Practical Certificateless Aggregate Signatures from Bilinear Maps*

ZHENG GONG¹, YU LONG¹, XUAN HONG² AND KEFEI CHEN²,³

¹Distributed and Embedded Security Group, Faculty of EEMCS, University of Twente, Enschede, The Netherlands.
²Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai, China.
³National Laboratory of Modern Communications, Chengdu, China.

Abstract

Aggregate signature is a digital signature with a striking property that anyone can aggregate \( n \) individual signatures on \( n \) different messages which are signed by \( n \) distinct signers, into a single compact signature to reduce computational and storage costs. In this work, two practical certificateless aggregate signature schemes are proposed from bilinear maps. The first scheme \( \text{CAS-1} \) reduces the costs of communication and signer-side computation but trades off the storage, while \( \text{CAS-2} \) minimizes the storage but sacrifices the communication costs. One can choose either of the schemes by consideration of the application requirement. Compare with ID-based schemes, our schemes do not entail public key certificates as well and achieve the trust level 3, which imply the frauds of the authority are detectable. Both of the schemes are proven secure in the random oracle model by assuming the intractability of the computational Diffie-Hellman problem over the groups with bilinear maps, where the forking lemma technique is avoided.

Keywords: Authentication, Aggregate signature, Certificateless.

1. INTRODUCTION

In nowadays network applications, digital signatures are widely used to provide the properties of authenticity and integrity on the messages. A well-designed signature must be as short as possible to save communication bandwidth and storage. In some special applications, such as sensor nets which are used to detect the status in danger environments, each sensor submits the newest information to the server, we need the information is signed by the sensor to immune forgery and replay attacks which may trigger a false alarm or conceal a real danger. But this promotion will heavily add the costs of the computation and communication in the system. From the statistics in [2, 11], each transmitted

* A preliminary version of this paper appears in SNPD 2007, IEEE Computer Society Proceedings. This is the full version. The first author acknowledges the financial support of SenterNovem for the ALwEN project, grant PNE07007. The authors are partially supported by NSFC (No.60803146), National 863 Projects of China (No.2007AA01Z456), National Basic Research Program (973) of China (No.2007CB311201) and the Foundation of NLMC (No.9140C1103020803).
Aggregate Signature. Aggregate signature (AS) scheme, which is first introduced by Boneh et al. in [4], is a digital signature scheme with an additional property that anyone can aggregate $n$ individual signatures (a sequence $\sigma_1, \sigma_2, \ldots, \sigma_n$) on $n$ different messages $(m_1, m_2, \ldots, m_n)$ which are signed by $n$ distinct signers, into a single compact signature $\sigma$. For all $i \in \{1, 2, \ldots, n\}$, every individual signature $(m_i, \sigma_i)$ can check its correctness by using the corresponding public key $pk_i$. There exists an algorithm called aggregate verification that takes inputs $\{\sigma, (pk_i, m_i)|i=1, 2, \ldots, n\}$ then returns if the compact signature $\sigma$ is valid or not. Aggregation property is promising to reduce the costs on computation, communication and storage. Consider some special environments, such as PDAs, cell phones and sensors, the limitation of battery life is more restricted than the processor speed. An aggregate signature will be applicable under those constrained scenarios. Aside from efficiency, aggregate signatures have other advantages. E.g., certificate chains in a hierarchical PKI of depth $n$ consist of $n$ signatures by $n$ different CAs on $n$ different public keys; by using an aggregate signature scheme, this chain can be compressed down to a single aggregate signature. Another example is secure routing. In Secure BGP [13], each router continuously signs its segment of a path in the topology, and forwards the collection of signatures associated with the path to the next router; forwarding these signatures requires a high transmission overhead that could also be reduced by using an aggregate signature scheme [8]. We note that digital signature with batch verification [6] goes similar on reducing the computational costs of verification, but it lacks of the compactness such that anyone should be able to compress $n$ distinct individual signatures into a single aggregate signature.

Certificateless Signature. To solve key escrow problem while maintain the advantages of identity-based public key cryptosystem (ID-PKC) [6, 15], Al-Riyami and Paterson introduced a new certificateless public key cryptosystem (CL-PKC) in [1]. In contrast to directory-based public key system (DB-PKC), the user's public key does not entail any certificate to authenticate its validity since it can be self-certificated. In ID-PKC, there exists a third party called the Private Key Generator (PKG) has to be completely trusted, because PKG is so powerful that it obtains the knowledge of all users' secret keys. In CL-PKC, there exists a less powerful trusted third party, which is called Key Generate Center (KGC). KGC has the master-key to generate a user's partial private key $D_s$, which is computed from the user's identity $ID_i$. The partial private key should be securely sent to the user. Afterward, the user adds his private secret information into the received partial private key, then derives his full private key $S_i$. Correspondingly, the user combines his secret information with the KGC's public key to generate his own public key. In this sense, KGC knows nothing about the user's private key, which means key escrow problem does not exist anymore in certificateless cryptosystems.

