An aggregate signature scheme is a digital signature scheme that allows aggregation of $n$ distinct signatures by $n$ distinct users on $n$ distinct messages. In this paper, we present an aggregate signcryption scheme (ASC) that is useful for reducing the size of certification chains (by aggregating all signatures in the chain) and for reducing message size in secure routing protocols. The new ASC scheme combines identity-based encryption and the aggregation of signatures in a practical way that can simultaneously satisfy the security requirements for confidentiality and authentication. We formally prove the security of the new scheme in a random oracle model with respect to security properties IND-CCA2, AUTH-CMA2, and EUF-CMA.

Keywords: Identity-based cryptography, signcryption, aggregate signature, bilinear pairing.

I. Introduction

An aggregate signature scheme is a digital signature scheme, the concept of which was first proposed by Boneh and others [1]. Aggregate signature allows aggregation of different signatures by $n$ different users ID on different messages $m_i$. The primary objective of an aggregate signature scheme is to achieve both computation and communication efficiency. In aggregate signature, multiple signatures from various users are combined into a single compact signature. Aggregation can be used to reduce the certificate chains in public key infrastructure (PKI) settings. The aggregate signature has many real world applications ranging from traffic control to documents signed by directors of a company for official purpose.

In certain scenarios, one may need to hide the sending message so that only the receiver will be able to get back the message. In such cases, signcryption comes into the picture. Consider the scenario of an online opinion poll. The verifier has to ensure that all the concerned persons have polled their votes in an efficient way, but one may want his opinion to be secret. Only the verifier will be able to decrypt the messages and get the opinions. Consider another scenario where the directors of a company have to vote on some controversial issue. Each of them wants their vote to be hidden from others since it may disrupt the friendly atmosphere prevailing in the company. In both of these cases, aggregate signcryption can be used to increase efficiency, provide secrecy, and decrease the communication overhead. Aggregate signcryption also has applications in military communication.

The concept of public key signcryption was proposed by Zheng [2]. The idea of this kind of cryptographic primitive is to perform encryption and signature in a single logical step to obtain confidentiality, integrity, authentication, and non-repudiation more efficiently than the sign-then-encrypt approach. Identity-based cryptosystems (IBCs) were first introduced by Shamir in 1984 [3]. IBCs eliminate trust problems...
II. Preliminaries

1. Computation Assumptions

There are some computation assumptions about preliminaries related to an ASC, such as bilinear pairing, the bilinear Diffie-Hellman (BDH) problem, the decisional bilinear Diffie-Hellman (DBDH) problem, and the discrete logarithm (DL) problem. A bilinear pairing is a map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ with the bilinearity, non-degeneracy, and computability properties, where $G_1$ is an additive cyclic group and $G_2$ is a multiplicative cyclic group of the same order $q$. Bilinearity means that given elements $P, Q, Q \in G_1$, then $\hat{e}(P + Q, R) = \hat{e}(P, R) \hat{e}(Q, R)$ and $\hat{e}(P, Q + R) = \hat{e}(P, Q) \hat{e}(P, R)$. In particular, for $a, b \in \mathbb{Z}_q^*$, $\hat{e}(P^a, Q^b) = \hat{e}(P, Q)^{ab} = \hat{e}(P^*, Q^*) = \hat{e}(P, Q^*b)$. Non-degeneracy means that there exist $P, Q \in G_1$, such that $\hat{e}(P, Q) \neq 1_{G_2}$, where $1_{G_2}$ is the identity element of $G_2$. Computability means that there exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in G_1$. The above properties can be derived from Weil or Tate pairing on an elliptic curve over a finite field [15].

For any probabilistic polynomial time algorithm $\mathcal{A}$, the BDH problem in $G_1$ is to compute $\hat{e}(g, g)^{abc}$, and the advantage in solving the BDH problem is defined as $Adv_{\mathcal{A}}^{BDH} = Pr[\mathcal{A}(g, g^a, g^b) = \hat{e}(P, P)^{ab}] - Pr[\mathcal{A}(g, g^a, g^b) = 1] - Pr[\mathcal{A}(g, g^a, g^b) = 1]$. The DL problem is to find $x$ and the advantage in solving the DL problem is defined as $Adv_{\mathcal{A}}^{DL} = Pr[\mathcal{A}(g, h) = x]$ when $g$ and $h$ are given.

2. Framework of ASC

An ASC consists of the following probabilistic polynomial time algorithms.

Setup($k$). Given the security parameter of the system $k$, the private key generator (PKG) generates the set of public parameters $\pi$ and the master secret key $s$ of the system.

