Identity-based Aggregate Signatures with Verifiable Single Ones

Yang ZHANG Jun-Liang CHEN

State Key laboratory of Networking and Switching Technology
Beijing University of Posts & Telecommunications
Beijing 100876, China
YangZhang@bupt.edu.cn

***Abstract*— In an aggregate signature scheme, different signatures from different signers on different messages can be aggregated to reduce the cost of computation and communication. Using an identity-based signature method, any one can verify signatures by the identity of the signer without transmitting certificates. Currently, in most identity-based aggregate signature schemes, aggregate signature verification might require complex pairing operations, or some interactions among the signers might be required. In addition, the individual signatures in those aggregate signatures are often insecure or restricted in special scenarios, which does not satisfy the requirement that an individual signature can be used independently and can also be aggregated on-demand. This paper tries to address this issue by proposing an identity-based aggregate signature scheme in which an individual one can be securely and conveniently used. Our scheme is efficient with constant paring operation, and different signers can concurrently sign different messages. The security of our scheme is proved in the random oracle model.**

***Keywords*—**Digital Signature Scheme; Identity-based Aggregate Signature;Random Oracle Model; Computaional Diffie-Hellman Problem

1. INTRODUCTION

1.1 Motivation

In an environment with constrained communication capability, aggregating different signatures from different signers on different messages is desirable, such as in ad hoc networks. Lots of researchers have been trying to design short and efficient signatures. However, to design a highly secure signature scheme with minimal time and communication complexity is often a challenge. Aggregate signature schemes work with low communication and computation costs. An aggregate signature is to integrate multiple signatures signed by multiple signers into a single signature such that the communication cost is reduced. It is also used to reduce the computation cost by verifying a single aggregated signature instead of multiple signatures. In an identity-based signature scheme, the verifier verifies a signature under the signer's identity and PKG's (Private Key Generator) public key without transmitting certificates. When all signers are clients of the same PKG, the verifier only needs one traditional public key to verify multiple identity-based signatures on multiple documents. Therefore, designing an identity-based aggregate signature scheme is very appealing.

Currently, most identity-based aggregate signature schemes do not achieve constant computation cost during verification, or need some special constraints such as complex interactions, global states, and so on [1]. Gentry and Ramzan in [2] had proposed the most efficient identity based aggregate scheme. But the requirement to agree upon a common random string makes it unsuitable for most real life scenarios. Even if we adopt a system time as the random string to avoid interactions, only one signature can be produced by one signer in each time interval. Otherwise, universal forgery of their signature is possible. In addition, their security model has the requirement not to query signature oracle for the challenge message, because according to an individual signature on a message, one can generate a different valid signature. This should be considered as a weakness in strong unforgeability security model. In many applications, an individual signature should be usable as a traditional one, and can be aggregated on demand.

S. Sharmila Deva Selvi et al. proposed an identity-based aggregate signature scheme without pairings, called IBAS-1 [14]. Their scheme is not a real identity-based scheme because another public key need to be published, although it is called token. In addition, its security is not proved, and the forking lemma [3] should be used in the proof, which does not yield some exact security bound. Therefore, we try to address this issue by proposing an identity-based aggregate signature scheme with security in the strong security model.

1.2 Related work

Since the aggregate signature notion was introduced by Boneh et al. [4], a lot of aggregate signature schemes have been proposed such as [5], [6], [7], [8], [9]. Gentry and Ramzan in [2] has presented the most efficient identity-based aggregate scheme where only three pairing operations are executed during signature verification phase. In their scheme, all the signers have to agree upon a common random string. However, the expensive common random string cannot be reused, otherwise forgery is possible.

Wen et al. in [10] proposed an aggregate signature scheme with constant pairing operation, but there exists a forgeability attack which has been pointed out in [11]. Wang Zhu et al. in [12] also proposed a practical aggregate signature scheme with

constant pairing operation. A valid user of the system will be able to forge a signature on any message if she gets an individual signature on some message by the corresponding user [11].

The aggregate signature schemes proposed by Shi et al. [9] and Xiangguo et al. [6] were efficient respectively in terms of computation complexity. Although the scheme in [9] achieves efficiency in computation, a universal forgery of the signature of any signer is still possible as shown in [11]. In [6], all the signers have to broadcast their own random number used for signing to all the signers. This is not practical in most environments.