We note that three trust levels of certificate authorities are defined in [9]. In Level 1, the authority knows (or can easily compute) users’ secret keys and can impersonate any user at any time without being detected. The authority of Level 2 does not know (or can-
not easily compute) users’ secret keys. Nevertheless, the authority can still impersonate a user by generating false guarantees (certificates). In Level 3, any fraud of the authority is detectable. After the intuitive work [1], many certificateless public key signature (CL-PKS) schemes are proposed, such as [12, 18]. The advantages of CL-PKC make them more competitive and feasible in many practical applications.

Table 1. The comparison of the public key cryptosystems

<table>
<thead>
<tr>
<th>Cryptosystem</th>
<th>DB-PKC</th>
<th>ID-PKC</th>
<th>CL-PKC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trust Level</td>
<td>Level 3</td>
<td>Level 1</td>
<td>Level 3</td>
</tr>
<tr>
<td>Key Distribution Channel</td>
<td>Authentic</td>
<td>Authentic and Private</td>
<td>Authentic</td>
</tr>
<tr>
<td>Retrieve Public Key</td>
<td>Directory</td>
<td>Communication</td>
<td>Communication</td>
</tr>
<tr>
<td>Public Key includes ID</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Our Contribution. In this paper, we propose two practical certificateless aggregate signature schemes (CAS) by using bilinear maps. The first scheme CAS-1 reduces the costs of communication and signer-side computation but loses on the storage, while CAS-2 minimizes the storage but sacrifices the communication costs. Users can choose one of the above schemes by the consideration of which advantage is more important in the application. Compare with the traditional PKI-based scheme [4], our schemes do not require public key certificates anymore. According to the CL-PKC, our schemes achieve the trust level 3 [9], which means the frauds of the authority are detectable. In formal, both of the schemes are proven secure in the random oracle model (ROM) by assuming the intractability of the computational Diffie-Hellman (CDH) problem over the groups with bilinear maps, without using the forking lemma technique [14].

Related Work. Boneh et al. first introduced an aggregate signature from bilinear maps in [3], and then in a survey paper [5], Boneh et al. also presented a variant based on [3]. The schemes is very simple, but it has a disadvantage that the verification costs will increase linearly ($O(n)$) with the number of messages ($n$) in the aggregated signature. Subsequent to those initial works, many improved schemes were proposed in the literature, such as [7, 16, 17], some of them are based on ID-PKC. Compares with the certificate-based scheme, identity-based scheme does not need certificates' storage and public key verification anymore. Recently, an efficient identity-based aggregate signature was proposed by Gentry and Ramzan in [8]. Their scheme takes advantage that the aggregate verification requires only three times pairing computation, regardless of the number of messages in the aggregated signature.

2. PRELIMINARIES

In this section, we recall some notions and definitions that will be used in the rest of the paper. Let $a|b$ denote the concatenation of two strings. The same terminology and abbreviations in different definitions are the same meaning, except there are special claims in the context.

2.1 Bilinear Maps

Our schemes use a bilinear map, which is often called “pairing” as well. Let $G_1$ and $G_2$
be two additive cyclic groups with the same prime order $q$. Let $\hat{e}$ be a bilinear map such that $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$. A map $\hat{e}$ has the following properties:

1. Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$, for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$.  
2. Non-degeneracy: $\hat{e}(P, Q) \neq 1_{\mathbb{G}_2}$.  
3. Symmetric: $\hat{e}(P, Q) = \hat{e}(Q, P)$, for all $P, Q \in \mathbb{G}_1$.  
4. Admissible: $\hat{e}(\cdot, \cdot)$ is efficiently computable.

2.2 Computational Assumptions

The security of our schemes is based on the intractability of the computational Diffie-Hellman (CDH) problem.

**Definition 1.** (CDH Problem) Given $(P, aP, bP) \in \mathbb{G}_1$ and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$, compute $abP$ (for unknown randomly chosen $a, b \in \mathbb{Z}_q$).

We say that the CDH problem is $(t, \epsilon)$-hard if there exists no algorithm can solve it in polynomial time at most $t$ with probability no more than $\epsilon$.

