Computer Networks 54 (2010) 2482–2491

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Efficient many-to-one authentication with certificateless aggregate signatures

Lei Zhang [a,*], Bo Qin [a,b], Qianhong Wu [a,c], Futai Zhang [d,e]

[a] Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Av. Països Catalans 26, E-43007 Tarragona, Catalonia, Spain
[b] Department of Maths, School of Science, Xi'an University of Technology, China
[c] School of Computer, Key Lab. of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan University, China
[d] School of Computer Science and Technology, Nanjing Normal University, Nanjing, China
[e] Jiangsu Engineering Research Center on Information Security and Privacy Protection Technology, Nanjing, China

ARTICLE INFO

Article history:
Received 1 March 2009
Received in revised form 20 November 2009
Accepted 6 April 2010
Available online 11 April 2010
Responsible Editor: R. Molva

Keywords:
Information security
Message authentication
Digital signature
Certificateless cryptography

ABSTRACT

Aggregate signatures allow an efficient algorithm to aggregate n signatures of n distinct messages from n different users into one single signature. The resulting aggregate signature can convince a verifier that the n users did indeed sign the n messages. This feature is very attractive for authentications in bandwidth-limited applications such as reverse multicasts and senor networks. Certificateless public key cryptography enables a similar functionality of public key infrastructure (PKI) and identity (ID) based cryptography without suffering from complicated certificate management in PKI or secret key escrow problem in ID-based cryptography. In this paper, we present a new efficient certificateless aggregate signature scheme which has the advantages of both aggregate signatures and certificateless cryptography. The scheme is proven existentially unforgeable against adaptive chosen-message attacks under the standard computational Diffie–Hellman assumption. Our scheme is also very efficient in both communication and computation and the proposal is practical for many-to-one authentication.

1. Introduction

Digital signature is one of the most important primitives in public key cryptography. Knowing the public key of a signer, anyone can verify whether a signature of this signer is valid or not. This feature enables signatures to be efficiently applied to one-to-one (unicast) and one-to-many (multicast) applications. Some multicast applications may require the root node to collect data from leaf nodes (e.g., real-time fee collection, user preferences) which results into a many-to-one communication. In these applications, the root node is very likely to be swamped when too many leaves transmit simultaneously. Hence, to provide the usual authenticity, integrity and non-repudiation, signatures have to be elegantly designed to avoid the known implosion problem in many-to-one communications [1].

1.1. Related work

Recently, the concept of aggregate signatures [4] was introduced by Boneh et al. at Eurocrypt 2003. This notion allows an efficient algorithm to aggregate n signatures of n distinct messages from n different users into one single signature. The resulting aggregate signature can convince a verifier that the n users did indeed sign the n original messages. These properties greatly reduce the length of the resulting signature to be verified. Hence, aggregate signatures can be applied to the above applications.

To let a signature scheme function, the public key has to be bound with the identity of the owner of the public key. Traditionally, this is provided by the public key infrastructure (PKI) in which one or more third parties, known as

* Corresponding author. Tel.: +34 977558270; fax: +34 977559710.
E-mail addresses: lei.zhang@urv.cat (L. Zhang), bo.qin@urv.cat (B. Qin), qianhong.wu@urv.cat (Q. Wu), zhangfutai@njnu.edu.cn (F. Zhang).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.04.008

certificate authorities (CAs), issue digital certificates to bind a user and his public key. In this paradigm, before using the public key of a user, the participant must first verify the certificate of the user, which implies a large amount of computing and storage cost to manage certificates. Observing these shortcomings, Shamir put forward identity-based public key cryptography (ID-PKC) [16] to simplify certificate management in PKI systems. In an ID-based cryptosystem, the identity, for instance, the telephone number, email or IP address, of a user functions as the public key of the user. However, in ID-PKC a trusted third party called Private Key Generator (PKG) must be employed to generate the private key for each user. The user's private key is computed from his public known identity and PKG's master secret key. Hence, ID-PKC suffers from a key escrow problem which implies that all the users have to fully trust PKG. This might be a too strong assumption in some applications.

To address the key escrow problem of ID-PKC, Al-Riyami and Paterson [2] invented a new paradigm called certificateless public key cryptography (CL-PKC). CL-PKC also exploits a third party called Key Generation Center (KGC) to help a user to generate his private key. However, the KGC can merely determine part of the private key for each user, rather than the whole private key of each user in ID-PKC systems. In CL-PKC, the user computes the resulting private key with the partial private key obtained from the KGC and the secret information chosen by the user. As for the public key of the user, it is computed from the KGC's public parameters and the user's secret information. As a result, CL-PKC systems avoid the key escrow problem in ID-PKC systems and the complicated certificate management problem in traditional PKI systems.

The above advantages in CL-PKC motivate a lot of further studies [11,14,18,22]. In [2], Al-Riyami and Paterson presented the first Certificateless Signature (CLS) scheme, however, no formal proof is given. Huang et al. [12] pointed out a security drawback of the primal CLS scheme in [2] and defined the first security model of CLS schemes. But the model in [12] did not fully catch the ability of the adversaries in CL-PKC. A CLS scheme proved secure in this model may be insecure in practice. A recent example is Yap et al.'s scheme [18], which was broken by Park [15] and Zhang and Feng [19] independently. Later, Zhang et al. [20] improved the security model of CLS schemes and presented a more efficient CLS scheme. The security model of CLS schemes was further developed in [11,13].

It is natural to investigate the notion of aggregate signatures in ID-based or certificateless cryptographic contexts. By far, several Identity-based Aggregate Signature (IDAS) schemes have been presented [6–8,10,17]. But most of them only achieve partial aggregation [7,10,17]. In [6], Cheng et al. proposed an aggregate signature scheme whose output has the same size as an individual signature. However, it needs that the signers are pre-determined, and without the individual signature from each pre-determined signer, all individual signatures cannot be aggregated. At present, in ID-PKC, the only instance that can aggregate n individual signatures into a single aggregate signature (whose length is as short as an individual signature) is Gentry and Ramzan's IDAS scheme. In addition, their scheme requires only three pairing operations and is proven secure under the standard computational Diffie–Hellman (CDH) assumption. As to the aggregate signatures in the certificateless public key setting, Gong et al. [9] presented two certificateless aggregate signature (CLAS) schemes which are provably secure in a relatively weak model similar to that in [12]. Later, Zhang and Zhang [21] presented a CLAS scheme which is provably secure in a stronger model. However, as for efficiency, all the previous CLAS schemes require a relatively large amount of paring computation in the process of verification and have long outputs and may be further improved.

1.2. Our contribution

In this paper, we present a novel certificateless aggregate signature (CLAS) scheme. By exploiting the random oracle model [3], our CLAS scheme is proven existentially unforgeable against adaptive chosen-message attacks under the standard CDH assumption. It allows multiple signers to sign multiple documents in an efficient way and the total verification information (the length of the signature), consists only two group elements. Our scheme is also very efficient in computation, and the verification procedure needs only a very small constant number of pairing computations, independent of the number of aggregated signatures. With our CLAS scheme, one can aggregate many different certificateless signatures into a single certificateless aggregate signature, and hence effectively reduce the signature size and verification cost. This implies that our scheme is very applicable to secure many-to-one communications.

The rest of the paper is organized as follows. Section 2 reviews the notion and the security model of CLAS schemes. We propose our new CLAS scheme in Section 3, and prove its security in Section 4. In Section 5, we compare our scheme with three existing proposals, followed by some concluding remarks in the last section.

2. Preliminaries

2.1. Modeling certificateless aggregate signature schemes

A CLAS scheme involves a KGC, an aggregating set U of n users $\mathcal{U}_1, \cdots, \mathcal{U}_n$, and an aggregate signature generater. It consists of following six algorithms.

- Setup: This algorithm is performed by KGC that accepts a security parameter $\ell$to generate a master-key and a list of system parameters params.
- Partial-Private-Key-Extract: This algorithm is performed by KGC that accepts a user's identity ID_i, a parameter list params and a master-key to produce the user's partial private key D_i.
- UserKeyGen: An algorithm which is run by a user that takes as input the user's identity ID_i, and selects a random $x_i \in Z_q^*$ and outputs the user's secret/public key x_i/P_i.
- Sign: This algorithm is run by each user $\mathcal{U}_i$ in an aggregating set U. $\mathcal{U}_i$'s inputs are the parameter list params, some state information Δ,[1] a message $M_i \in \mathcal{M}$ ($\mathcal{M}$ is

[1] Depending on the instantiation, the state information Δ can be empty.

the message space), his identity ID_i, his corresponding public key P_i, and his signing key (x_i, D_i). The output of $\mathcal{U}_i$ is a signature σ_i on message M_i which is valid under his identity ID_i and the corresponding public key P_i.