Key Extract($ID_a$). Given an identity $ID_a$, the PKG, using the set of public parameters $\pi$ and the master secret key $s$, computes the corresponding private key $<s_i, d_i>$, which is transmitted to ID to be used in ASC in a secure way, and the public key $<X_i, q_i>$.

Signcrypt($m_i, X_i, d_i$, ID$_a$, ID$_b$). Let $\mathcal{M}$ be the message space, $\mathcal{W}$ - the signcrypted message space, and $\mathcal{R}$ - the space of senders. We will identify any member $X \in \mathcal{R}$ by its identity ID$_s$.

For any $m_i \in \mathcal{M}$, $i (1 \leq i \leq n) (n \in \mathbb{Z}^*)$ is an arbitrary fixed integer, the algorithm Signcrypt($m_i, X_i, d_i$, ID$_a$, ID$_b$) is defined as follows:

The sender ID$_a$ having a private key $<s_i, d_i>$ runs this algorithm to generate a signcryption on message $m_i$ that will be aggregated and send it to a receiver with identity ID$_b$. The output is a ciphertext $c_i \in \mathcal{W}$.

Aggregate($\sigma_i$, ID$_b|_1,...,n$). Let $\mathcal{W}$ be the aggregate signcrypted message space. Given a set of $n$ signcryptions $\{c_i\}_{i=1,...,n}$ and the corresponding identity ID$_b$ this algorithm outputs the final
aggregate signcryption $\sigma_{agg}$ in $\mathcal{V}^*$. 

Unsigncrypt $(\sigma_{agg}, s_h, d_h)$. For any $\sigma_{agg}$ in $\mathcal{V}^*$ and a receiver with identity IDB, with the private key $<s_h, d_h>$, the algorithm Unsigncrypt $(\sigma_{agg}, s_h, d_h)$ is defined as follows:

The receiver IDB receives $\sigma_{agg}$ and runs Unsigncrypt $(\sigma_{agg}, s_h, d_h)$. If $\sigma_{agg}$ is a valid aggregate signcryption from $\{ID\}_{i=1,...,n}$ then the output is a plaintext $\{m_i\}_{i=1,...,n}$ $(m_i \in \mathcal{M})$; otherwise, the output is "Invalid."

3. Formal Security Model for Aggregate Signcryption

Three security properties that are desired out of any ASC scheme are message confidentiality, ciphertext authentication, and signature non-repudiation.

Definition 1. An ASC is said to be semantically secure against indistinguishability under adaptive chosen ciphertext attack (IND-ASC-CCA2) if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

Start. The simulator $\mathcal{C}$ runs Setup($\pi$) and sends the set of public parameters $\pi$ to the adversary $\mathcal{A}$.

Phase 1. The adversary $\mathcal{A}$ makes a polynomially bounded number of queries to the simulator $\mathcal{C}$.

--Keygen queries. The adversary $\mathcal{A}$ produces an identity ID and obtains the corresponding secret key of ID.

--Signcrypt queries. $\mathcal{A}$ produces a message $m_i \in \mathcal{M}$, a signer identity ID, and a target identity IDB. Then, $\mathcal{C}$ returns the signcrypted ciphertext $\sigma = \text{Signcrypt}(m_i, X, d, ID, IDB)$ to $\mathcal{A}$, where the private key $d$ is generated by querying the Keygen oracle.

--Unsigncrypt queries. $\mathcal{A}$ produces a receiver identity IDB $\neq \{ID\}_{i=1,...,n}$ and an aggregate signcryption $\sigma_{agg}$. The simulator $\mathcal{C}$ generates the private key $s_h$ by querying the Keygen oracle. $\mathcal{C}$ returns the result of Unsigncrypt $(\sigma_{agg}, s_h, d_h)$ to $\mathcal{A}$. The result returned is $\perp$ if $\sigma$ is an invalid signcrypted ciphertext from $\{ID\}_{i=1,...,n}$ to $\mathcal{A}$.

Selection. $\mathcal{A}$ produces two messages sets $m_{10}$ and $m_{11}$ with equal length from the message space $\mathcal{M}$, identities $\{ID\}_{i=1,...,n}$ and sends them to $\mathcal{C}$. The adversary $\mathcal{A}$ must not have queried the private key corresponding to IDB $\neq \{ID\}_{i=1,...,n}$ in the first phase.