Xu et al. in [7] proposed an identity-based aggregate signature scheme, which requires complex pairing operations during signature verification. Javier Herranz et al. gave an identity based signature scheme [8] with partial aggregation. But her scheme produced deterministic signature and used complex pairing operations during verification. Lei Zhang et al. proposed a certificateless aggregate scheme in [13]. Although it is secure, the pairing operation is not constant during signature verification.

S. Sharmila Deva Selvi [14] proposed efficient and provably secure identity-based aggregate signature schemes with partial and full aggregation. In his first scheme, partial aggregate was achieved and no pairing operation was required. However, this scheme was not strict identity-based and did not provide security proof. The security proof could use Forking Lemma [3], which does not yield exact security bound. The full aggregate signature scheme in [14] required complex paring operations during signature verification which linearly grow in the number of signers, and the individual signature was not secure in strong unforgeability security model as in [2].

1.3. Our contribution

The contribution of this paper is two-fold. Our first contribution is that an identity-based aggregate signature scheme is introduced with constant pairing computation, where different signers can concurrently generate signatures without number restriction and individual signatures can be securely used. Our second contribution is that a strong security model is defined to prove the security of our scheme, where a message/signature pair is taken as challenge unlike the one in [2] with only message.

This paper is structured as follows. Section 2 gives a description on preliminaries. In Section 3, the general scheme and security model are described. Section 4 contains our constructions. Section 5 gives the security analysis of our scheme. Section 6 gives the performance analysis. Finally, conclusions are drawn in Section 7.

2. PRELIMINARIES

Suppose there are two groups G_1 and G_2 of the same prime order p and security parameter κ. Assume there is a discrete logarithm problem with hardness in both groups. A cryptographic bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$ should satisfy the following properties [15], [16]:

1) Bilinearity: $\forall a,b \in Z_p^*$, $P,Q \in G_1$, $\hat{e}(aP,bQ) = \hat{e}(P,Q)^{ab}$.

2) No-degeneracy: for any point $P \in G_1$, $\hat{e}(P,P) \neq 1_{G_2}$.

3) Computability: there exists an efficient algorithm to compute $\hat{e}(P,Q)$ for $\forall P,Q \in G_1$.

Definition 1. Computational Diffie-Hellman Problem (CDH). *Let* (G_1,G_2,e) *be as above. Given* $P,aP,bP \in G_1$ *for unknown* $a,b \in Z_p$, *to evaluate* abP.

The CDH problem will be considered to be hard. It means that the succeeding probability of any probabilistic, polynomial-time, 0/1 valued function in solving the CDH problem is negligible. A function $F(y)$ will be said to be negligible when it is not greater than $1/y^l$ for every fixed $l > 0$ and sufficiently large integer y.

3. IDENTITY-BASED AGGREGATE SIGNATURES

We first define the procedures of an Identity-based aggregate signature (IBAS) scheme, and then describe what it means for IBAS schemes to be secure.

3.1 Components of IBAS

An IBAS scheme is composed of six algorithms: (1) an algorithm to build system parameters and a master private key by

the private key generator (PKG), (2) an algorithm to generate key by the PKG, (3) an algorithm to extract key by the PKG for individual users, (4) an algorithm to sign by an individual user, (5) an algorithm to aggregate multiple individual signatures, (6) and an algorithm to verify an identity-based aggregate signature:

- **Setup :** The PKG provides the security parameter κ as the input to this algorithm and generates the system parameters $param$ and the master private key msk. The PKG publishes the $param$ and keeps the msk secret.
- **KeyGen :** A user provides her identity ID_i to the PKG. The PKG runs this algorithm with identity ID_i, $param$ and msk as the input and securely outputs the private key S_i to the user.
- **Signing :** For generating a signature on a message m_i, the user provides her ID_i, her private key S_i, $param$, msk and message m_i as input to this algorithm. This algorithm generates a valid signature sig_i on message m_i by the user.
- **Verification :** This algorithm on input of a signature sig_i on message m_i by the user with her identity ID_i checks whether sig_i is a valid signature on message m_i by ID_i. If true, it outputs "`valid`", else it outputs "`invalid`".
- **Aggregate :** On receiving various signatures sig_i $(i = 1 \, to \, n)$ from different users ID_i $(i = 1 \, to \, n)$, any the third party or one of the signers can run this algorithm and generate the aggregate signature sig.
- **Aggregate Verification :** This algorithm on input of an aggregate signature sig, the list of message and identity pairs (m_i, ID_i) $(i = 1 \, to \, n)$ and the $param$ checks whether sig is a valid aggregate signature. If true, it outputs "`valid`", else output "`invalid`".