2.3 Certificateless Aggregate Signature

Here we give the notion of CAS scheme. Generally speaking, a CAS scheme is defined by eight polynomial time bound algorithms: **Setup**, **Set-Secret-Value**, **Set-Public-Key**, **Partial-Private-Key-Extract**, **Set-Full-Private-Key**, **Sign**, **Agg** and **Ver**. Different from the standard certificateless signature scheme, the public key binding technique in **Partial-Private-Key-Extract** [1] will be used in our schemes. Through the binding, the schemes can achieve the trust level 3 [9], which is the same level as traditional PKI. Thus in our schemes, the forgery of one’s public key by a malicious KGC will be detectable since one public key is binding to one partial private key. Moreover, the private channel between KGC and user is unnecessary (just require an authentic channel). The details of the algorithms are described as follows.

1. **Setup**: This algorithm inputs security parameter $1^k$, then returns the system parameters $\text{params}$ and KGC’s secret value $\text{master-key}$.
2. **Set-Secret-Value**: This algorithm inputs $\text{params}$ and an identifier $\text{ID}_A$, then outputs $A$’s secret value $x_A$.
3. **Set-Public-Key**: This algorithm inputs $\text{params}$ and $x_A$, then outputs $A$’s public key $P_A$.
4. **Partial-Private-Key-Extract**: This algorithm inputs $\text{params}$, $\text{master-key}$, public key $P_A$ and an identifier $\text{ID}_A$ for entity $A$, and then returns a partial private key $D_A$.
5. **Set-Full-Private-Key**: This algorithm inputs $\text{params}$, $D_A$ and $x_A$, then outputs $A$’s full private key $S_A$.
6. **Sign**: This algorithm inputs an accepted message $m$, $\text{params}$, the $i$-th user’s identifier $ID_i$ and the full private key $S_i$, then outputs a signature $\sigma_i$.
7. **Agg**: For $i = 1, 2, ..., n$, inputs $n$ distinct users’ public keys and identifiers $\{(P_i, ID_i)\}_{i = 1, 2, ..., n}$, $n$ distinct messages $\{m_i | i = 1, 2, ..., n\}$ and $n$ individual signatures
\{ \sigma_i \mid i=1, 2, \ldots, n \}, \text{ then outputs a compact signature } \sigma.

8. \textbf{Ver}: For i= 1, 2, \ldots, n, inputs } n \text{ distinct messages } \{m_i \mid i=1, 2, \ldots, n\}, \text{ a compact signature } \sigma, \text{ params, } n \text{ distinct users’ public keys and identifiers } \{(P_i, ID_i) \mid i=1, 2, \ldots, n\}, \text{ then outputs true if the signature } \sigma \text{ is valid, otherwise returns false.}

Due to the notion of aggregate signature [4], a compact signature \( \sigma \) is declared to be valid only if the aggregator who created \( \sigma \) was derived from all valid individual signatures \( \{\sigma_i \mid i=1, 2, \ldots, n\} \). Thus an aggregate signature provides non-repudiation at once on many messages by many users.

2.4 Security Model of Certificateless Aggregate Signature Schemes

In CL-PKC, two types of adversaries with different capabilities [1] are considered. In our security analysis, these adversaries are also imported to simulate the adaptive chosen-message attack. A CAS scheme should be secure against the existential forgery under these adaptive adversaries.

\textbf{TYPE-I Adversary}: This type of adversary \( A_I \) cannot access the KGC’s master-key, but has the ability to replace the public key of any entity, because there are no certificates involved in CL-PKC.

\textbf{TYPE-II Adversary}: This type of adversary \( A_H \) can access the KGC’s master-key, but he has no ability to replace the public key of any entity.

We notice that \( A_I \) acts as a common adaptive forger, while \( A_H \) is defined to model the security against a malicious KGC or adversaries who compromised master-key. According to the different types of the adversary, we define the following games between an adversary \( A \in \{A_I, A_H\} \) and a challenger \( C \).

1. \textbf{Setup}: \( C \) takes a security parameter \( 1^k \) and runs the \textbf{Setup} algorithm, publishes the resulting system parameters \textbf{params}, and then

   - \textbf{Type-I}: \( C \) keeps master-key to himself. For any user ID, \( A_I \) can request a partial private key of the identifier ID, \( C \) responses \( D_{ID} \). \( A_I \) can select a new secret value \( x' \) and compute the corresponding public key \( (X'_{ID}, Y'_{ID}) \). \( C \) will record these replacements \( (x', X'_{ID}, Y'_{ID}) \) as valid.