- Aggregate: This algorithm is run by an aggregate signature generater that takes as inputs a state information Δ, an aggregating set U of n users $\{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$, the identity ID_i of each user $\mathcal{U}_i$, the corresponding public key P_i of $\mathcal{U}_i$, and a signature σ_i on a message M_i with state information Δ under identity ID_i and public key P_i for each user $\mathcal{U}_i \in U$. The output of this algorithm is an aggregate signature σ on messages $\{M_1, \ldots, M_n\}$.
- Aggregate Verify: This algorithm takes as input Δ, an aggregating set U of n users $\{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$, the identity ID_i and the corresponding public key P_i of each user $\mathcal{U}_i$, an aggregate signature σ on messages $\{M_1, \ldots, M_n\}$. It outputs true if the aggregate signature is valid, or $\perp$otherwise.

2.2. Security definitions

Two types of adversaries are considered in CL-PKC – Type I adversary and Type II adversary.[2] A Type I adversary $\mathcal{A}_I$ does not have access to the master-key, but he has the ability to replace the public key of any entity with a value of his choice. While a Type II Adversary $\mathcal{A}_{II}$ has access to the master-key but cannot perform public key replacement. More details can be found in [2].

The security of a CLAS scheme is modeled via the following two games[3] between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_I$ or $\mathcal{A}_I$.

Game 1 (for *Type I Adversary).*

Setup: $\mathcal{C}$ runs the Setup algorithm, takes as input a security parameter $\ell$to obtain a master-key and the system parameter list params. $\mathcal{C}$ then sends params to the adversary $\mathcal{A}_I$ while keeping the master-key secret.

Attack: The adversary $\mathcal{A}_I$ can perform a polynomially bounded number of the following types of queries in an adaptive manner.

- Partial-Private-Key *queries* $PPK(ID_i)$: $\mathcal{A}_I$ can request the partial private key of any user with identity ID_i. In response, $\mathcal{C}$ outputs the partial private key D_i of the user.
- Public-Key *queries* $PK(ID_i)$: $\mathcal{A}_I$ can request the public key P_i of a user whose identity is ID_i. In response, $\mathcal{C}$ outputs the public key for identity ID_i.
- Secret-Key *queries* $SK(ID_i)$: $\mathcal{A}_I$ can request the secret key of a user whose identity is ID_i. In response, $\mathcal{C}$ outputs the secret key x_i for identity ID_i (It outputs $\perp$, if the user's public key has been replaced).
- Public-Key-Replacement *queries* $PKR(ID_i, P_i')$: For any user whose identity is ID_i, $\mathcal{A}_I$ can choose a new public key P_i'. A_I then sets P_i' as the new public key of this user. $\mathcal{C}$ will record this replacement.
- Sign *queries* $S(\Delta_i, M_i, ID_i, P_i)$: $\mathcal{A}_I$ can request a user's (whose identity is ID_i) signature on a message M_i under a state information Δ_i. On receiving a query $S(\Delta_i, M_i, ID_i, P_i)$, $\mathcal{C}$ generates a signature σ_i on message M_i under identity ID_i and public key P_i using Δ_i as the state information, and replies with σ_i.

Forgery: $\mathcal{A}_I$ outputs a set of n users whose identities form the set $L^*_{ID} = \{ID_1^*, \ldots, ID_n^*\}$ and corresponding public keys form the set $L^*_{PK} = \{P_1^*, \ldots, P_n^*\}$, n messages form the set $L^*_M = \{M_1^*, \ldots, M_n^*\}$, a state information Δ^* and an aggregate signature σ^*.

We say that $\mathcal{A}_I$ wins Game 1, iff.

(1) σ^* is a valid aggregate signature on messages $\{M_1^*, \ldots, M_n^*\}$ with state information Δ^* under identities $\{ID_1^*, \ldots, ID_n^*\}$ and the corresponding public keys $\{P_1^*, \ldots, P_n^*\}$.
(2) At least one of the identities, without loss of generality, say $ID_1^* \in L^*_{ID}$ has not submitted during the Partial-Private-Key queries. And $S(\Delta^*, M_1^*, ID_1^*, *)$ has never been queried during the Sign queries.

Game 2 (for *Type II Adversary).*

Setup: $\mathcal{C}$ runs the Setup algorithm, takes as input a security parameter $\ell$to obtain the system parameter list params and also the system's master-key. $\mathcal{C}$ then sends params and master-key to the adversary $\mathcal{A}_{II}$.

Attack: The adversary $\mathcal{A}_{II}$ can perform a polynomially bounded number of the following types of queries in an adaptive manner.

- Public-Key *queries* $PK(ID_i)$: $\mathcal{A}_{II}$ can request the public key of a user (whose identity is ID_i) of his choice. In response, $\mathcal{C}$ outputs the public key P_i for identity ID_i.
- Secret-Key *queries* $SK(ID_i)$: $\mathcal{A}_{II}$ can choose a user whose identity is ID_i, and request this user's secret key. In response, $\mathcal{C}$ outputs the secret key x_i for identity ID_i.
- Sign *queries* $S(\Delta_i, M_i, ID_i, P_i)$: $\mathcal{A}_{II}$ can request a user's (whose identity is ID_i) signature on a message M_i with a state information Δ_i. On receiving a query $S(\Delta_i, M_i, ID_i, P_i)$, $\mathcal{C}$ replies with a signature σ_i on message M_i with state information Δ_i under identity ID_i and public key P_i.

Forgery: $\mathcal{A}_{II}$ outputs a set of n users whose identities form the set $L^*_{ID} = \{ID_1^*, \ldots, ID_n^*\}$ and corresponding public keys form the set $L^*_{PK} = \{P_1^*, \ldots, P_n^*\}$, n messages form the set $L^*_M = \{M_1^*, \ldots, M_n^*\}$, a state information Δ^* and an aggregate signature σ^*.

We say that $\mathcal{A}_{II}$ wins Game 2, iff.

(1) σ^* is a valid aggregate signature on messages $\{M_1^*, \ldots, M_n^*\}$ with state information Δ^* under identities $\{ID_1^*, \ldots, ID_n^*\}$ and corresponding public keys $\{P_1^*, \ldots, P_n^*\}$.
(2) One of the identities, without loss of generality, say $ID_1^* \in L^*_{ID}$ has not submitted during the Secret-Key queries. And $S(\Delta^*, M_1^*, ID_1^*, *)$ has never been queried during the Sign queries.

[2] In this paper, the ability of our Type I/II Adversary follows the original definition which is introduced by Al-Riyami and Paterson [2].

[3] Note in Game 1, when a type I adversary $\mathcal{A}_I$ submits a Public-Key-Replacement query $PKR(ID_i, P_i')$ to $\mathcal{C}$, $\mathcal{A}_I$ needs not submit the corresponding secret key which is used to generate P_i' to $\mathcal{C}$ [20].

Definition 1. A CLAS scheme is existentially unforgeable under adaptively chosen-message attack *iff* the success probability of any polynomially bounded adversary in any of the above two games is negligible.

3. Our certificateless aggregate signature scheme

3.1. Bilinear maps

Our scheme is realized in groups which allowing efficient bilinear maps [5]. Let G_1 be an additive group of prime order q and G_2 be a multiplicative group of the same order. A map $e{:}G_1 \times G_1 \to G_2$ is called a bilinear map if it satisfies the following properties:

(1) Bilinearity: $e(aP, bQ) = e(P,Q)^{ab}$ for all $P, Q \in G_1, a, b \in Z_q^*$.
(2) Non-degeneracy: there exists P, $Q \in G_1$ such that $e(P,Q) \neq 1$.
(3) Computability: there exists an efficient algorithm to compute $e(P,Q)$ for any P, $Q \in G_1$.

Computational Diffie–Hellman (CDH) problem in G_1: given a generator P of the group G_1 whose order is q, and given (aP,bP) for unknown $a, b \in Z_q^*$; compute abP.

3.2. Our certificateless aggregate signature scheme

The specification of the scheme is as follows.