3.2 Security Model

An IBAS scheme should be secure against existential forgery under an adaptive-chosen-message and an adaptive-chosen-identity attack. Informally, existential forgery here means that the adversary attempts to forge an identity-based aggregate signature on identities and messages of her choice.

We formalize the identity-based aggregate signature model as below. The adversary's goal is the existential forgery of an aggregate signature. We give the adversary the power to choose the identities on which it wishes to forge a signature, the power to request the identity-based private key on all but one of these identities. The adversary is also given access to a signing oracle on any desired identity.

Definition 2. Unforgeability security notion for IBAS.

An identity-based signature scheme is said to be strongly unforgeable under adaptive chosen-message attacks if no probabilistic polynomial time adversary has a non-negligible advantage in this game:

1) The challenger runs the setup algorithm to generate the system's parameters and master secret key, and then sends them to the adversary A.

2) The adversary A performs a series of queries:

- *Private Key queries: A impersonates a user with her identity ID to query private keys. The challenger computes and returns a private key.*
- *Signature queries: A sends a message m, a time, and an identity ID, then receives a signature on m by ID.*

3) After a polynomial number of queries, A produces an aggregate signature sig on messages $(m_1, \cdots, m_n)$ under $(ID_1, \cdots, ID_n)$. The private key for one ID_i was not queried by A. The message signature pair (m_i, sig_i) for one ID_i was not returned by the signature oracle during stage 2 either.

The adversary A wins the game if the signature verification algorithm outputs 1 when it is run on the tuple

$(sig,(ID_1,\cdots,ID_n),(m_1,\cdots,m_n))$. *The adversary's advantage is defined to be its probability of producing a forgery taken over the coin-flippings of the challenger and* A.

4. CONSTRUCTIONS

In this section, we describe an identity-based aggregate signature scheme with verifiable single ones. This scheme consists of six algorithms where the current time period ϕ takes *time* as a time slot. For example, if $time = 2010-09-12:18:51$, then $\phi = 2010-09-12:17$ under the rule that $(X:00-00:30, X:00+00:30]$ is mapped to X. In our scheme, no strict time synchronization is needed and each signer can use her local system time during signing messages. If the map rule is defined, no interaction is required.

Definition 3 (IBAS). Identity-based aggregate signature Π_{AS}. Π_{AS} *is made up of six algorithms as follows:*

Setup(1^{κ})

The PKG setups $G_1, G_2, \hat{e}$ *and* $P \in G_1$.

It then picks cryptographic hash functions $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to G_1$, *and* $H_3 : \{0,1\}^* \to Z_p^*$;

chooses a master-key $s \in_R Z_p$; *and computes* $P_{pub} = sP$.

Its secret is s *and public parameters are* $param = (G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3)$.

KeyGen$(s, param, ID_i)$

The PKG issues a private key to every user according to her identity ID_i. *It computes as follows:*

$Q_i = H_1(ID_i)$;

$S_i = sQ_i$.

The private key is S_i.

Signing $(m_i, S_i, param)$

The user ID_i *computes as follows:*

gets the current time period ϕ;

computes $T_i = \alpha_i P$ *where* $\alpha_i \in_R Z_p^*$;

computes $Q = H_2(\phi)$;

computes $c_i = H_3(m_i, Q, Q_i, T_i)$;

computes $\sigma_i = c_i \alpha_i Q + S_i$;

returns $sig_i = (\phi, \sigma_i, T_i)$.

Verificati on (m_i, sig_i, ID_i)

Any user verifies the signature sig_i *as follows:*

parses $sig_i = (\phi, \sigma_i, T_i)$;

computes $Q_i = H_1(ID_i)$;

computes $Q = H_2(\phi)$;

computes $c_i = H_3(m_i, Q, Q_i, T_i)$;

checks $\hat{e}(\sigma_i, P) \stackrel{?}{=} \hat{e}(c_i T_i, Q)\hat{e}(Q_i, P_{pub})$;

if true, returns "valid", else return "invalid".