   - \textbf{Type-II}: \( C \) gives master-key to \( A \).

2. \textbf{Queries}: For any user ID, \( A \) can submit a query to \( C \) on an arbitrary message \( m_i \).
returns the signature $\sigma_i$ which is valid under the user's public key, and records the signed message $m_i$ in the tape $M$.

3. **Response:** After the above experiments, $A$ outputs a valid aggregate signature $\sigma'$, which satisfies $\text{Ver}((pk_1, m_1), \ldots, (pk_n, m_n), \sigma') = \text{true}$ and there exists at least one message $m_i, i \in \{1, 2, \ldots, n\}$ that is not recorded on $C$’s tape $M$.

Therefore, the advantage of adversary $A$ wins the above game can be defined by the following equation.

$$\text{ADV}_{\text{CAS}}^{\text{Agg-CMA}}(A) = \Pr[\text{Ver}((pk_1, m_1), \ldots, (pk_n, m_n), \sigma') = \text{true} | \exists m_i \in M, i \in \{1, \ldots, n\}]$$

Consequently, we give the security definitions on certificateless aggregate signature schemes as follows.

**Definition 2.** An adversary is $(t, \epsilon, n, q_H, q_E, q_S)$-breaks a CAS scheme if: there exist $n$ individual users; $A$ runs in polynomial time at most $t$; $A$ makes at most $q_H$ times hash queries, $q_E$ times partial private key extractions and $q_S$ times signing queries; and $\text{ADV}_{\text{CAS}}^{\text{Agg-CMA}}(A)$ is at least non-negligible $\epsilon$.

**Definition 3.** A CAS scheme is $(t, \epsilon, n, q_H, q_E, q_S)$-secure against existential forgery if there is no adversary $(t, \epsilon, n, q_H, q_E, q_S)$-breaks it.

### 3. TWO CERTIFICATELESS AGGREGATE SIGNATURES FROM BILINEAR MAPS

Here we propose two certificateless aggregate signature schemes based on bilinear maps, which are denoted by CAS-1 and CAS-2 respectively. The first scheme CAS-1 reduces the costs of communication and signer-side computation but trades off the storage, while the second scheme CAS-2 minimizes the storage but sacrifices the communication. Before the detailed descriptions, we recall some basic definitions and notions which will be used in both of the schemes. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two additive cyclic groups with the same prime order $q$. Let $\hat{\cdot}$ be a bilinear map such that $\hat{\cdot} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_2$. Let $n$ be the maximum number of the users in the schemes where $i \in \{1, 2, \ldots, n\}$.

#### 3.1 CAS-1 Scheme
Setup: KGC generates system parameters $\textbf{params}$ and secret value $\textbf{master-key}$ as follows:

1. Randomly selects a generator $P \in \mathbb{G}_1$;
2. Chooses a random value $s \in \mathbb{Z}_q^+$ as the $\textbf{master-key}$, and then computes $Q = sP$;
3. Chooses two cryptographic hash functions $H_1, H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$;
4. Publishes the system parameters $\textbf{params} = \{ \mathbb{G}_1, \mathbb{G}_2, \hat{\mathbf{e}}, P, Q, H_1, H_2 \}$.

Set Secret Value: The $i$-th user randomly chooses a secret random value $x_i \in \mathbb{G}_1$ and securely keeps $x_i$.

Set Public Key: The $i$-th user computes the user’s public key $P_i = (X_i, Y_i)$ where $X_i = x_iP$ and $Y_i = x_iQ$. Anyone can check if $P_i$ is valid by the following equation.

$$\hat{\mathbf{e}}(X_i, Q) = \hat{\mathbf{e}}(Y_i, P_i).$$

Partial Private Key Extract: The $i$-th user sends his identifier $ID_i \in \{0, 1\}^*$ and the public key $P_i$ to KGC, KGC constructs the partial private key $D_i = sH_1(ID_i \| P_i)$.

Set Full Private Key: When received his partial private key $D_i$ from the KGC, the $i$-th user computes $S_i = x_iD_i$ as the full private key.

Sign: Given an arbitrary message $m_i \in \{0, 1\}^*$, the $i$-th user processes the signing algorithm as follows:

1. Selects $r_i \in \mathbb{Z}_q^+$, then computes $U_i = r_iP$;
2. Computs $T_i = H_2(ID_i \| m_i \| U_i)$;
3. Computes $V_i = r_iT_i + S_i$;
4. Outputs $\sigma_i = (U_i, V_i)$ as the signature on $m_i$.

