction of certificateless signature, in: L. Batten, R. Safavi-Naini (Eds.), ACISP 2006, LNCS 4058, Springer-Verlag, Melbourne, Australia, 2006, pp. 235- 346.
14. X. Huang, Y.Mu, W. Susilo, D. S.Wong, and W.Wu "Certificateless signature revisited in Information Security and Privacy", vol. 4586 of Lecture Notes in Computer Science, pp. 308-322, Springer, Berlin, Germany, 2007.
15. K. Choi, J. Park, J. Hwang, D. Lee Efficient certificateless signature schemes, in: J. Katz, M. Yung (Eds.), ACNS 2007, LNCS 4521, Springer-Verlag, Zhuhai, China, 2007, pp. 443-458.
16. Z. Gong, Y. Long, X. Hong, K. Chen Two certificateless aggregate signatures from bilinear maps, in: SNPD 2007, IEEE Press, Qingdao, China, 2007, pp. 188- 193.
17. Yu-Chi Chen, R. Tso and G. Horng "Cryptanalyis of a Provably Secure Certificateless Short Signature Scheme", *Intelligence Systems & Applications*, pp-61-68, 2013.
18. X. Li, K. Chen, L. Sun "Certificateless signature and proxy signature schemes from bilinear pairings", *Lithuanian Mathematical Journal* 45(1), pp.76–83, 2005.

19. W.S. Yap, S.H. Heng, B.M. Goi "An efficient certificateless signature scheme, emerging directions in embedded and ubiquitous computing",*in EUC Workshops 2006, LNCS, vol. 4097, Springer-Verlag*, pp. 322–331, 2006.
20. K. Y. Choi, J. H. Park, and D. H. Lee "A new provably secure certificateless short signature scheme",*Computers and Mathematics with Applications*, vol. 61,no.7, pp. 1760–1768, 2011.
21. D. Boneh, C. Gentry, B. Lynn, H. Shacham Aggregate and verifiably encrypted signatures from bilinear maps, in: E. Biham (Ed.), EUROCRYPT 2003, LNCS 2656, Springer-Verlag, Warsaw, Poland, 2003, pp. 416-432.
22. A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham Sequential aggregate signatures from trapdoor permutations, in: C. Cachin, J. Camenisch (Eds.), Eurocrypt 2004, LNCS 3027, Springer-Verlag, Interlaken, Switzerland, 2004, pp. 74-90.
23. Z. Zhang, D. Feng Key replacement attack on a certificateless signature scheme, Cryptology ePrint Archive: Report 2006/453.
24. M.H. Au, J. Chen, J.K. Liu, Y. Mu, D.S. Wong, G. Yang " Malicious KGC attacks in certificateless cryptography", *in: ASIACCS'07, ACM*, pp. 302-311, 2007. available at Cryptology ePrint Archive: Report 2006/255.
25. X. Cao, K.G. Paterson, W. Kou An attack on a certificateless signature scheme, Cryptology ePrint Archive: Report 2006/367.