Aggregate $(sig_1, sig_2, \cdots)$

For the aggregation subset of signers, we assign to each signer an index i, *ranging from 1 to* n. *Anyone can*

aggregate a collection of individual signatures $sig_i = (\phi, \sigma_i, T_i)$, $i = 1$ *to* n *. It computes*

$\sigma = \sum_{i=1}^{n} \sigma_i$.

It returns $sig = (\phi, \sigma, T_1, T_2, \cdots, T_n)$.

Aggregate Verificati on$(sig, (m_1, ID_1), \cdots, (m_1, ID_n))$

Any user verifies the signature sig *as follows:*

parses $sig = (\phi, \sigma, T_1, T_2, \cdots, T_n)$ *;*

computes $Q = H_2(\phi)$ *;*

computes

for $i = 1, \cdots, n$

$Q_i = H_1(ID_i)$,

$c_i = H_3(m_i, Q, Q_i, T_i)$,

checks $\hat{e}(\sigma, P) \stackrel{?}{=} \hat{e}(\sum_{i=1}^{n} c_i T_i, Q) \hat{e}(\sum_{i=1}^{n} Q_i, P_{pub})$ *;*

if true, returns "valid", else return "invalid".

In our scheme, an individual signature produced by one signer can be used independently, and can be aggregated on demand. In one time interval, one signer can sign many messages and many different signatures can be generated for one message.

Assume the elements in the individual signature $sig_i = (\phi, \sigma_i, T_i)$ are the same as in *Signing* . The correctness of Π_{AS} is illustrated as follows:

$$Q = H_2(\phi)$$

$$\begin{aligned} \hat{e}(\sigma_i, P) &= \hat{e}(c_i \alpha_i Q + S_i, P) \\ &= \hat{e}(c_i \alpha_i P, Q) \hat{e}(sQ_i, P) \\ &= \hat{e}(c_i T_i, Q) \hat{e}(Q_i, P_{pub}) \end{aligned}$$

Assume the elements in the aggregate signature $sig = (\phi, \sigma, T_1, T_2, \cdots, T_n)$ are the same as in *Aggegate* . Then,

$$Q = H_2(\phi)$$

for $i = 1, \cdots, n$

$$Q_i = H_1(ID_i), \quad c_i = H_3(m_i, Q, Q_i, T_i)$$

$$\begin{aligned} \hat{e}(\sigma, P) &= \hat{e}(\sum_{i=1}^{n} c_i \alpha_i Q + S_i, P) \\ &= \hat{e}(\sum_{i=1}^{n} c_i \alpha_i P, Q) \hat{e}(\sum_{i=1}^{n} sQ_i, P) \\ &= \hat{e}(\sum_{i=1}^{n} c_i T_i, Q) \hat{e}(\sum_{i=1}^{n} Q_i, P_{pub}) \end{aligned}$$

This scheme can be adapted to get a full aggregate signature scheme if we adopt the global state method in the Gentry's work [2]. Using the method, each public random element T_i of G_1 can be computed based on one global state ϖ and the signer's public seed P_i. P_i can be pre-published by the signer for each time period ϕ , and ϖ is uniquely chosen for each aggregate signature. $T_i = P_i' = H_4(ID_i, P_i, \varpi)$, $1 \le i \le n$. Thus, each signature verifier can publicly compute T_i and T_i can be omitted from the signature result. In addition, P_i is with respect to multiple aggregate signatures and ϖ is shared by all signers, which results in full aggregation.

Definition 4 (FIBAS). Full identity-based aggregate signature Π_{FAS}. Π_{FAS} *is made up of seven algorithms as follows:*

Setup(1^{κ})

As Π_{AS} *except that* $H_4 : \{0,1\}^* \to G_1$.

KeyGen $(s, param, ID_i)$

As Π_{AS}.

SeedPub $(\phi, param, ID_i)$

Each user ID_i *publishes her public seed* P_i *for the time period* ϕ, *where*

$s_i \in_R Z_p^*$, $P_i = s_i P$.