Aggregate: For $n$ individual signatures given by $n$ distinct users where $n = 1, 2, \ldots$, the aggregation goes:

1. parses $\sigma_i$ into $U_i$, $V_i$;
2. computes $\overline{V} = \sum_{i=1}^{n} V_i$;
3. outputs a compact signature $\overline{\sigma} = (U_1, U_2, \ldots, U_n, \overline{V})$.

Verify: Given a signature $\overline{\sigma}$, one can check if the following equation
\[ \hat{e}(P, \tilde{V}) = \prod_{i=1}^{n} \hat{e}(Y_i, H_1(ID_i \Vert |P_i|)) \cdot \prod_{i=1}^{n} \hat{e}(U_i, T_i) \]  

holds, where \( P_i = (X_i, Y_i) \) and \( T_i = H_2(ID_i \Vert |m_i| \Vert U_i) \), and then returns valid or not. The correctness follows:

\[
\hat{e}(P, \tilde{V}) = \hat{e}(P, \sum_{i=1}^{n} r_i T_i) \cdot \hat{e}(P, \sum_{i=1}^{n} S_i) \\
= \hat{e}(Y_i, \sum_{i=1}^{n} H_1(ID_i \Vert |P_i|)) \cdot \hat{e}(P, \sum_{i=1}^{n} r_i T_i) \\
= \prod_{i=1}^{n} \hat{e}(Y_i, H_1(ID_i \Vert |P_i|)) \cdot \prod_{i=1}^{n} \hat{e}(U_i, T_i).
\]

### 3.2 Security Analysis

Under the random oracle model, while assuming the intractability of CDH problem in the groups with bilinear maps, we will prove \( \text{CAS-1} \) is existentially unforgeable in the security model of certificateless aggregate signatures.

**Theorem 1.** If there exists an adversary \( A \) can \((t, \epsilon, n, q_{H_1}, q_{H_2}, q_E, q_S)\)-breaks \( \text{CAS-1} \), then an algorithm \( B \) interacts with \( A \) can be constructed to solve CDH problem in polynomial time bound with a non-negligible probability.

**Proof.** By assuming that \( B \) is given an instance \((q, P, aP, bP)\) of the CDH problem, \( B \) will interact with a Type-I adversary \( A \) to compute the CDH answer \( abP \) in the following steps.

**Setup:** \( B \) first sets the KGC’s params = \( (\hat{e}, \circ_1, \circ_2, \hat{e}, P, Q, H_1, H_2) \) such that the public key \( Q = aP. H_1, H_2 \) are two random oracles controlled by \( B \).

**Hash Queries:** \( A \) can make hash queries at any time. \( B \) maintains a list for each random oracle.

- For \( H_1 \)-query on \((ID_i, P_i)\):
  
  1. If \((ID_i, P_i)\) is queried before, \( B \) retrieves \((k_{i,1}, l_{i,1})\) from \( H_1 \)-list.
  2. Else \( B \) generates \( k_{i,1} \in \mathbb{Z}_q^* \), \( l_{i,1} = 0 \) and \( H_1\text{-coin}_{i} = 0 \) with the probability \( 1 - \frac{1}{q_{H_1}} \); or \( B \) generates \( k_{i,1}, l_{i,1} \in \mathbb{Z}_q^* \) and \( H_1\text{-coin}_{i} = 1 \) with the probability \( \frac{1}{q_{H_1}} \). \( B \) logs \((ID_i, P_i, H_1\text{-coin}_{i}, k_{i,1}, l_{i,1})\) in the \( H_1 \)-list.
3. $B$ responds with $H_1(ID_i \| P_i) = k_{i,1}P + b_l_{i,1}P$.

   - For $H_2$-query on $(ID_n, m_n, U_i)$:
     
     1. If $(ID_n, m_n, U_i)$ is queried before, $B$ retrieves $k_{i,2}$ from $H_2$-list.
     2. Else $B$ chooses $k_{i,2} \in \mathbb{Z}_q^*$, then he logs $(ID_n, m_n, U_i, k_{i,2})$ in the $H_2$-list.
     3. $B$ responds with $H_2(ID_i \| m_i \| U_i) = k_{i,2}P$.

**Partial Private Key Extraction:** For $A$ asks the partial private key for $(ID_n, P_i)$:

   1. If $(ID_n, P_i)$ is queried previously, $B$ retrieves $(H_1\text{-coin}_n, k_{i,1})$ from $H_1$-list.
   2. Else $B$ makes $H_1$-query on $(ID_n, P_i)$.
   3. If $H_1\text{-coin}_n = 0$, $B$ responds $D_i = k_{i,1}Q$. If $H_1\text{-coin}_n = 1$, $B$ aborts.