- Setup: Given a parameter ℓ, the KGC chooses a cyclic additive group G_1 which is generated by P with prime order q, chooses a cyclic multiplicative group G_2 of the same order and a bilinear map $e{:}G_1 \times G_1 \to G_2$. The KGC also chooses a random $\lambda \in Z_q^*$ as the master-key and sets $P_T = \lambda P$, chooses cryptographic hash functions $H_1 \sim H_4 : \{0,1\}^* \to G_1, H_5 : \{0,1\}^* \to Z_q^*$. The system parameter list is params = $(G_1, G_2, e, P, P_T, H_1 \sim H_5)$. The message space is $\mathcal{M} = \{0,1\}^*$.
- Partial-Private-Key-Extract: This algorithm accepts params, master-key λ and a user's identity $ID_i \in \{0,1\}^*$, and generates the partial private key for the user as follows.
  (1) Compute $Q_{i,0} = H_1(ID_i, 0)$, $Q_{i,1} = H_1(ID_i, 1)$.
  (2) Output the partial private key $(D_{i,0}, D_{i,1}) = (\lambda Q_{i,0}, \lambda Q_{i,1})$.
- UserKeyGen: This algorithm takes as input params, a user's identity ID_i, selects a random $x_i \in Z_q^*$ and sets his secret/public key as $x_i / P_i = x_i P$.
- Sign: To sign a message $M_i \in \mathcal{M}$ using the signing key $(x_i, D_{i,0}, D_{i,1})$, the signer, whose identity is ID_i and the corresponding public key is P_i, first chooses a one-time-use state information Δ then performs the following steps.
  (1) Choose a random $r_i \in Z_q^*$, compute $R_i = r_i P$.
  (2) Compute $h_i = H_5(M_i \| \Delta \| ID_i \| P_i)$, $T = H_2(\Delta)$, $V = H_3(\Delta)$, $W = H_4(\Delta)$.
  (3) Compute $S_i = D_{i,0} + x_i V + h_i(D_{i,1} + x_i W) + r_i T$.
  (4) Output $\sigma_i = (R_i, S_i)$ as the signature on M_i.
- Aggregation: Anyone can act as an aggregate signature generater who can aggregate a collection of individual signatures that use the same state information Δ. For an aggregating set (which has the same state information Δ) of n users $\{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$ with identities $\{ID_1, \ldots, ID_n\}$ and the corresponding public keys $\{P_1, \ldots, P_n\}$, and message-signature pairs $(M_1, \sigma_1 = (R_1, S_1)), \ldots, (M_n, \sigma_n = (R_n, S_n))$ from $\{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$ respectively, the aggregate signature generater computes $R = \sum_{i=1}^n R_i, S = \sum_{i=1}^n S_i$ and outputs the aggregate signature $\sigma = (R,S)$.
- Aggregate Verify: To verify an aggregate signature $\sigma = (R,S)$ signed by n user $\{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$ with identities $\{ID_1, \ldots, ID_n\}$ and corresponding public keys $\{P_1, \ldots, P_n\}$ on messages $\{M_1, \ldots, M_n\}$ with the same state information Δ, the verifier performs the following steps.
  (1) Compute $T = H_2(\Delta), V = H_3(\Delta)$, $W = H_4(\Delta)$, and for all i, $1 \leqslant i \leqslant n$ compute $h_i = H_5(M_i \| \Delta \| ID_i \| P_i)$, $Q_{i,0} = H_1(ID_i, 0), Q_{i,1} = H_1(ID_i, 1)$.
  (2) Verify

$$e(S,P) \stackrel{?}{=} e\left(P_T, \sum_{i=1}^n Q_{i,0} + \sum_{i=1}^n h_i Q_{i,1}\right) e\left(V, \sum_{i=1}^n P_i\right) \times e\left(W, \sum_{i=1}^n h_i P_i\right) e(T,R).$$

If the equation holds, output true. Otherwise, output $\perp$.

In our scheme, each user in an aggregating set should use the same one-time-use state information Δ when signing. As mentioned in [8], it is straightforward to choose such a Δ in certain settings. For example, if the signers have access to some loosely synchronized clocks, Δ can be chosen based on the current time. Furthermore, if Δ is sufficiently long, then it will be statistically unique. And to remove the one-time-use restriction of Δ, Gentry and Ramzan also provided a method. For details, please refer to [8].

We notice that, in this paper, we follow the original definition of certificateless cryptography in [2]. In this setting, the KGC can choose a secret key $x_i' \in Z_q^*$ and compute $P_i' = x_i P$ as the new public key of the user with ID_i. By this replacement, the KGC can know both the partial private key and secret key of the user. This implies that KGC can generate signatures on behalf of that user. It seems that both the KGC and a user may deny having generated a signature. However, in [2], the authors introduced a binding technique[4] which ensures that users can only create one public key for which they know the corresponding private key. This way, the systems in certificateless cryptography can enjoy the same trust level as the systems in a traditional PKI. In other words, if the KGC replaces a public key, then the KGC necessarily leaves evidence of its bad behavior; and such an action can be easily detected, as the KGC is the only entity having that capability. Note that this scenario is equivalent to a CA forging a certificate in a traditional PKI system: the existence of two valid certificates for a single identity would implicate the CA behaved maliciously.

4. Security proof

Assuming that the CDH problem is hard, we now show the security of our CLAS scheme.

[4] This technique can also be applied to our scheme

Theorem 1. *In the random oracle model, if there exists a type I adversary $\mathscr{A}$ who has an advantage ε in forging a signature of our CLAS scheme in an attack modeled by Game 1, within a time span t for a security parameter ℓ; then the CDH problem in G_1 can be solved within time* $t + (4q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 2q_K + q_P + 6q_S)\tau_{G_1}$ *and with probability*

$$\varepsilon' \geqslant \left(1-\frac{1}{q_{H_1}}\right)^{q_K}\left(1-\frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H_5'}}\right)\right)^{q_S}\frac{1}{q_{H_1}} \times \frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H_5'}}\right)\varepsilon,$$

where τ_{G_1} is the time to compute a scalar multiplication in G_1, n is the aggregating set's scale.

Proof. Let $\mathscr{C}$ be a CDH attacker who receives a random instance (P, aP, bP) of the CDH problem in G_1 and has to compute the value of abP. $\mathscr{A}$ is a type I adversary who interacts with $\mathscr{C}$ as modeled in Game 1. We show how $\mathscr{C}$ can use $\mathscr{A}$ to solve the CDH problem, i.e. to compute abP.

To make the proof easy to read, we first briefly show where the reduction is. When the game begins, $\mathscr{C}$ sets $P_T = aP$ which is an instance of the CDH problem, and simulates hash functions as random oracles. During the simulation, $\mathscr{C}$ needs to guess $\mathscr{A}$'s target identity (without loss of generality, say ID_1^*), message (without loss of generality, say M_1^*) and state information Δ^*. $\mathscr{C}$ will set $Q_{1,0}^* = H_1(ID_1^*, 0) = \alpha_{i,0}^* P + \alpha_{i,0}'^* bP$, $Q_{1,1}^* = H_1(ID_1^*, 1) = \alpha_{1,1}^* P + \alpha_{1,1}'^* bP$, $T^* = H_2(\Delta^*) = \beta^* P$, $V^* = H_3(\Delta^*) = \gamma^* P$, $W^* = H_4(\Delta^*) = \pi^* P$ and $H_5(M_1^*, \Delta^*, ID_1^*, P_1^*) = h_1^*$. (For the other settings, please refer to our proof.) At the end of the game, $\mathscr{A}$ needs to output a set of n users whose identities form the set $L_{ID}^* = \{ID_1^*, \ldots, ID_n^*\}$ and corresponding public keys form the set $L_{PK}^* = \{P_1^*, \ldots, P_n^*\}$, n messages form the set $L_M^* = \{M_1^*, \ldots, M_n^*\}$, a state information Δ^* and an aggregate signature σ^*. Since σ^{ast} is valid, we have

$$e(S^*, P) = e\left(P_T, \sum_{i=1}^n Q_{i,0}^* + \sum_{i=1}^n h_i^* Q_{i,1}^*\right) e\left(V^*, \sum_{i=1}^n P_i^*\right) \times e\left(W^*, \sum_{i=1}^n h_i^* P_i^*\right) e(T^*, R^*),$$

where $Q_{i,j}^* = H_1(ID_i^*, j), j \in \{0,1\}$. We note that

$$e\left(P_T, Q_{1,0}^* + h_1^* Q_{1,1}^*\right) = e\left(aP, \alpha_{i,0}^* P + \alpha_{i,0}'^* bP + h_1^* \alpha_{1,1}^* P + h_1^* \alpha_{1,1}'^* bP\right).$$