Signing $(m_i, S_i, param)$

The first signer chooses a string ϖ *that it has never used before. Each subsequent signer checks that it has not used the string* ϖ *chosen by the first signer [2].*

The user ID_i *computes as follows:*

gets the current time period ϕ;

computes $P_i' = H_4(ID_i, P_i, \varpi)$;

computes $Q = H_2(\phi)$;

computes $c_i = H_3(m_i, Q, Q_i, P_i, P_i')$;

computes $\sigma_i = c_i s_i Q + S_i + s_i P_i'$;

returns $sig_i = (\phi, \varpi, \sigma_i)$.

Verificati on (m_i, sig_i, ID_i)

Any user verifies signature sig_i *as follows:*

parses $sig_i = (\phi, \varpi, \sigma_i)$;

computes $Q_i = H_1(ID_i)$;

computes $Q = H_2(\phi)$;

computes $P_i' = H_4(ID_i, P_i, \varpi)$;

computes $c_i = H_3(m_i, Q, Q_i, P_i, P_i')$;

checks $\hat{e}(\sigma_i, P) \stackrel{?}{=} \hat{e}(c_i P_i, Q)\hat{e}(Q_i, P_{pub})\hat{e}(P_i, P_i') = \hat{e}(P_i, c_i Q + P_i')\hat{e}(Q_i, P_{pub})$;

if true, returns "valid", else return "invalid".

Aggregate $(sig_1, sig_2, \cdots)$

For the aggregation subset of signers, we assign to each signer an index i, *ranging from 1 to* n. *Anyone can aggregate a collection of individual signatures* $sig_i = (\phi, \varpi, \sigma_i)$, $i = 1$ *to* n *if they use the same string* ϖ.

The computation is $\sigma = \sum_{i=1}^{n} \sigma_i$.

The aggregate signature is $sig = (\phi, \varpi, \sigma)$.

Aggregate Verificati on $(sig, (m_1, ID_1), \cdots, (m_1, ID_n))$

Any user verifies signature sig *as follows:*

parses $sig = (\phi, \varpi, \sigma)$;

computes $Q = H_2(\phi)$;

computes

for $i = 1, \cdots, n$

$Q_i = H_1(ID_i)$,

$P_i' = H_4(ID_i, P_i, \varpi)$,

$c_i = H_3(m_i, Q, Q_i, P_i, P_i')$;

checks $\hat{e}(\sigma, P) \stackrel{?}{=} \hat{e}(\sum_{i=1}^{n} c_i P_i, Q)\hat{e}(\sum_{i=1}^{n} Q_i, P_{pub})\prod_{i=1}^{n} \hat{e}(P_i, P_i')$;

if true, returns "valid", else return "invalid".

We prove the security of Π_{AS}. The proof of Π_{FAS} is similar to the former and is not given.

5. SECURITY PROOFS

Theorem 1. *If the CDH assumption is true, then* Π_{AS} *is existentially unforgeable under adaptive chosen message attacks in the random oracle model, where the adversary asks at most* q_k *times Private-key queries,* q_s *times Signature queries,* q_{H_1} *times* H_1 *queries,* q_{H_2} *times* H_2 *queries, and* q_{H_3} *times* H_3 *queries.*

Proof. The correctness of the scheme is straightforward. So we prove it is unforgable. If there is an adversary A to succeed in attacking the scheme with an advantage ε, then we can construct a probabilistic polynomial time algorithm B to solve the CDH problem with non-negligible probability. Let $P_a = aP \in G_1$, $P_b = bP \in G_1$ be a random instance of the CDH problem taken as input by B. B computes abP by A as a sub-algorithm. B initializes A as follows:

Setup: B setups G_1, G_2, e, $P \in G_1$, and H_1, H_2, H_3. Let the master-key $ms = a$ (*unknown*) and $P_{pub} = P_a = aP$. Send to A the parameter $param = (G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3)$.