Challenge. The simulator $\mathcal{C}$ chooses randomly a bit $b^* \leftarrow \{0, 1\}$ and obtains the challenge aggregate signcryption $\sigma_{agg}$ by running $\sigma_{agg}^* = \text{Signcrypt}(m_{b^*}, X, d^*, ID^*, IDB^*)$ and Aggregat ($\sigma_{agg}^*, \{ID\}_{i=1,...,n}$), and returns $\sigma_{agg}$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ is allowed to make polynomially bounded number of new queries as in Phase 1 with the restrictions that it should not have made the Unsigncryption queries for the unsigncryption of $\sigma_{agg}$ or the Keygen queries for the private keys of IDB $\neq *$.

Response. $\mathcal{A}$ outputs a bit $b'$ and wins the game if $b' = b^*$. The advantage of $\mathcal{A}$ is defined as

$$\text{Adv}_{\mathcal{A}}^{\text{IND-ASC-CCA2}} = |2\text{Pr}[b^* = b'] - 1|.$$ 

Definition 2. An ASC is said to be existentially unforgeable under adaptive chosen message outsider attack, or AUTH-ASC-CMA secure, if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

Start. The simulator $\mathcal{C}$ runs Setup($\pi$) and sends the set of public parameters $\pi$ to the adversary $\mathcal{A}$.

Query. The adversary $\mathcal{A}$ makes a polynomially bounded number of queries to the simulator $\mathcal{C}$. The attack may be conducted adaptively, and allows the same queries as in the IND-ASC-CCA2 game, namely, Keygen queries, Signcrypt queries, and Unsigncrypt queries.

Forgery. $\mathcal{A}$ produces a new aggregate signcryption $\sigma_{agg}$ sent from a set $\{ID\}_{i=1,...,n}$ of $n$ users on messages $\{m_i\}_{i=1,...,n}$ to a final receiver IDB $\neq \{ID\}_{i=1,...,n}$, where the private keys of the users in $\{ID\}_{i=1,...,n}$ was not queried in query phase and $\sigma_i$ is not the output of a previous query to the Signcrypt queries.

Outcome. The adversary $\mathcal{A}$ wins the game if $\perp$ is not returned by Unsigncrypt $(\sigma_{agg}, s_h, d_h)$.

Definition 3. An ASC is said to be existentially signature unforgeable against chosen message insider attack, or EUF-ASC-CMA secure, if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

Start. The simulator $\mathcal{C}$ runs Setup($\pi$) and sends the set of public parameters $\pi$ to the adversary $\mathcal{A}$.

Query. The adversary $\mathcal{A}$ makes a polynomially bounded number of queries to the simulator $\mathcal{C}$. The attack may be conducted adaptively, and allows the same queries as in the IND-ASC-CCA2 game, namely, Keygen queries, Signcrypt queries, and Unsigncrypt queries.

Forgery. The adversary $\mathcal{A}$ returns a cipher text $\sigma$.

Outcome. The adversary $\mathcal{A}$ wins the game if: Under the private key of IDB, the ciphertext $\sigma_i$ is decrypted as a signed message $(ID_h, \hat{m}_i, \hat{V}_i)$ that satisfies ID$\neq$IDh, ID$\in\{ID\}_{i=1,...,n}$ $\perp$ is not returned by Unsigncrypt $(\sigma_{agg}, s_h, d_h)$ provided that: (1) the private key of the user ID $\in \{ID\}_{i=1,...,n}$ was not queried in query phase; (2) $\sigma_i$ is not the output of a previous query to the Signcrypt queries that involved $m_i$, IDh, and recipient IDh', and resulted in a ciphertext $\sigma_i'$ whose decryption under the private key of IDh' is the claimed forgery $(ID_h, \hat{m}_i, \hat{V}_i')$. 

III. ASC

We propose an ASC scheme in this section. We follow the
framework of ASC that we presented in section II.2.

**Setup**($k$). Given the security parameter of the system $k$, the PKG chooses two groups, $G_1$ and $G_2$, of the same prime order $q$, the generator $g$ of $G_1$, and a bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$. The PKG then chooses four cryptographic hash functions $H_0: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $H_2: G_1 \times \{0, 1\} \rightarrow \mathbb{Z}_q^*$, $H_3: \{0, 1\}^* \times G_1 \rightarrow \mathbb{Z}_q^*$, $H_4: \{0, 1\}^* \times G_1 \times G_2 \rightarrow \mathbb{Z}_q^*$, and $\mathcal{M}=\{0, 1\}^*$, where $le\mathbb{Z}_q^*$ is arbitrarily fixed. The PKG chooses its secret key $s\in\mathbb{Z}_q^*$. The set of system public parameters is $\pi = \{q, G_1, G_2, \hat{e}, g, g^*, H_0, H_1, H_2, H_3\}$.