It is easy to have

$$e(aP, bP) = \left(e\left(P_T, Q_{1,0}^* + h_1^* Q_{1,1}^*\right) \times e\left(\left(-\alpha_{i,0}^* - h_1^* \alpha_{1,1}^*\right) aP, P\right)\right)^{(\alpha_{1,0}'^* + h_1^* \alpha_{1,1}'^*)^{-1}}.$$

Furthermore

$$e\left(P_T, Q_{1,0}^* + h_1^* Q_{1,1}^*\right) = e(S^*, P)\left(e\left(P_T, \sum_{i=2}^n Q_{i,0}^* + \sum_{i=2}^n h_i^* Q_{i,1}^*\right) \times e\left(V^*, \sum_{i=1}^n P_i^*\right) e\left(W^*, \sum_{i=1}^n h_i^* P_i^*\right) e(T^*, R^*)\right)^{-1}.$$

By our setting, for $2 \leqslant i \leqslant n, Q_{i,j}^* = \alpha_{i,j}^* P$, where $j \in \{0,1\}$; $\mathscr{C}$ can get the solution of the CDH problem $abP = (\alpha_{1,0}'^* + h_1^* \alpha_{1,1}'^*)^{-1}(S^* - \sum_{i=2}^n \alpha_{1,0}^* P_T - \sum_{i=2}^n \alpha_{1,1}^* h_i^* P_T - \gamma^* \sum_{i=1}^n P_i^* - \pi^* \sum_{i=1}^n h_i^* P_i^* - \beta^* R^* - (\alpha_{1,0}^* + h_1^* \alpha_{1,1}^*) P_T)$.

Next, we begin to propose the concrete proof.

Setup: Firstly, $\mathscr{C}$ sets $P_T = aP$ and selects params = $(G_1, G_2, e, P, P_T, H_1 \sim H_5)$ then he sends params to $\mathscr{A}$.

Attack: We consider hash functions $H_1 \sim H_5$ as random oracles. And we assume $\mathscr{A}$ can ask at most q_{H_i} times H_i $(i = 1, \ldots, 5)$ *queries*, q_K times Partial-Private-Key *queries*, q_P times Public-Key *queries*, q_S times Sign *queries*. $\mathscr{A}$ can perform the following types of queries in an adaptive manner.

$\mathbf{H}_1$ *queries*: $\mathscr{C}$ maintains a list $\mathbf{H}_1$ of tuples $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ which is initially empty. $\mathscr{C}$ picks $I \in [1, q_{H_1}]$ uniformly at random. Whenever $\mathscr{C}$ receives an H_1 query on (ID_i, j) for $j \in \{0,1\}$, $\mathscr{C}$ does the following:

(1) If there is a tuple $(ID_k, \alpha_{k,0}, \alpha_{k,0}', \alpha_{k,1}, \alpha_{k,1}', Q_{k,0}, Q_{k,1})$ on $\mathbf{H}_1$ such that $ID_i = ID_k$, return $Q_{k,j}$ as answer.
(2) Else if $i = I$, randomly choose $\alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}' \in Z_q^*$, set $Q_{i,0} = \alpha_{i,0} P + \alpha_{i,0}' bP$, $Q_{i,1} = \alpha_{i,1} P + \alpha_{i,1}' bP$, add $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ to $\mathbf{H}_1$ and return $Q_{i,j}$ as answer.
(3) Else set $\alpha_{i,0}' = 0, \alpha_{i,1}' = 0$, randomly choose $\alpha_{i,0}, \alpha_{i,1} \in Z_q^*$, set $Q_{i,0} = \alpha_{i,0} P$, $Q_{i,1} = \alpha_{i,1} P$, add $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ to $\mathbf{H}_1$ and return $Q_{i,j}$ as answer.

H_2 *queries*: $\mathscr{C}$ keeps a initially empty list $\mathbf{H}_2$ of tuples (Δ_i, T_i, β_i), picks $J \in [1, q_{H_2}]$ uniformly at random. Whenever $\mathscr{A}$ issues a query $H_2(\Delta_i)$, the same answer from the list $\mathbf{H}_2$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ selects a random $\beta_i \in Z_q^*$; if $i = J$, computes $T_i = \beta_i P$, else sets $T_i = \beta_i aP$. Finally, $\mathscr{C}$ adds (Δ_i, T_i, β_i) to $\mathbf{H}_2$ and returns T_i as answer.

H_3 *queries*: $\mathscr{C}$ keeps a list $\mathbf{H}_3$ of tuples $(\Delta_j, V_j, \gamma_j)$. This list is initially empty. Whenever $\mathscr{A}$ issues a query Δ_i to H_3, the same answer from the list $\mathbf{H}_3$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ first selects a random $\gamma_i \in Z_q^*$, then computes $V_i = \gamma_i P$, adds $(\Delta_i, V_i, \gamma_i)$ to $\mathbf{H}_3$ and returns V_i as answer.

H_4 *queries*: $\mathscr{C}$ keeps a list $\mathbf{H}_4$ of tuples (Δ_j, W_j, π_j). This list is initially empty. Whenever $\mathscr{A}$ issues a query Δ_i to H_4, the same answer from the list $\mathbf{H}_4$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ first selects a random $\pi_i \in Z_q^*$, then computes $W_i = \pi_i P$, adds (Δ_i, W_i, π_i) to $\mathbf{H}_4$ and returns W_i as answer.

H_5 *queries*: $\mathscr{C}$ keeps a list $\mathbf{H}_5$ of tuples $(M_i, \Delta_i, ID_i, P_i, h_i)$. This list is initially empty. Whenever $\mathscr{A}$ issues a query $(M_i \| \Delta_i \| ID_i \| P_i)$ to H_5, the same answer from the list $\mathbf{H}_5$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ first submits $(ID_i, 0)$ to H_1 oracle, then finds the tuple $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ on $\mathbf{H}_1$, finally does the following:

(1) If $ID_i = ID_l$ and $\Delta_i = \Delta_J$ (we assume that $\mathscr{A}$ can ask at most $q_{H_5'} < q_{H_5}$ times such kind of queries), randomly choose $K \in [1, q_{H_5'}]$.
   (a) If it is K-th query, set $h_i = -\alpha_{i,0}'/\alpha_{i,1}'$, add $(M_i, \Delta_i, ID_i, P_i, h_i)$ to $\mathbf{H}_5$ and return h_i.
   (b) Else select a random $h_i \in Z_q^*$, add $(M_i, \Delta_i, ID_i, P_i, h_i)$ to $\mathbf{H}_5$ and return h_i as answer.

(2) Else, select a random $h_i \in Z_q^*$, add $(M_i, \Delta_i, ID_i, P_i, h_i)$ to $\mathbf{H}_5$ and return h_i as answer.

Partial-Private-Key *queries*: $\mathscr{C}$ keeps a list $\mathbf{K}$ of tuples $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$. This list is initially empty. When $\mathscr{A}$ issues a query $PPK(ID_i)$, the same answer from the list $\mathbf{K}$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ checks whether there is a tuple $(ID_i, \alpha_{i,0}, \alpha'_{i,0}, \alpha_{i,1}, \alpha'_{i,1}, Q_{i,0}, Q_{i,1}, c_i)$ on $\mathbf{H}_1$, if no, $\mathscr{C}$ makes an H_1 query on (ID_i, j) ($j = 0$ or 1) to generate such a tuple, finally does as follows.

(1) If $ID_i = ID_I$, abort.
(2) Else if there's a tuple $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ on $\mathbf{K}$, set $D_{i,0} = \alpha_{i,0} P_T, D_{i,1} = \alpha_{i,1} P_T$ and return $(D_{i,0}, D_{i,1})$ as answer.
(3) Else, first compute $D_{i,0} = \alpha_{i,0} P_T, D_{i,1} = \alpha_{i,1} P_T$, set $x_i = P_i = \perp$, then return $(D_{i,0}, D_{i,1})$ as answer and add $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ to $\mathbf{K}$.

Public-Key *queries*: On receiving a query $PK(ID_i)$, if the request has been asked before, the current public key from the list $\mathbf{K}$ will be given. Otherwise, $\mathscr{C}$ does as follows.