The adversary A then starts performing queries such as those described in definition 2. These queries are answered by B as follows:

Queries on oracle H_1**:** B maintains a list L_1 of tuples $(ID_i, \lambda_i, Coin)$. This list is initially empty. When a new (ID_i) is submitted to the random oracle H_1, B flips a coin $Coin_i \in \{0,1\}$ that yields 0 with probability $1-\delta$ and 1 with δ. B then picks $\lambda_i \in_R Z_p^*$. If $Coin_i = 0$, then the hash value $H_1(ID_i)$ is defined as $H_1(ID_i) = \lambda_i P$. If $Coin_i = 1$, then $H_1(ID_i) = \lambda_i P_b = \lambda_i bP$. In both cases, B inserts a tuple $(ID_i, \lambda_i, Coin_i)$ in a list L_1. If the request has been asked before, the same answer from L_1 is given.

Queries on oracle H_2**:** B maintains a list L_2 of tuples $(\phi, \lambda, Coin_Q)$. This list is initially empty. When a new (ϕ) is submitted to the random oracle H_2, B flips a coin $Coin_Q \in \{0,1\}$ that yields 0 with probability δ and 1 with $1-\delta$. B then picks $\lambda \in_R Z_p^*$. If $Coin_Q = 0$, then the hash value $H_2(\phi)$ is defined as $H_2(\phi) = \lambda P$. If $Coin_Q = 1$, then $H_2(\phi) = \lambda P_b = \lambda bP$. In both cases, B inserts a tuple $(\phi, \lambda, Coin_Q)$ in a list L_2. If the request has been asked before, the same

answer from L_2 is given.

Queries on oracle H_3 : When a new message (m_i, Q, Q_i, T_i) is submitted to the random oracle H_3, B picks $c_i \in_R Z_p^*$, and defines the hash value $H_3(m_i, Q, Q_i, T_i)$ as $H_3(m_i, Q, Q_i, T_i) = c_i$. B inserts a tuple $((m_i, Q, Q_i, T_i), c_i)$ in a list L_3. If the request has been asked before, the same answer from L_3 is given.

Private Key Queries: When a new (ID_i) is submitted to the $Private-KeyGen$ oracle, B makes a H_1 query on ID_i and finds the tuple $(ID_i, \lambda_i, Coin_i)$ on L_1, then does as follows:

(1) If $Coin_i = 1$, abort.

(2) Else computes $Q_i = \lambda_i P$, $S_i = \lambda_i P_{pub} = \lambda_i aP$, returns S_i as a private key. B inserts a tuple (ID_i, S_i) in a list $L_{privateKey}$.

If the request has been asked before, the same answer from $L_{privateKey}$ is given.

Signature Queries: When A queries the signature oracle on a message m_i under the identity ID_i. B makes a H_1 query on ID_i and finds the tuple $(ID_i, \lambda_i, Coin_i)$ on L_1. B also makes a H_2 query on time period ϕ and finds the tuple $(Q, \lambda, Coin_Q)$ on L_2.

(1) If $Coin_i = 1$ and $Coin_Q = 0$, abort.

(2) If $Coin_i = 1$ and $Coin_Q = 1$, $Q_i = \lambda_i bP$ and $Q = \lambda bP$, picks $c_i, x_i \in_R Z_p^*$, computes $T_i = -\lambda_i / (c_i \lambda) aP + x_i P$, $\sigma_i = c_i x_i Q$, and gets $sig_i = (\phi, \sigma_i, T_i)$. Finally, B lets the answer of the random oracle H_3 is c_i when it takes as input (m_i, Q, Q_i, T_i). If this causes a collision, i.e., if B previously set the oracle at this point to some other c', the simulation halts and fails. $sig_i = (\phi, \sigma_i, T_i)$ is returned to A and appears as a valid signature from A's point of view because

$$\begin{aligned}\hat{e}(c_i T_i, Q)\hat{e}(Q_i, P_{pub}) &= \hat{e}(-\lambda_i / \lambda aP + c_i x_i P, \lambda bP)\hat{e}(\lambda_i bP, aP) \\ &= \hat{e}(c_i x_i P, \lambda bP) \\ &= \hat{e}(c_i x_i Q, P) \\ &= \hat{e}(\sigma_i, P)\end{aligned}$$

(3) Otherwise, B computes the signature sig_i as the standard Signing algorithm.