**Keygen**(ID). For an identity $ID \in \{0, 1\}^*$, the algorithm does the following:

1. Choose a random $r_i \in \mathbb{Z}_q^*$ and computes $X_i = g^{r_i}$, $q_i = H_2(ID, X_i)$;
2. Computes $s_i = X_i^{s_i} \mod q_i$;
3. The PKG sends the corresponding private key $<s_i, d_i>$, which is transmitted in a secure way, and the public key $<X_i, q_i>$ to the user $ID_i$.

**Signcryption**($m_i, X_i$, $d_i$, $ID_i, ID_B$). To send a message $m_i$ to the receiver ID$_B$ with the public key $<X_i, q_i>$, the sender ID$_i$ with the private key $<s_i, d_i>$ and the public key $<X_i, q_i>$ does the following:

1. Choose a random $r \in \mathbb{Z}_q^*$ and computes $W_i = g^{r}$, $w_i = \hat{e}(W_i, X_B)^r$;
2. Let $h_{i1} = H_1(w_i); h_{i2} = H_2(m_i, ID_i, X_i, w_i, ID_B, X_B)$;
3. Computation $h_{i3} = H_3(m_i, ID_i, X_i, h_{i2}, w_i, ID_B, X_B)$;
4. Computes $V_i = (r, h_{i2} + h_{i3}) \mod q_i$;
5. Sets $c_i = (m_i, V_i, h_{i2}); Z_i = g^{V_i}$;
6. Outputs $\sigma_i = <c_i, W_i, Z_i, X_i>$ as the signcryption of ID$_i$ on message $m_i$.

**Aggregate**($\sigma_{agg}, ID_{agg} , \ldots , n$). On input, a set of $n$ signcryption $\sigma_i = <c_i, W_i, Z_i, X_i>, i=1$ to $n$, and the corresponding identity $ID_i$ (such that $\forall i=1$ to $n$, $\sigma_i$ is the signcryption on message $m_i$ by ID$_i$):

1. Let $Z_{agg} = \prod_{i=1}^{n} Z_{i}$;
2. Output the final aggregate signcryption $\sigma_{agg} = <\{c_i, W_i, X_i, ID_{agg} \}_{i=1 \ldots , n}, Z_{agg}>$.

The signcryption can be done by any of the senders or by any third party.

**Unsigncryption**($\sigma_{agg}, s_{agg}, d_{agg}$). On input, the aggregate signcryption $\sigma_{agg} = <\{c_i, W_i, X_i, ID_i \}_{i=1 \ldots , n}, Z_{agg}>, the receiver with identity ID$_{agg}$, the public key $<X_{agg}, q_{agg}>$, and the private key $<s_{agg}, d_{agg}>$, do the following:

1. Compute $w_i = \hat{e}(W_i, s_{agg})$, recover $m_i || V_i = c_i \oplus H_1(w_i)$;
2. For $i=1$ to $n$, compute $h_{i2} = H_2(m_i, ID_i, X_i, w_i, ID_B, X_B)$,$h_{i3} = H_3(m_i, ID_i, X_i, h_{i2}, w_i, ID_B, X_B)$;
3. Check if $Z_{agg} = g^{\sum_{i=1}^{n} V_i}$, and $Z_{agg} = \prod_{i=1}^{n} (W_i)^{h_{i2}} g^{h_{i3}} (g^r)^{\sum_{i=1}^{n} h_{i3}}$.

If the check succeeds, the output is $m_i$, which has been decrypted; otherwise, the output reads “Invalid.”

**IV. Analysis of Proposed Method**

1. Correctness

Correctness of the unsigncryption algorithm:

\[ w_i = \hat{e}(W_i, s_{agg}) = \hat{e}(g^r, X_B^r) = \hat{e}(g^r, X_B)^r, \]
\[ Z_i = g^{h_{i2}} = g^{h_{i2} + h_{i3}} = (g^r)^{h_{i2}} g^{h_{i3}}, \]
\[ Z_{agg} = \prod_{i=1}^{n} Z_{i} = \prod_{i=1}^{n} (W_i)^{h_{i2}} (X_i)^{h_{i3}} (g^r)^{h_{i3}} \]
\[ = \prod_{i=1}^{n} (W_i)^{h_{i2}} \prod_{i=1}^{n} (X_i)^{h_{i3}} (g^r)^{\sum_{i=1}^{n} h_{i3}}. \]

2. Proof of Confidentiality

**Theorem 1.** Assume there is an IND-ASC-CA2 adversary $\mathcal{A}$ that is able to distinguish two valid ciphertexts during the game defined in Definition 1 with a non-negligible advantage and asks Keygen queries, Signcryption queries, and Unsigncryption queries; then, there exists a simulator $\mathcal{C}$ that can solve an instance of the DBDH problem with a non-negligible advantage.