(1) If there is a tuple $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ on $\mathbf{K}$ (in this case, the public key P_i of ID_i is $\perp$), choose $x'_i \in Z_q^*$, compute $P'_i = x'_i P$, return P'_i as answer and update $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ to $(ID_i, x'_i, D_{i,0}, D_{i,1}, P'_i)$.
(2) Otherwise, choose $x_i \in Z_q^*$, compute $P_i = x_i P$, return P_i as answer, set $D_{i,0} = D_{i,1} = \perp$ and add $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ to $\mathbf{K}$.

Secret-Key *queries*: On receiving a query $SK(ID_i)$, $\mathscr{C}$ first makes $PK(ID_i)$ then finds the tuple $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ on $\mathbf{K}$ and returns x_i as answer (note that the value of x_i maybe $\perp$).

Public-Key-Replacement *queries*: $\mathscr{A}$ can choose a new public key for the user whose identity is ID_i. On receiving a query $PKR(ID_i, P'_i)$, $\mathscr{C}$ first finds the tuple $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ on $\mathbf{K}$ (if such a tuple does not exists on $\mathbf{K}$ or $P_i = \perp$, $\mathscr{C}$ first makes $PK(ID_i)$), then $\mathscr{C}$ updates P_i to P'_i.

Sign *queries*: On receiving a Sign query $S(\Delta_i, M_i, ID_i, P_i)$, $\mathscr{C}$ first makes $H_1(ID_i, 0), H_1(ID_i, 1)$, $H_2(\Delta_i)$, $H_3(\Delta_i)$, $H_4(\Delta_i)$ and $H_5(M_i \| \Delta_i \| ID_i \| P_i)$ queries if they are not queried before, then recovers $(ID_i, \alpha_{i,0}, \alpha'_{i,0}, \alpha_{i,1}, \alpha'_{i,1}, Q_{i,0}, Q_{i,1})$ from $\mathbf{H}_1$, (Δ_i, T_i, β_i) from $\mathbf{H}_2$, $(\Delta_i, V_i, \gamma_i)$ from $\mathbf{H}_3$, (Δ_i, W_i, π_i) from $\mathbf{H}_4$, $(M_i, \Delta_i, ID_i, P_i, h_i)$ from $\mathbf{H}_5$ and generates the signature as follows.

(1) If $ID_i = ID_I$, $\Delta_i = \Delta_J$, and $h_i = -\alpha'_{i,0}/\alpha'_{i,1}$, choose $R_i \in G_1^*$, compute $S_i = \beta_i R_i + \gamma_i P_i + \pi_i h_i P_i + \alpha_{i,0} P_T + \alpha_{i,1} h_i P_T$, output $\sigma_i = (R_i, S_i)$.
(2) Else if $ID_i = ID_I$, $\Delta_i = \Delta_J$, abort.
(3) Else if $ID_i = ID_I$, choose $r_i \in Z_q^*$, set $R_i = r_i P - \beta_i^{-1}(Q_{i,0} + h_i Q_{i,1})$ compute $S_i = \gamma_i P_i + \pi_i h_i P_i + r_i T_i$, output $\sigma_i = (R_i, S_i)$.
(4) Else, randomly choose $r_i \in Z_q^*$, compute $R_i = r_i P$, set $S_i = \alpha_{i,0} P_T + h_i \alpha_{i,1} P_T + \gamma_i P_i + h_i \pi_i P_i + r_i T_i$, output $\sigma_i = (R_i, S_i)$.

Note that in our CLAS scheme Δ_i is only for one-time use. Hence, it is reasonable for $\mathscr{C}$ to abort when $ID_i = ID_I$, $\Delta_i = \Delta_J$ and $h_i \neq -\alpha'_{i,0}/\alpha'_{i,1}$.

Forgery: Eventually, $\mathscr{A}$ returns a set of n users, whose identities form the set $L_{ID}^* = \{ID_1^*, \ldots, ID_n^*\}$ and corresponding public keys form the set $L_{PK}^* = \{P_1^*, \ldots, P_n^*\}$; n messages form the set $L_M^* = \{M_1^*, \ldots, M_n^*\}$; a state information Δ^* and a forged aggregate signature $\sigma^* = (R^*, V^*)$.

$\mathscr{C}$ recovers (Δ^*, T^*, β^*) from $\mathbf{H}_2$, $(\Delta^*, V^*, \gamma^*)$ from $\mathbf{H}_3$, (Δ^*, W^*, π^*) from $\mathbf{H}_4$ and the tuples $(ID_i^*, \alpha_{i,0}^*, \alpha_{i,0}'^*, \alpha_{i,1}^*, \alpha_{i,1}'^*, Q_{i,0}^*, Q_{i,1}^*)$ from $\mathbf{H}_1$, $(M_i^*, \Delta^*, ID_i^*, P_i^*, h_i^*)$ from $\mathbf{H}_5$ for all $i, 1 \leqslant i \leqslant n$.

It requires that $\Delta^{ast} = \Delta_J$ and there exists $i \in \{1, \ldots, n\}$ such that $ID_i^* = ID_I$, $h_i^* \neq -\alpha_{i,0}'^*/\alpha_{i,1}'^*$ and $\mathscr{A}$ has not made a $S(M_i^*, \Delta^*, ID_i^*, *)$ query. Without loss of generality, we let $i = 1$. In addition, the forged aggregate signature must satisfy

$$e(S^*, P) = e\left(P_T, \sum_{i=1}^{n} Q_{i,0}^* + \sum_{i=1}^{n} h_i^* Q_{i,1}^*\right) e\left(V^*, \sum_{i=1}^{n} P_i^*\right) \times e\left(W^*, \sum_{i=1}^{n} h_i^* P_i^*\right) e(T^*, R^*).$$

Otherwise, $\mathscr{C}$ aborts.

If $\mathscr{C}$ does not abort, from the above equation, we have

$$e\left(P_T, Q_{1,0}^* + h_1^* Q_{1,1}^*\right) = e(S^*, P) \left(e\left(P_T, \sum_{i=2}^{n} Q_{i,0}^* + \sum_{i=2}^{n} h_i^* Q_{i,1}^*\right) \times e\left(V^*, \sum_{i=1}^{n} P_i^*\right) e\left(W^*, \sum_{i=1}^{n} h_i^* P_i^*\right) e(T^*, R^*) \right)^{-1}.$$

And by our setting, $Q_{1,0}^* = \alpha_{1,0}^* P + \alpha_{1,0}'^* bP$, $Q_{1,1}^* = \alpha_{1,1}^* P + \alpha_{1,1}'^* bP$, $T^* = \beta^* P$, $V^* = \gamma^* P$, $W^* = \pi^* P$; and for i, $2 \leqslant i \leqslant n$, $Q_{i,j}^* = \alpha_{i,j}^* P$, where $j \in \{0,1\}$; hence, $\mathscr{C}$ can compute

$$abP = \left(\alpha_{1,0}'^* + h_1^* \alpha_{1,1}'^*\right)^{-1} \left(S^* - \sum_{i=2}^{n} \alpha_{1,0}^* P_T - \sum_{i=2}^{n} \alpha_{1,1}^* h_i^* P_T - \gamma^* \sum_{i=1}^{n} P_i^* - \pi^* \sum_{i=1}^{n} h_i^* P_i^* - \beta^* R^* - \left(\alpha_{1,0}^* + h_1^* \alpha_{1,1}^*\right) P_T \right).$$

To complete the proof, we shall show that $\mathscr{C}$ solves the given instance of CDH problem with probability at least ε'. First, we analyze the four events needed for $\mathscr{C}$ to succeed:

- $\Sigma 1$: $\mathscr{C}$ does not abort as a result of any of $\mathscr{A}$'s Partial-Private-Key queries.
- $\Sigma 2$: $\mathscr{C}$ does not abort as a result of any of $\mathscr{A}$'s signature queries.
- $\Sigma 3$: $\mathscr{A}$ generates a valid and nontrivial aggregate signature forgery.
- $\Sigma 4$: Event $\Sigma 3$ occurs, $\Delta^* = \Delta_J$ and there exists $i \in \{1, \ldots, n\}$ such that $ID_i^* = ID_I$, $h_i^* \neq -\alpha_{i,0}'^*/\alpha_{i,1}'^*$ (as mentioned previously, we assume $i = 1$).