**Signing Query:** While $A$ requests a signature on $(ID_n, P_i, m_i)$, $B$ retrieves $H_1\text{-coin}_i$ from $H_1$-list. If $H_1\text{-coin}_i = 0$, $B$ processes as follows:

   1. Selects $r_i \in \mathbb{Z}_q^*$, then computes $U_i = r_iP$;
   2. Computes $T_i = H_2(ID_i \| m_i \| U_i) = k_{i,2}P$;
   3. Computes $V_i = r_iT_i + S_i$, $S_i = xk_{i,2}Q = k_{i,1}Y_i$;
   4. Outputs $\sigma_i = (U_i, V_i)$ as the signature on $m_i$.

If $H_1\text{-coin}_i = 1$, $B$ aborts. This is the point that $B$ uses the forgeability of the adversary $A$ to solve the CDH problem.

Takes $B$’s answers to the verifying equation (2), it is easy to see that the above simulation is perfect. If $B$ does not abort during the simulation, the interaction is indistinguishable with a legal one from the view of $A$.

**Output:** After the adaptive training, $A$ forges a valid signature $\sigma_j = (U_j, V_j)$ on the message $m_j$ and the identifier $ID_n$, while $m_j$ never showed in the Signing Query phase.

If it is not the case that $H_1\text{-coin}_i = 0$, then $B$ returns failure. Since Type-I adversary can select a new secret value $x'$ and compute the corresponding public key $(X'_{ID}, Y'_{ID})$ for the user ID, we can derive its CDH answer $bQ = abP$ from the following equation.

$$V_j = r_jT_j + S_i = r_jk_{j,2}P + x_s(k_{i,1}P + b_l_{i,1}P) = r_jk_{j,2}P + k_{i,1}Y_i + x_l_{i,1}bQ.$$

(4)
It is easy to analyze that \textit{CAS-1} is also unforgeable against Type-II adversary under the same assumption. The proof goes similar to Theorem 1 and hence, it is omitted. So the theorem follows. □

3.1 \textit{CAS-2} Scheme

Now we present the description of the \textit{CAS-2} scheme. Compare with \textit{CAS-1}, \textit{CAS-2} minimizes the storage but sacrifices the communication. We note that the \textit{CAS-2} scheme can be looked as a variant of Gentry and Ramzan’s identity-based aggregate signature scheme in CL-PKC [8].

**Setup**: KGC generates system parameters \texttt{params} and secret value \texttt{master-key} as follows:

1. Randomly selects a generator \(P \in \mathbb{G}_1\);
2. Chooses a random value \(s \in \mathbb{F}_q^*\) as the \texttt{master-key}, and then computes \(Q = sP\);
3. Chooses three cryptographic hash functions \(H_1, H_2, H_3 : \{0, 1\}^* \rightarrow \mathbb{G}_1\);
4. Publishes the system parameters \texttt{params} = \{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H_1, H_2, H_3\}.

**Set Secret Value**: The \(i\)-th user randomly chooses a secret random value \(x_i \in \mathbb{G}_1\) and securely keeps \(x_i\).

**Set Public Key**: The \(i\)-th user computes the user’s public key \(P_i = (X_i, Y_i)\) where \(X_i = x_iP\) and \(Y_i = x_iQ\). Anyone can check if \(P_i\) is valid by the equation (1).

**Partial Private Key Extract**: The \(i\)-th user sends his identifier \(ID_i \in \{0, 1\}^*\) and the public key \(P_i\) to KGC. KGC constructs the partial private key \(D_i = (D_{i,1}, D_{i,2})\) for the \(i\)-th user, where \(D_{i,1} = sH_3(ID_i \| X_i)\) and \(D_{i,2} = sH_4(ID_i \| Y_i)\).

**Set Full Private Key**: After received his partial private key \(D_i\) from the KGC, the \(i\)-th user computes \(S_{i,1} = x_iD_{i,1}\) and \(S_{i,2} = x_iD_{i,2}\), then sets \(S_i = (S_{i,1}, S_{i,2})\) as the full private key.