*Proof.* The proof proceeds in the IND-ASC-CA2 game. Assume that the simulator $\mathcal{C}$ receives a random DBDH problem instance $(g, g^r, g^s, g^w)$, then, the goal is to decide whether $w = \hat{e}(g, g^r)^{s}\neq w$ or $c$. $\mathcal{C}$ will run the adversary $\mathcal{A}$ as a subroutine and act as the adversary’s challenger in the IND-ASC-CA2 game.

Start. The simulator $\mathcal{C}$ sets the master public key $g^r = g^w$ and gives the system public parameters to $\mathcal{A}$.

Phase 1. The simulator $\mathcal{C}$ will set the random oracles of $H_0$, $H_1$, $H_2$, $H_3$, Keygen, Signcryption, and Unsigncryption. To maintain consistency, the simulator $\mathcal{C}$ will maintain four lists: $H_0$-List, $H_1$-List, $H_2$-List, and $H_3$-List, which will be detailed later. $\mathcal{C}$ will also simulate all oracles required during the game and control the $H_0$ random oracle. The adversary $\mathcal{A}$ outputs and threatens to attack the identity ID$_B$.

$H_0$ Oracle. When the $H_0$ oracle is queried with $ID = \{0, 1\}^*$, $C$ does the following: checks the $H_0$-List $<ID_B, q_B, x_B, x_i \in \mathbb{Z}_q^*$,
if IDi = IDb, selects a new random \(\lambda_i \in \mathbb{Z}_q^*\); sets \(X^i = g^{\lambda_i}\); adds the tuple \(<X_i, q_i, *\rangle\) to the H0-List; and returns \(q_i\). Otherwise, \(C\) selects a new random \(\lambda_i \in \mathbb{Z}_q^*, x_i \in \mathbb{Z}_q\); sets \(X^i = g^{x_i}\); \(q_i = \lambda_i\); adds the tuple \(<X_i, q_i, x_i\rangle\) to the H0-List, and returns \(q_i\).

\(H_0.\) When the \(H_0\) oracle is queried with an input \(w_i\), \(C\) checks the H0-List. If there exists a tuple \(<w_i, h_{0i}\rangle\) in H0-List, \(C\) returns \(h_{0i}\). Otherwise, \(C\) selects a new random \(h_{0i} \in \mathbb{Z}_q^*\), adds the tuple \(<w_i, h_{0i}\rangle\) to the H0-List, and returns \(h_{0i}\).

\(H_2.\) When the \(H_2\) oracle is queried with an input \((m_i, ID_i, X_i, w_i, ID_b, X_b)\), \(C\) checks the H2-List. If there exists a tuple \(<m_i, ID_i, X_i, h_{2i}, w_i, ID_b, X_b, h_{2i}\rangle\) in the H2-List, \(C\) returns \(h_{2i}\). Otherwise, \(C\) chooses a new random \(h_{2i} \in \mathbb{Z}_q^*\), adds the tuple \(<m_i, ID_i, X_i, h_{2i}, w_i, ID_b, X_b, h_{2i}\rangle\) to the H2-List, and returns \(h_{2i}\).

\(Keygen.\) When \(A\) makes a Keygen query with ID, as the input, \(C\) checks the H0-List to verify whether or not there is an entry for ID. If the H0-List does not contain an entry for ID, return \(\perp\). Otherwise, \(C\) selects a new random \(\lambda_i \in \mathbb{Z}_q^*\), sets \(X^i = g^{\lambda_i}\), and \(q_i = \lambda_i\); adds the tuple \(<X_i, q_i, \lambda_i, \sigma_i, 0\rangle\) to the H0-List, and returns \(\sigma_i\).

\(Signcrypt.\) When \(A\) makes a Signcrypt query with ID, as the input, \(C\) checks the H0-List to verify whether or not there is an entry for ID. If the H0-List does not contain an entry for ID, return \(\perp\). Otherwise, \(C\) executes \(Signcrypt(m_i, X_i, d_i, ID_i, ID_b, \sigma_i, 0)\) as usual and returns what the Signcrypt algorithm returns.