$\mathscr{C}$ succeeds if all of these events happen. The probability $\Pr[\Sigma 1 \wedge \Sigma 2 \wedge \Sigma 3 \wedge \Sigma 4]$ can be decomposed as

$$\Pr[\Sigma 1 \wedge \Sigma 2 \wedge \Sigma 3 \wedge \Sigma 4] = \Pr[\Sigma 1] \Pr[\Sigma 2 | \Sigma 1] \Pr[\Sigma 3 | \Sigma 1 \wedge \Sigma 2] \Pr[\Sigma 4 | \Sigma 1 \wedge \Sigma 2 \wedge \Sigma 3].$$

Claim 1. *The probability that $\mathscr{C}$ does not abort as a result of $\mathscr{A}$'s key extraction queries is at least $(1 - \frac{1}{q_{H_1}})^{q_K}$. Hence we have*

$$Pr[\Sigma 1] \geqslant \left(1-\frac{1}{q_{H_1}}\right)^{q_K}.$$

Proof. For a Partial-Private-Key query, $\mathscr{C}$ will abort iff the query is on $ID_1^* = ID_I$. It is easy to see that the probability $\mathscr{C}$ does not abort for a Partial-Private-Key query is $1-\frac{1}{q_{H_1}}$. Since $\mathscr{A}$ can make at most q_K times Partial-Private-Key queries, the probability that $\mathscr{C}$ does not abort as a result of $\mathscr{A}$'s Partial-Private-Key queries is at least $\left(1-\frac{1}{q_{H_1}}\right)^{q_K}$. □

Claim 2. *The probability that $\mathscr{C}$ does not abort as a result of $\mathscr{A}$'s signature queries is at least* $\left(1-\frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)\right)^{q_S}$. *Thus there hold*

$$\Pr[\Sigma 2|\Sigma 1] \geqslant \left(1-\frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)\right)^{q_S}.$$

Proof. When $\mathscr{C}$ receives a Sign query, he will abort iff $ID_i = ID_I, \Delta_i = \Delta_J$ and $h_i \neq -\alpha'_{i,0}/\alpha'_{i,1}$ happens. So for a Sign query, the probability that $\mathscr{C}$ does not abort is $1-\frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)$. Since $\mathscr{A}$ makes at most q_S times Sign queries, the probability that $\mathscr{C}$ does not abort as a result of$A$'s Sign queries is at least $\left(1-\frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)\right)^{q_S}$. □

Claim 3. $\Pr[\Sigma 3|\Sigma 1 \wedge \Sigma 2] \geqslant \varepsilon$.

Proof. If algorithm $\mathscr{C}$ does not abort as a result of $\mathscr{A}$'s signature queries and key extraction queries then algorithm $\mathscr{A}$'s view is identical to its view in the real attack. Hence, $\Pr[\Sigma 3|\Sigma 1 \wedge \Sigma 2] \geqslant \varepsilon$. □

Claim 4. *The probability that $\mathscr{C}$ does not abort after $\mathscr{A}$ outputting a valid and nontrivial forgery is at least* $\frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)$. *Hence*

$$\Pr[\Sigma 4|\Sigma 1 \wedge \Sigma 2 \wedge \Sigma 3] \geqslant \frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right).$$

Proof. Events $\Sigma 1$, $\Sigma 2$ and $\Sigma 3$ have occurred, $\mathscr{C}$ will abort unless $\mathscr{A}$ generates a forgery such that $ID_1^* = ID_I, \Delta^* = \Delta_J$ and $h_1^* \neq -\alpha'^*_{1,0}/\alpha'^*_{1,1}$. Therefore, $\Pr[\Sigma 3|\Sigma 1 \wedge \Sigma 2] \geqslant \frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)$. □

Totally, we have

$$\begin{aligned}\varepsilon' &= \Pr[\Sigma 1 \wedge \Sigma 2 \wedge \Sigma 3 \wedge \Sigma 4] \\ &\geqslant \left(1-\frac{1}{q_{H_1}}\right)^{q_K}\left(1-\frac{1}{q_{H_1}}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)\right)^{q_S}\frac{1}{q_{H_1}} \\ &\quad\times\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)\varepsilon. \quad \square\end{aligned}$$

Theorem 2. *In the random oracle model, if there exists a type II adversary $\mathscr{A}$ has an advantage ε in forging a signature of our CLAS scheme in an attack modeled by Game 2, within a time span t for a security parameter ℓ; then the CDH problem in G_1 can be solved within time $t+(q_{H_2}+2q_{H_3}+2q_{H_4}+q_P+6q_S)\tau_{G_1}$ and with probability*

$$\varepsilon' \geqslant \left(1-\frac{1}{q_{H_P}}\right)^{q_K}\left(1-\frac{1}{q_P}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)\right)^{q_S}\frac{1}{q_P}\frac{1}{q_{H_2}}\left(1-\frac{1}{q_{H'_5}}\right)\varepsilon.$$

Proof. Let $\mathscr{C}$ be a CDH attacker who receives a random instance (P, aP, bP) of the CDH problem in G_1 and has to compute the value of abP. $\mathscr{A}$ is a type II adversary who interacts with $\mathscr{C}$ as defined in Game 2. We show how $\mathscr{C}$ can use $\mathscr{A}$ to solve the CDH problem, i.e. to compute abP.

Setup: Firstly, $\mathscr{C}$ selects a random $\lambda \in Z_q^*$ as the master-key, computes $P_T = \lambda P$, then selects the system parameters params $= (G_1, G_2, e, P, P_T, H_1 \sim H_5)$. When the simulation is started, $\mathscr{A}$ is provided with params and the master-key λ. Since $\mathscr{A}$ has access to the master-key, he can do Partial-Private-Key-Extract himself.

Attack: We assume $\mathscr{A}$ asking at most q_{H_i} times H_i $(i = 2, \ldots, 5)$ queries, q_P times Public-Key queries, q_K times Secret-Key queries, q_S times Sign queries. $\mathscr{A}$ can perform the following types of queries in an adaptive manner. Note that we need not model the hash function H_1 as a random oracle in this case.

H_2 queries: $\mathscr{C}$ keeps a initially empty list $\mathbf{H}_2$ of tuples (Δ_i, T_i, β_i), randomly selects $I \in [1, q_{H_2}]$. Whenever $\mathscr{A}$ issues a query $H_2(\Delta_i)$, the same answer from the list $\mathbf{H}_2$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ selects a random $\beta_i \in Z_q^*$, if $i = I$, computes $T_i = \beta_i P$, else sets $T_i = \beta_i aP$. Finally, $\mathscr{C}$ adds (Δ_i, T_i, β_i) to $\mathbf{H}_2$ and returns T_i as answer.

H_3 queries: $\mathscr{C}$ keeps a list $\mathbf{H}_3$ of tuples $(\Delta_i, V_i, \gamma_i, \gamma'_i)$. This list is initially empty. Whenever $\mathscr{A}$ issues a query Δ_i to H_3, the same answer from the list $\mathbf{H}_3$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ randomly selects $\gamma_i, \gamma'_i \in Z_q^*$, computes $V_i = \gamma_i P + \gamma'_i aP$, adds $(\Delta_i, V_i, \gamma_i, \gamma'_i)$ to $\mathbf{H}_3$ and returns V_i as answer.

H_4 queries: $\mathscr{C}$ keeps a list $\mathbf{H}_4$ of tuples $(\Delta_i, W_i, \pi_i, \pi'_i)$. This list is initially empty. Whenever $\mathscr{A}$ issues a query Δ_i to H_3, the same answer from the list $\mathbf{H}_4$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ randomly selects $\pi_i, \pi'_i \in Z_q^*$, computes $W_i = \pi_i P + \pi'_i aP$, adds $(\Delta_i, W_i, \pi_i, \pi'_i)$ to $\mathbf{H}_4$ and returns W_i as answer.

Public-Key *queries*: $\mathscr{C}$ keeps a initially empty list $\mathbf{K}$ of tuples (ID_i, x_i, P_i), chooses $J \in [1, q_P]$. On receiving a query $PK(ID_i)$, the same answer from the list $\mathbf{K}$ will be given if the request has been asked before. Otherwise, $\mathscr{C}$ selects $x_i \in Z_q^*$ and dose the following.

(1) If $i = J$, set $P_i = x_i bP$, add (ID_i, x_i, P_i) to $\mathbf{K}$ and return P_i as answer.
(2) Else, compute $P_i = x_i P$, add (ID_i, x_i, P_i) to $\mathbf{K}$ and return P_i as answer.