Forgery: Eventually, A produces a fake signature $sig = (\phi, \sigma, T_1, T_2, \cdots, T_n)$ with non-negligible advantage ε for messages $(m_1, m_2, \cdots, m_n)$ under identities $(ID_1, ID_2, \cdots, ID_n)$. Without loss of generality, we assume the private key of ID_1 is not queried by A. The individual signature $(m_1, \bullet, T_1)$ in σ is not queried by A either.

B recovers the corresponding n tuples $(ID_i, \lambda_i, Coin_i)$ on the list L_1. B proceeds only if $Coin_Q = 0$, $Coin_1 = 1$ and $Coin_i = 0$ for $2 \le i \le n$. Otherwise, abort. Since $Coin_Q = 0$, it follows that $Q = \lambda P$. Since $Coin_1 = 1$, it follows that $Q_1 = \lambda_1 bP$. Since $Coin_i = 0$ for $2 \le i \le n$, it follows that $Q_i = \lambda_i P$. The aggregate signature $sig = (\phi, \sigma, T_1, T_2, \cdots, T_n)$ satisfies the aggregate verification equation

$$\hat{e}(\sigma, P) = \hat{e}(\sum_{i=1}^{n} c_i T_i, Q)\hat{e}(\sum_{i=1}^{n} Q_i, P_{pub})$$

where $c_i = H_3(m_i, Q, Q_i, T_i),\ i = 1, \cdots, n$.

Then, B knows that

$$\hat{e}(\sigma, P) = \hat{e}(\sum_{i=1}^{n} c_i T_i, \lambda P)\hat{e}(\sum_{i=2}^{n} Q_i, P_{pub})\hat{e}(Q_1, P_{pub})$$

$$= \hat{e}(\sum_{i=1}^{n} c_i T_i, \lambda P)\hat{e}(\sum_{i=2}^{n} \lambda_i P, aP)\hat{e}(\lambda_1 bP, aP)$$

Thus,

$$\hat{e}(\sigma - \sum_{i=1}^{n} c_i \lambda T_i - \sum_{i=2}^{n} \lambda_i aP, P) = \hat{e}(\lambda_1 bP, aP) \Rightarrow$$

$$\hat{e}(\frac{1}{\lambda_1}\sigma - \frac{\lambda}{\lambda_1}\sum_{i=1}^{n} c_i T_i - \frac{\sum_{i=2}^{n} \lambda_i}{\lambda_1} aP, P) = \hat{e}(bP, aP)$$

Finally, B outputs the required abP as $\frac{1}{\lambda_1}\sigma - \frac{\lambda}{\lambda_1}\sum_{i=1}^{n} c_i T_i - \frac{\sum_{i=2}^{n} \lambda_i}{\lambda_1} aP$

This completes the description of algorithm B. To complete the proof, we show that B solves the given instance of CDH problem with non-negligible probability**.** First, we analyze the four events needed for B to succeed:

- ***E1:*** B does not abort during answering private key queries.
- ***E2:*** B does not abort during answering signing queries.
- ***E3:*** A generates a valid aggregate signature forgery.
- ***E4: E3*** occurs, and $Coin_Q = 0$, $Coin_1 = 1$, $Coin_i = 0$ for $2 \le i \le n$.

B succeeds if all of these events happen. The probability $\Pr[E1 \wedge E2 \wedge E3 \wedge E4]$ can be computed as

$$\Pr[E1 \wedge E2 \wedge E3 \wedge E4]$$
$$= \Pr[E1]\Pr[E2 \mid E1]\Pr[E3 \mid E2 \wedge E1]\Pr[E4 \mid E3 \wedge E2 \wedge E1]$$

Claim 1. *The probability that* B *does not abort during answering private key queries is at least* $(1-\delta)^{q_k}$. *Hence we have* $\Pr[E1] \geq (1-\delta)^{q_k}$.

Proof. As $\Pr[Coin_i = 0] = (1-\delta)$, the probability that B does not abort is $(1-\delta)$ for one private key query. B makes at most q_k queries to the private key queries. Hence the probability that B does not abort during answering private key queries is at least $(1-\delta)^{q_k}$. $\Pr[E1] \geq (1-\delta)^{q_k}$

Claim 2. *The probability that* B *does not abort during answering signature queries is* $(1-\delta^2)^{q_s}$, *because when* $Coin_Q = 0$ *and* $Coin_1 = 1$, *the abort happens during answering one signature queries where* $\Pr[Coin_1 = 1] = \delta$ *and* $\Pr[Coin_Q = 0] = \delta$. *Hence we have* $\Pr[E2 \mid E1] = (1-\delta^2)^{q_s}$.