\(Unsigncrypt.\) When \(A\) makes an Unsigncrypt query with \(\sigma_{agg} = \langle c_i, W_i, X_i, ID_i \rangle_{i=1,\ldots,n} \in Z_{agg}\rangle\) and the receiver with identity IDb, \(A\) first verifies whether or not there are entries for ID, \((ID \neq ID_b)\) and IDb in H0-List and there is an entry of the form \(<X_i, q_i, \lambda_i, \sigma_i\rangle\). If at least one of these conditions is not satisfied, return \(\perp\). Otherwise, \(C\) executes \(Unsigncrypt(\sigma_{agg}, s_b, d_b)\) in the normal way and returns what the Unsigncrypt algorithm returns.

\(Challenge.\) After sufficient training, \(A\) submits two equal length messages \(m_0\) and \(m_1\). \(C\) randomly chooses a bit \(b' \leftarrow\{0, 1\}\) and obtains the challenge signcrypted ciphertext by running \(Signcrypt(m_{b'}, X_{b'}, d_{b'}, ID_{b'}, ID_b)\) and \(Aggregate(\{\sigma_i, ID_i\}_{i=1,\ldots,n})\), then returns \(\sigma_{agg}\) to \(A\).

\(Phase 2.\) This phase is similar to Phase 1. However, in Phase 2, \(A\) cannot ask for Unsigncrypt on the challenge aggregate signcrypted \(\sigma_{agg} = \langle c_i, W_i, X_i, ID_i \rangle_{i=1,\ldots,n} \in Z_{agg}\rangle\) or the Keygen queries for the secret keys of IDb.

\(Output.\) After \(A\) has made a sufficient number of queries, \(A\) returns its guess: a bit \(b\). If \(b = b'\), then \(C\) outputs 1 as the answer to the DBDH problem. Otherwise, it outputs 0. Since the adversary is denied access to the Unsigncrypt oracle with the challenge signcryption, for \(A\) to find that \(\sigma_i\) is not a valid ciphertext, \(A\) should have queried the \(H_1\) Oracle with \(w_i = \hat{e}(W_i, s_b)\). Here, \(s_b\) is the private key of the receiver, and it is \((X_b)^s = \hat{e}(g^s, g^{ab}) = \hat{e}(g, g)^{ab}\). Also, \(C\) has set \(W_i = g^s\). We have \(w_i = \hat{e}(W_i, s_b) = \hat{e}(g^s, g^{ab}) = \hat{e}(g, g)^{ab}\). \(\square\)

3. Proof of Authentication

\(Theorem 2.\) The ASC proposed is secure against any probabilistic polynomial-time AUTH-ASC-CMA2 adversary \(A\) under the random oracle model if the DL problem is hard in \(G_1\).

\(Proof.\) On getting a DL problem instance \((g, W_r = g^{r'})\) and \((g, g^{d'})\) as a challenge in the AUTH-ASC-CMA2 game defined in Definition 2, the simulator \(C\) uses \(A\) to solve the DL problem. The goal of \(C\) is to determine \(r\) and \(d\). The simulator \(C\) gives the system public parameters to \(A\). \(A\) knows \(g^i\) from computing \(W_f(g^{i'})^y = g^{d'} = g^s \cdot W_r = W_r(g^{r'})\). The proof of Theorem 2 is similar to the proof of Theorem 1, with some changes given in the following random oracles.

\(H_0.\) When the \(H_0\) oracle is queried with an input \(ID_i \in \{0, 1\}^*, C\) checks the H0-List; if the tuple \(<ID_i, X_i, q_i, \lambda_i, \sigma_i, 0\rangle\) exists, \(C\) returns \(\sigma_i\). Otherwise, \(C\) selects a new random \(\lambda_i \in \mathbb{Z}_q^*, x_i \in \mathbb{Z}_q\); sets \(X_i = g^{x_i}\), \(q_i = \lambda_i\); adds the tuple \(<ID_i, X_i, q_i, \lambda_i, \sigma_i, 0\rangle\) to the H0-List, and returns \(\sigma_i\).

\(Keygen.\) When \(A\) makes a Keygen query with ID, as the input, \(C\) checks the H0-List to verify whether or not there is an entry for ID. If the H0-List does not contain an entry for ID, return \(\perp\). Otherwise, \(C\) executes \(Keygen(m_i, X_i, d_i, ID_i, ID_b, \sigma_i, 0)\) as usual and returns what the Keygen algorithm returns.

\(Unsigncrypt.\) When \(A\) makes an Unsigncrypt query with \(\sigma_{agg} = \langle c_i, W_i, X_i, ID_i \rangle_{i=1,\ldots,n} \in Z_{agg}\rangle\) and the receiver with identity IDb, \(A\) first verifies whether or not there are entries for ID, \((ID \neq ID_b)\) and IDb in H0-List and there is an entry of the form \(<X_i, q_i, \lambda_i, \sigma_i\rangle\). If at least one of these conditions is not satisfied, return \(\perp\). Otherwise, \(C\) executes \(Unsigncrypt(\sigma_{agg}, s_b, d_b)\) in the normal way and returns what the Unsigncrypt algorithm returns.
The ID oracle. We only provide the changes.