H_5 queries: $\mathscr{C}$ keeps a list $\mathbf{H}_5$ of tuples $(M_i, \Delta_i, ID_i, P_i, h_i)$. This list is initially empty. Whenever $\mathscr{A}$ issues a query $(M_i \| \Delta_i \| ID_i \| P_i)$ to H_5, the same answer from the list $\mathbf{H}_5$ will

be given if the request has been asked before. Otherwise, $\mathscr{C}$ first makes $H_2(\Delta_i)$, $PK(ID_i)$ then finds $(\Delta_i, T_i, \beta_i, c_i)$ on $\mathbf{H}_2$, (ID_i, x_i, P_i, c_i') on $\mathbf{K}$; finally does the following:

(1) If $\Delta_i = \Delta_I$ and $P_i = P_J$ (we assume that $\mathscr{A}$ can ask at most $q_{H_5'} < q_{H_5}$ times such kind of queries), randomly choose $K \in [1, q_{H_5'}]$.
   (a) If it is the K-th query, set $h_i = -\gamma_i'/\pi_i'$, add $(M_i, \Delta_i, ID_i, P_i, h_i)$ to $\mathbf{H}_5$ and return h_i as answer.
   (b) Else select a random $h_i \in Z_q^*$, add $(M_i, \Delta_i, ID_i, P_i, h_i)$ to $\mathbf{H}_5$ and return h_i as answer.
(2) Else select a random $h_i \in Z_q^*$, add $(M_i, \Delta_i, ID_i, P_i, h_i)$ to $\mathbf{H}_5$ and return h_i as answer.

Secret-Key *queries*: On receiving a query $SK(ID_i)$, $\mathscr{C}$ first makes $PK(ID_i)$ then recovers the tuple (ID_i, x_i, P_i) from $\mathbf{K}$. If $ID_i = ID_J$, $\mathscr{C}$ aborts; otherwise, returns x_i as answer.

Sign *queries*: On receiving a Sign query $S(M_i, \Delta_i, ID_i, P_i)$, $\mathscr{C}$ first makes $H_2(\Delta_i)$, $H_3(\Delta_i)$, $H_4(\Delta_i)$, $H_5(M_i \| \Delta_i \| ID_i \| P_i)$ and $PK(ID_i)$ queries if they are not queried before, then finds the tuples (Δ_i, T_i, β_i) on $\mathbf{H}_2$, $(\Delta_i, V_i, \gamma_i, \gamma_i')$ on $\mathbf{H}_3$, $(\Delta_i, W_i, \pi_i, \pi_i')$ on $\mathbf{H}_4$, $(M_i, \Delta_i, ID_i, P_i, h_i)$ on $\mathbf{H}_5$, (ID_i, x_i, P_i) on $\mathbf{K}$ and generates the signature as follows:

(1) If $\Delta_i = \Delta_I, P_i = P_J$ and $h_i = -\gamma_i'/\pi_i'$, randomly choose $R_i \in G_1^*$, compute $S_i = \lambda\ (H_1(ID_i, 0) + h_i H_1(ID_i, 1)) + \gamma_i P_i + \pi_i h_i P_i + \beta_i R$.
(2) Else if $\Delta_i = \Delta_I$, $P_i = P_J$, abort.
(3) Else if $P_i = P_J$, choose $r_i \in Z_q^*$, set $R_i = r_i P - \beta_i^{-1}(\gamma_i' + \pi_i' h_i) P_i$, compute $S_i = r_i T_i + (\gamma_i + \pi_i h_i) P_i + \lambda(H_1(ID_i, 0) + h_i H_1(ID_i, 1))$, output $\sigma_i = (R_i, S_i)$.
(4) Else randomly choose $r_i \in Z_q^*$, compute $R_i = r_i P$, compute $S_i = \lambda(H_1(ID_i, 0) + h_i\ H_1(ID_i, 1)) + x_i V + h_i x_i W + r_i T_i$, output $\sigma_i = (R_i, S_i)$.

Forgery: Finally, $\mathscr{A}$ returns a set of n users, whose identities form the set $L_{ID}^* = \{ID_1^*, \ldots, ID_n^*\}$ and corresponding public keys form the set $L_{PK}^* = \{P_1^*, \ldots, P_n^*\}$; n messages form the set $L_M^* = \{M_1^*, \ldots, M_n^*\}$; a state information Δ^* and a forged aggregate signature $\sigma^* = (R^*, S^*)$.

$\mathscr{C}$ recovers (Δ^*, T^*, β^*) from $\mathbf{H}_2$, $(\Delta^*, V^*, \gamma^{ast}, \gamma'^*)$ from $\mathbf{H}_3$, $(\Delta^*, W^*, \pi^{ast}, \pi'^*)$ from $\mathbf{H}_4$ and the tuples $(M_i^*, \Delta^*, ID_i^*, P_i^*, h_i^*)$ from $\mathbf{H}_5$, (ID_i^*, x_i^*, P_i^*) from $\mathbf{K}$ for all $i, 1 \leqslant i \leqslant n$.

It requires that $\Delta^{ast} = \Delta_I$ and there exists $i \in \{1, \ldots, n\}$ such that $P_i^* = P_J$, $h_i^* \neq -\gamma_i'^*/\pi_i'^*$ and $S(\Delta^*, M_i^*, ID_i^*, *)$ has never been queried. Without loss of generality, we let $i = 1$. In addition, the forged aggregate signature must satisfy

$$e(S^*, P) = e\left(P_T, \sum_{i=1}^{n} Q_{i,0}^* + \sum_{i=1}^{n} h_i^* Q_{i,1}^*\right) e\left(V^*, \sum_{i=1}^{n} P_i^*\right) \times e\left(W^*, \sum_{i=1}^{n} h_i^* P_i^*\right) e(T^*, R^*),$$

where $Q_{i,j}^* = H_1(ID_i^*, j), j \in \{0, 1\}$. Otherwise, $\mathscr{C}$ aborts.

If $\mathscr{C}$ does not aborts, we have

$$abP = \left(\gamma'^* x_1^* + \pi^* h_1^* x_1^*\right)^{-1} \left(S^* - \lambda\left(\sum_{i=1}^{n} Q_{i,0} + \sum_{i=1}^{n} h_i^* Q_{i,1}^*\right) - \sum_{i=2}^{n} x_i^* V^* - \sum_{i=2}^{n} h_i^* x_i^* W^* - \beta^* R^* - \gamma^* P_1^* - \pi^* h_1^* P_1^*\right).$$

Now we determine the probability ε' for $\mathscr{C}$ to solve the given instance of CDH problem. We analyze the four events needed for $\mathscr{C}$ to succeed:

- $\Sigma 5$: $\mathscr{C}$ does not abort as a result of any of $\mathscr{A}$'s Secret-Key queries.
- $\Sigma 6$: $\mathscr{C}$ does not abort as a result of any of $\mathscr{A}$'s signature queries.
- $\Sigma 7$: $\mathscr{A}$ generates a valid and nontrivial aggregate signature forgery.
- $\Sigma 8$: Event $\Sigma 7$ occurs, $\Delta^* = \Delta_I, P_1^* = P_J, (h_1^* \neq -\gamma_1'^*/\pi_1'^*$.

$\mathscr{C}$ succeeds if all of these events happen. The probability $\Pr[\Sigma 5 \wedge \Sigma 6 \wedge \Sigma 7 \wedge \Sigma 8]$ can be decomposed as

$$\Pr[\Sigma 5 \wedge \Sigma 6 \wedge \Sigma 7 \wedge \Sigma 8] = \Pr[\Sigma 5]\Pr[\Sigma 6|\Sigma 5]\Pr[\Sigma 7|\Sigma 5 \wedge \Sigma 6]\Pr[\Sigma 8|\Sigma 5 \wedge \Sigma 6 \wedge \Sigma 7].$$

Similar to Theorem 1, we have

$$\begin{cases} \Pr[\Sigma 5] \geqslant \left(1 - \frac{1}{q_P}\right)^{q_K} \\ \Pr[\Sigma 6|\Sigma 5] \geqslant \left(1 - \frac{1}{q_P}\frac{1}{q_{H_2}}(1 - q_{H_5'})\right)^{q_S} \\ \Pr[\Sigma 7|\Sigma 5 \wedge \Sigma 6] \geqslant \varepsilon \\ \Pr[\Sigma 8|\Sigma 5 \wedge \Sigma 6 \wedge \Sigma 7] \geqslant \frac{1}{q_P}\frac{1}{q_{H_2}}(1 - q_{H_5'}) \end{cases}$$

Finally, we have

$$\varepsilon' = \Pr[\Sigma 5 \wedge \Sigma 6 \wedge \Sigma 7 \wedge \Sigma 8] \geqslant \left(1 - \frac{1}{q_{H_P}}\right)^{q_K} \left(1 - \frac{1}{q_P}\frac{1}{q_{H_2}}\left(1 - \frac{1}{q_{H_5'}}\right)\right)^{q_S} \frac{1}{q_P} \times \frac{1}{q_{H_2}}\left(1 - \frac{1}{q_{H_5'}}\right)\varepsilon. \quad \square$$

5. Comparison

In this section we compare our scheme with the schemes in [9,21]. Firstly, we list some costly operations, i.e. Pairing Operation ($\mathbb{P}$), Scalar Multiplication in G_1 ($\mathbb{S}$) and MapToPoint Hash ($\mathbb{H}$). Among those operations, Pairing Operation is the most time consuming one. We use the notation SL meaning signature length, PKL meaning public key length, $\mathbb{P}_1$ meaning the length of a point in G_1. And we omit the cost of the operations which can be pre-computed by the signer such as $H_1(ID_i j)$ etc (see Table 1).