**Sign**: Given an arbitrary message \(m_i \in \{0, 1\}^*\), the first user randomly chooses \(\alpha \in \mathbb{G}_1\). Each subsequent signer checks that \(\alpha\) was not used before. Alternatively, different signers may arrive at the same \(\alpha\) according to a pre-established negotiation. The \(i\)-th user processes the signing algorithm as follows:

1. Selects \(r_i \in \mathbb{F}_q^*\), then computes \(P_{\alpha} = H_2(\alpha)\);
2. Computes \(c_j = H_5(ID_i \| m_i \| \alpha)\);
3. Computes the signature \((\alpha, U_i, V_i)\), where 
\[ U_i = r_i P \quad \text{and} \quad V_i = r_i P_i + S_{i,1} + c_j S_{i,2}. \]

**Aggregate:** For \(n\) individual signatures given by \(n\) distinct users where \(n = 1, 2, \ldots\), the aggregation goes:

1. Parses \(\sigma_i\) into \(U_i, V_i\);  
2. Computes \(\overline{V} = \sum_{i=1}^{n} V_i\);  
3. Computes \(\overline{U} = \sum_{i=1}^{n} U_i\);  
4. Outputs a compact signature \(\overline{\sigma} = (\overline{U}, \overline{V})\).

**Verify:** Given a signature \(\overline{\sigma}\), one can check its correctness if the following equation holds.

\[
\hat{e}(P, \overline{V}) = \prod_{i=1}^{n} \hat{e}(Y_i, H_1(ID_i || X_i) + c_i H_1(ID_i || Y_i)).
\]  

This ends the description of **CAS-2** scheme. The security analysis is similar to those of **CAS-1** as stated in Section 3.2 and the proof sketch in [8].

**Theorem 2.** If there exists an adversary \(A\) can \((t, \epsilon, n, q_{H_1}, q_{H_2}, q_{H_3}, q_E, q_S)\)-breaks **CAS-2**, then an algorithm \(B\) interacts with \(A\) can be constructed to solve CDH problem in polynomial time bound with a non-negligible probability.

**Proof.** Assume that \(B\) is given an instance \((q, P, aP, bP)\) of the CDH problem, \(B\) will interact with a Type-I adversary \(A\) as follows to compute the CDH answer \(abP\).

**Setup:** \(B\) sets the KGC’s **params** = \(\{\ panties_1, \ panties_2, \hat{e}, P, Q, H_1, H_2, H_3\}\) such that the public key \(Q = aP\). \(H_1, H_2, H_3\) are three random oracles controlled by \(B\).

**Hash Queries:** \(A\) can make hash queries at any time. \(B\) maintains a list to each random oracle.

- For \(H_1\)-query on \((ID, P_i)\) where \(P_i = (X_i, Y_i)\):
  1. If \((ID, P_i)\) is queried before, \(B\) retrieves \((k_{i,1}, l_{i,1})\) from \(H_1\)-list.  
  2. Else \(B\) generates \(k_{i,1} \in \mathbb{Z}_q^*, l_{i,1} = 0\) and \(H_1\)-coin = 0 with the probability \(1 - \frac{1}{q_{H_1}}\); or \(B\) generates \(k_{i,1}, l_{i,1} \in \mathbb{Z}_q^*\) and \(H_1\)-coin = 1 with the probability \(\frac{1}{q_{H_1}}\). \(B\) logs \((ID, P_i, H_1\text{-coin}, k_{i,1}, l_{i,1})\) in the \(H_1\)-list.
3. $B$ responds with $H_1(ID_i \| X_i) = k_{i,1}P$ and $H_1(ID_i \| Y_i) = b_{i,1}P$.

- For $H_2$-query on $\alpha$:

1. If $\alpha$ is queried before, $B$ retrieves $r$ from $H_2$-list.
2. Else $B$ randomly chooses $r \in \mathbb{Z}_q^*$, then logs $(r, \alpha)$ in the $H_2$-list.
3. $B$ responds with $H_2(\alpha) = rP$.

- For $H_3$-query on $(ID_i, m_i, \alpha)$:

1. If $(ID_i, m_i, \alpha)$ is queried before, $B$ retrieves $k_{i,2}$ from $H_3$-list.
2. Else $B$ chooses $k_{i,2} \in \mathbb{Z}_q^*$, then he logs $(ID_i, m_i, \alpha, k_{i,2})$ in the $H_3$-list.
3. $B$ responds with $H_3(ID_i \| m_i \| \alpha) = k_{i,2}P$.