A machine again with the same random tape but with a different hash value. C obtains the ‘signatures’ \( V_r = r \cdot h_{\gamma_2} + h_{\gamma_d} d \) and \( V_r' = r \cdot h_{\gamma_2}' + h_{\gamma_d}' d \), with \( h_{\gamma_2} \neq h_{\gamma_2}' \) and \( h_{\gamma_d} \neq h_{\gamma_d}' \). C successfully computes \( r \) and \( d \). Indeed, from \( V_r = r \cdot h_{\gamma_2} + h_{\gamma_d} d \) and \( V_r' = r \cdot h_{\gamma_2}' + h_{\gamma_d}' d \), we have

\[
\begin{align*}
    r &= \frac{V_r - V_r'}{h_{\gamma_2} - h_{\gamma_2}'} - \frac{h_{\gamma_d} \cdot h_{\gamma_2} - h_{\gamma_d}^* \cdot h_{\gamma_2}'}{h_{\gamma_2} - h_{\gamma_2}'} \neq 0, \\
    d &= \frac{V_r - V_r'}{h_{\gamma_2} - h_{\gamma_2}'} - \frac{h_{\gamma_d} \cdot h_{\gamma_2} - h_{\gamma_d}^* \cdot h_{\gamma_2}'}{h_{\gamma_2} - h_{\gamma_2}'} \neq 0.
\end{align*}
\]

4. Proof of Non-repudiation

**Theorem 3.** The ASC proposed is secure against any probabilistic polynomial-time EUF-ASC-CMA adversary \( A \) under the random oracle model if the DBDH problem is hard in \( G_1 \).

**Proof.** The proof of Theorem 3 is similar to the proof of Theorem 2, with some changes in the following random oracles. We only provide the changes.

**Keygen Oracle.** If the \( H_0 \)-List does not contain an entry for \( ID_h \), \( C \) returns \( \bot \). Otherwise, if \( ID_h = ID_s \) (this corresponding identity ID is called the “guessed” sender), \( C \) recovers the tuple \( < X_h, q_h, x_i > \) from the \( H_0 \)-List and return \( < X_h, q_h, \ast, \ast > \). If \( ID_h \neq ID_s \), the tuple \( < X_h, q_h, x_i > \) from the \( H_0 \)-List and return \( < X_s, q_i, s_i, d_i > \), where \( s_i = (g^i)^v \), and \( d_i \in \mathbb{Z}_q \) is randomly selected.

Eventually, \( A \) returns a forgery, consisting of a ciphertext \( \sigma \) and a recipient identity \( ID_h \). \( C \) decrypts the ciphertext for \( ID_h \) (by invoking its own decryption oracle), which causes the plaintext forgery (ID, \( m_s, V_s \)) to be revealed. Note that if \( C \) has made the correct guess, that is, \( ID_h = ID_s \), then \( ID_h \neq ID_s \) and the decryption works.

If \( \sigma \) is a valid signature from \( ID_s \) to receiver \( ID_h \), that is, a message \( m_s \) is returned by the Unsigncrypt algorithm, then \( C \) applies the oracle replay technique to produce two valid signed messages (ID, \( m_s, V_s \)) and (ID, \( m_s, V_s' \)) on a message \( m \), from the \( ID \) to receiver \( ID_h \). This is achieved by running the turing machine again with the same random tape but with a different hash value. \( C \) obtains the ‘signatures’ \( V_s = r \cdot h_{\gamma_2} + h_{\gamma_d} d \) and \( V_s' = r \cdot h_{\gamma_2}' + h_{\gamma_d}' d \), with \( h_{\gamma_2} \neq h_{\gamma_2}' \) and \( h_{\gamma_d} \neq h_{\gamma_d}' \).

5. Efficiency

The primary objective of the aggregate signature scheme is to achieve both computation and communication efficiency. Using aggregate signature schemes, signatures from different users on different messages can be aggregated into a single compact signature. We eliminate the interaction among the signers before signcryption generation, which reduces the communication complexity to a large extent. Also, our ASC is more efficient than the sign-then-encrypt approach. We have compared our scheme (ASC) with the sign-then-encrypt scheme (AS and BF) in Table 1, where the aggregate signature scheme (AS) is proposed in [9] and the encryption scheme (BF) is proposed in [15]. In the comparison table, \( |M| \) represents the length of a message. \( |G_i| \) is the number of \( G_i \) elements.