From the table, the Sign procedure of our scheme is slightly less efficient than that of the schemes in [9,21]. However, our Aggregate Verify procedure is much more effi-

Table 1
Comparison of three CLS schemes.

Schemes	Sign	Aggregate Verify	SL	PKL
First scheme in [9]	$2\mathbb{S}, \mathbb{H}$	$(4n+1)\mathbb{P}, 2n\mathbb{H}$	$(n+1)\mathbb{P}_1$	$2\mathbb{P}_1$
Second scheme in [9]	$3\mathbb{S}, \mathbb{H}$	$(3n+2)\mathbb{P}, n\mathbb{S}, 3n\mathbb{H}$	$2\mathbb{P}_1$	$2\mathbb{P}_1$
Scheme in [21]	$3\mathbb{S}, 2\mathbb{H}$	$(n+3)\mathbb{P}, (2n+1)\mathbb{H}$	$(n+1)\mathbb{P}_1$	$1\mathbb{P}_1$
Our scheme	$5\mathbb{S}, 3\mathbb{H}$	$5\mathbb{P}, 2n\mathbb{S}, (2n+3)\mathbb{H}$	$2\mathbb{P}_1$	$1\mathbb{P}_1$

cient than those of the schemes in [9,21]. For application purposes the practicality of an aggregate signature scheme is mainly dominated by Aggregate Verify algorithm. This is due to the fact that the verifier must verify n different signatures distributively generated by each users. This implies that our scheme may enjoy better practicality.

As for the signature size, our signature requires only two elements in G_1 and approximately 320 bits. Note that in CL-PKC a signer has to send the signature together with the corresponding public key to a verifier for a verification. One can see that our scheme is the most bandwidth-saving one among the three schemes, observing that the public key of a user in our scheme is only one element in G_1.

6. Conclusion

We presented an efficient certificateless aggregate signature scheme. The proposal is proven existentially unforgeable against adaptively chosen-message attack in the random oracle model assuming that the CDH problem is hard. Our CLAS scheme can be applied to authentications in bandwidth-limited scenarios such as many-to-one communications.

Acknowledgments

This work is supported by the Spanish Government through Projects TSI2007-65406-C03-01 "E-AEGIS" and CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES", and by the Government of Catalonia under Grant 2009 SGR 1135, and by the Chinese NSF Projects 60673070, 60970114, 60970115 and 60970116. The views of the author with the UNESCO Chair in Data Privacy do not necessarily reflect the position of UNESCO nor commit that organization.

References

[1] C.K. Miller, Multicast Networking and Applications, Addison Wesley, Reading, MA, 1999.
[2] S. Al-Riyami, K. Paterson, Certificateless public key cryptography, in: ASIACRYPT 2003, LNCS, vol. 2894, 2003, pp. 452–473.
[3] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: ACM CCCS '93, 1993, pp. 62–73.
[4] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, in: EUROCRYPT 2003, LNCS, vol. 2656, 2003, pp. 416–432.
[5] D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing, in: Asiacrypt 2001, LNCS, vol. 2248, 2001, pp. 514–532.
[6] X. Cheng, J. Liu, X. Wang. Identity-based aggregate and verifiably encrypted signatures from bilinear pairing, in: ICCSA 2005, LNCS, vol. 3483, 2005, pp. 1046–1054.
[7] J. Cheon, Y. Kim, H. Yoon, A new id-based signature with batch verification, Cryptology ePrint Archive, Report 2004/131 2006.
[8] C. Gentry, Z. Ramzan, Identity-based aggregate signatures, in: PKC 2006, LNCS, vol. 3958, 2006, pp. 257–273.
[9] Z. Gong, Y. Long, X. Hong, K. Chen, Two certificateless aggregate signatures from bilinear maps, in: IEEE SNPD 2007, vol. 3, 2007, pp. 188–193, <http://eprints.eemcs.utwente.nl/14802/02/Practical_Certificateless_Aggregate_Signatures_From_Bilinear_Maps.pdf>.
[10] J. Herranz, Deterministic identity-based aggregate signatures for partial aggregation, The Computer Journal 49 (3) (2006) 322–330.
[11] B. Hu, D. Wong, Z. Zhang, X. Deng, Key replacement attack against a generic construction of certificateless signature, in: ACISP 2006, LNCS, vol. 4058, 2006, pp. 235–346.
[12] X. Huang, W. Susilo, Y. Mu, F. Zhang, On the security of a certificateless signature scheme, in: CANS 2005, LNCS, vol. 3810, 2005, pp. 13–25.
[13] X. Huang, Y. Mu, W. Susilo, D. Wong, and W. Wu. Certificateless signature revisited. ACISP 2007, LNCS 4586, 2007, pp. 308–322.
[14] J. Liu, M. Au, W. Susilo, Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model, in: ACM ASIACCS'07, 2007.
[15] J. Park, An attack on the certificateless signature scheme from EUC Workshops 2006, Cryptology ePrint Archive, Report 2006/442, 2006.
[16] A. Shamir, Identity based cryptosystems and signature schemes, in: Crypto'84, LNCS, vol. 196, 1984, pp. 47–53.
[17] J. Xu, Z. Zhang, D. Feng, ID-based aggregate signatures from bilinear pairings, in: CANS 2005, LNCS, vol. 3810, 2005, pp. 110–119.
[18] W. Yap, S. Heng, B. Goi, An efficient certificateless signature scheme, in: EUC Workshops 2006, LNCS, vol. 4097, 2006, pp. 322–331.
[19] Z. Zhang, D. Feng, Key replacement attack on a certificateless signature scheme, Cryptology ePrint Archive, Report 2006/453, 2006.
[20] Z. Zhang, D. Wong, J. Xu, D. Feng, Certificateless public-key signature: security model and efficient construction, in: ACNS 2006, LNCS, vol. 3989, 2006, pp. 293–308.
[21] L. Zhang, F. Zhang, A New Certificateless Aggregate Signature Scheme, Computer Communications (2009), doi:10.1016/j.comcom.2008.12.042.
[22] L. Zhang, F. Zhang, A new provably secure certificateless signature scheme, in: IEEE ICC'08, 2008, pp. 1685–1689.

Lei Zhang is currently a PhD candidate in the Department of Computer Engineering and Mathematics at Universitat Rovira i Virgili of Tarragona, Catalonia. His research interests include public key cryptography, network security and information security. He is authored/coauthored over 20 publications.

Bo Qin obtained her PhD from Xidian University in 2008. She is now with Universitat Rovira i Virgili as postdoctoral researcher in Catalonia. Her research interests include cryptography, data privacy, and network security. She has been a holder/coholder of five R+D funds from Spanish/Chinese government, and authored/coauthored over 30 publications in information security.

Qianhong Wu has been with University of Wollongong as associate research fellow in Australia, Wuhan University as associate professor in China, and now with Universitat Rovira i Virgili as senior researcher in Catalonia. His research interests include cryptography, information security and privacy, and ad hoc networks security. He has been a holder/coholder of six R+D funds from Spanish/Chinese/Australian government, and authored over 60 publications. He served in the program committee of several international conferences on information security and privacy. Dr. Qianhong Wu is a member of the International Association for Cryptologic Research (IACR).

Futai Zhang is a professor in School of Computer Science and Technology at Nanjing Normal University, Nanjing, China. His research interests include information security, network security and cryptography.