Partial Private Key Extraction: For $A$ asks the partial private key for $(ID_i, P_i)$:

1. If $(ID_i, P_i)$ is queried previously, $B$ retrieves $(H_1\text{-}\operatorname{coin}_i, k_{i,1}, l_{i,1})$ from $H_1$-list.
2. Else $B$ makes $H_1$-query on $(ID_i, P_i)$.
3. If $H_1\text{-}\operatorname{coin}_i = 0$, $B$ responds $D_i = (D_{i,1}, D_{i,2})$ for the $i$-th user where $D_{i,1} = k_{i,1}Q$ and $D_{i,2} = b_{i,1}Y_i$. If $H_1\text{-}\operatorname{coin}_i = 1$, $B$ aborts.

Signing Query: While $A$ requests a signature on $(ID_i, P_i, m_i)$, $B$ retrieves $H_1\text{-}\operatorname{coin}_i$ from $H_1$-list. If $H_1\text{-}\operatorname{coin}_i = 0$, $B$ processes as follows:

1. Selects $r_i \in \mathbb{Z}_q^*$, then computes $U_i = r_iP$;
2. Computes $P_{i,a} = H_2(\alpha)$;
3. Computes $c_i = H_3(ID_i \| m_i \| \alpha)$;
4. Computes $V_i = r_iP_a + S'_{i,1} + S'_{i,2}$, where $S'_{i,1} = xk_{i,1}Q$ and $S'_{i,2} = b_{i,1}Y_i$;
5. Outputs $\sigma_i = (U_i, V_i)$ as the signature on $m_i$.

If $H_1\text{-}\operatorname{coin}_i = 1$, $B$ aborts. This is the point that $B$ uses the forgeability of the adversary $A$ to solve the CDH problem.

Takes $B$’s answers to the verifying equation (5), it is easy to see that the above simulation is perfect. If $B$ does not abort during the simulation, the interaction is indistinguishable with a legal one from the view of $A$.

Output: After the adaptive training, $A$ forges a valid signature $\sigma_j = (U_j, V_j)$ on the message $m_j$ and the identifier $ID_j$, while $m_j$ never showed in the Signing Query phase.
If it is not the case that $H_1(\text{coin}) = 0$, then $B$ returns failure. Since Type-I adversary can select a new secret value $x'$ and compute the corresponding public key $(X', Y')$ for the user ID, we can derive its CDH answer $bQ = abP$ from the following equation.

\[ V_j = r_j T_j + S_i \]

\[ = r_j k_{ij}P + x_s(k_{ij}P + b_{ij}P) \]

\[ = r_j k_{ij}P + k_{ij} Y_i + x_{ij} bQ. \quad (6) \]

It is easy to analyze that \textit{CAS-2} is also unforgeable against Type-II adversary under the same assumption. The proof goes similar to Theorem 2 and hence, it is omitted. So the theorem follows.

\[ \square \]

4. PEFORMANCE COMPARISON

Here we give a performance comparison amongst some related schemes and ours. In order to show the trade-offs for the certificateless and the trust level 3, we choose a PKI-based scheme [4] and an ID-based scheme [8]. Moreover, we also select a general certificateless signature scheme [1] to show the advantages of aggregation. $T_p$ denotes time for one pairing operation in the elliptic curve groups. $T_e$ denotes time for one exponential operation. $n$ is the number of the individual signatures. Let $\ell$ be the length of a group element.

<table>
<thead>
<tr>
<th>Table 2. The comparison of the aggregate signature schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Costs</td>
</tr>
<tr>
<td>Aggregate Length</td>
</tr>
<tr>
<td>Certificate</td>
</tr>
<tr>
<td>Trust Level</td>
</tr>
</tbody>
</table>

From Table 2, we can understand that both \textit{CAS-1} and \textit{CAS-2} pay more computation costs for simultaneously realizing certificateless and the trust level 3. The trade-off is reasonable since an authority in a lower trust level is unacceptable in some applications, e.g., military and government networks. Because \textit{CAS-1} relies less computation in signing process, so it is feasible for the environments where signer sides are limited computational ability. Differently, \textit{CAS-2} is better for the limited storage applications since the length of its aggregate signature is a const value, no matter how many users and messages are involved.
5. CONCLUSION

In this paper, two practical certificateless aggregate signature schemes are proposed. One can adaptively choose either of the proposed CAS schemes by the consideration of which advantage is more important in practice. Both of the CAS schemes are proven secure in the random oracle model (ROM) by assuming the intractability of the computational Diffie-Hellman (CDH) problem over the groups with bilinear maps, without using the forking lemma technique. An interesting open problem is to design such an aggregate signature scheme based on the CL-PKC that neither the storage nor the computation costs is linearly increased with the number of signing messages or involving users in the scheme.

REFERENCES