The major parameters involved in our ASC scheme are the computation costs for Signcrypt and Unsigncrypt operations. For computation cost, we mainly consider the number of pairing \( \hat{e} \) computations performed, as they are the costliest operations involved. In our ASC scheme, the sender only performs one pairing \( \hat{e} \) computation, and the receiver only performs \( n \) pairings \( \hat{e} \) computations. In the IBAS scheme of Selvi and others [13], each sender only performs one pairing \( \hat{e} \) computation, but the receiver performs \( 2n+3 \) pairings \( \hat{e} \) computations. Otherwise, the Unsigncrypt algorithm in Selvi and others’ IBAS scheme [13] does not explain how to recover the unknown users’ signatures from the aggregate signature. Thus, our ASC is more efficient than the IBAS scheme of Selvi and others [13].

We conducted five experiments for one pairing \( \hat{e} \) operation using a pairing-based cryptography library [16], running each experiment 1,000 times. The results of the five experiments were 8.258133 s, 8.265714 s, 8.243305 s, 8.394295 s, and 8.384370 s. Therefore, the average time of one pairing \( \hat{e} \) operation is about 0.008309 s.

**V. Conclusion**

We studied an ASC built upon the identity-based aggregate signature scheme proposed by Selvi and others [9]. Our proposed ASC was formally proven to be secure with respect to its IND-CCA2, AUTH-CMA2, and EUF-CMA security properties in a random oracle model. The ASC does not need the interaction among the signers, which is a requirement in existing efficient aggregate signature schemes, and it is efficient in pairing \( \hat{e} \) computations.

### Table 1. Ciphertext size.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Ciphertext size for sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASC</td>
<td>(</td>
</tr>
<tr>
<td>AS and BF</td>
<td>(</td>
</tr>
</tbody>
</table>
Appendix:

In Table A1, we summarize all the notations of our paper.

<table>
<thead>
<tr>
<th>Name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition 1</td>
<td>An ASC is said to be semantically secure against indistinguishability under adaptive chosen ciphertext attack (IND-ASC-CCA2) if no probabilistic polynomial time adversary $A$ has a non-negligible advantage in the following game.</td>
</tr>
<tr>
<td>Definition 2</td>
<td>An ASC is said to be existentially ciphertext unforgeable under adaptive chosen message outsider attack, or AUTH-ASC-CMA2 secure, if no probabilistic polynomial time adversary $A$ has a non-negligible advantage in the following game.</td>
</tr>
<tr>
<td>Definition 3</td>
<td>An ASC is said to be existentially signature unforgeable against chosen message insider attack, or EUF-ASC-CMA secure, if no probabilistic polynomial time adversary $A$ has a non-negligible advantage in the following game.</td>
</tr>
<tr>
<td>Theorem 1</td>
<td>Assume there is an IND-ASC-CCA2 adversary $A$ that is able to distinguish two valid Ciphertexts during the game defined in definition 1 with non-negligible advantage and asking Keygen queries, Signcrypt queries, and Unsigncrypt queries, then there exists a simulator $C$ that can solve an instance of the DBDH problem with non-negligible advantage.</td>
</tr>
<tr>
<td>Theorem 2</td>
<td>The ASC proposed is secure against any probabilistic polynomial-time AUTH-ASC-CMA2 adversary $A$ under the random oracle model if the DL problem is hard in $G_1$.</td>
</tr>
<tr>
<td>Theorem 3</td>
<td>The ASC proposed is secure against any probabilistic polynomial-time EUF-ASC-CMA adversary $A$ under the random oracle model if the DBDH problem is hard in $G_1$.</td>
</tr>
</tbody>
</table>

References


Xun-Yi Ren received his BS in computer application from the University of Science and Technology, Beijing, China, in 1998, and his MS in management science and PhD in information networks, from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004 and 2008, respectively. At present, he is an associate professor in the Department of Information Security, in the College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China. His current research interests include information security, grid & cloud computing, and machine learning.
Zheng-Hua Qi received her BS in computer application from the University of Science and Technology, Beijing, China, in 1998, and her MS in computer science from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004. She is currently a PhD candidate in information security at the same university. Her main research interests include cryptography and information security.

Yang Geng received his PhD in computer science from Université Laval, Québec, Canada, in 1994. From 1995 to 1996, he worked with the Application Research Center, Canada University of Computing Technology, Montreal, Canada, as a postdoctoral researcher. He is currently a professor and doctorate supervisor in the College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China. His research interests include information security, parallel and distributed computing, and wireless sensor networks.