Claim 3. *Suppose* B *does not abort during answering signature queries and private key queries, then* A*'s view is identical to its view in the real attack. Hence, we have* $\Pr[E3 \mid E1 \wedge E2] \geq \varepsilon$.

Claim 4. *The probability that* B *does not abort after* A *outputting a valid forgery is at least* $\delta(1-\delta)^{n-1}$, *because* $Coin_Q = 0$, $Coin_1 = 1$ *and* $Coin_i = 0$ for $2 \leq i \leq n$ *during generating the forgery signature where* $\Pr[Coin_Q = 0] = \delta$, $\Pr[Coin_i = 0] = (1-\delta)$ *and* $\Pr[Coin_1 = 1] = \delta$. *Hence, we have* $\Pr[E3 \mid E1 \wedge E2] \geq \delta^2(1-\delta)^{n-1}$

Totally, we have

$$\Pr[E1 \wedge E2 \wedge E3 \wedge E4] \geq \delta^2(1-\delta)^{n-1}(1-\delta^2)^{q_s}(1-\delta)^{q_k}\varepsilon$$

$$= \delta^2(1-\delta^2)^{q_s}(1-\delta)^{q_k+n-1}\varepsilon = f(\delta)$$

How to maximize $\delta^2(1-\delta^2)^{q_s}(1-\delta)^{q_k+n-1}\varepsilon$? We differentiate $\delta^2(1-\delta^2)^{q_s}(1-\delta)^{q_k+n-1}\varepsilon$, let it be zero, then get

$$(2q_s + q_k + n + 1)\delta^2 + (q_k + n - 1)\delta - 2 = 0.$$

Because $(q_k + n - 1)^2 + 8(2q_s + q_k + n + 1) > 0$, the equation has two real roots, and the valid root δ satisfies $1/(2q_s + q_k + n + 1) < \delta < 1/(2q_s + q_k + n + 1)^{1/2}$, i.e. $\max(f(\delta)) \geq \max(f(1/(2q_s + q_k + n + 1)), f(1/(2q_s + q_k + n + 1)^{1/2}))$

$= \frac{\varepsilon}{(2q_S + q_k + n + 1)^2 e^2}$ (e is the base of natural logarithms). Assume $q_s = q_k + n + 91/16$ to illustrate the result (q_s could be others as long as the square root of $(q_k + n - 1)^2 + 8(2q_s + q_k + n + 1)$ is simple). Then, we get $\delta = \frac{11}{6q_k + 6n + 99/4}$ where $\delta^2(1-\delta^2)^{q_s}(1-\delta)^{q_k+n-1}\varepsilon$ is maximized at $\frac{121\varepsilon}{(6q_k + 6n + 99/4)^2 e^{11/6}}$ with sufficient large $q_k + n$. It comes that the advantage for B to solve the CDH problem successfully is non-negligible.

6. PERFORMANCE COMPARISON

In Table 1, we compare our re-encryption scheme with other schemes. Here $|Z_p^*|$, $|G_1|$ and $|G_2|$ denote the bit-length of an element in groups Z_p^*, G_1 and G_2 respectively. We use t_p, t_s and t_e to represent the computation cost of a

bilinear pairing, a scalar multiplication and an exponentiation respectively.

The comparison results indicate that our aggregate signature scheme is non-interactive with constant pairing operations, and is appropriate in terms of both computation and communication costs. Our scheme has almost equivalent signing computation performance to Wang Zhu's scheme [12], Wen's scheme [10], Seung's scheme [17], and Shi Cui's schme [9], which are as efficient as ours during verification with constant pairing operations. Unfortunately, universal forgery is possible in those constructs. Yu's IBAS [18] is similar to ours and achieves constant computational cost in pairings. However, an interaction protocol must be executed among signers during signing. Zhang's CLAS [13] is a secure certificateless aggregate signature scheme. But it is much slower than ours for its linear pairing operations in verification phase. Gentry's identity-based aggregate signature scheme [2] is efficient as well as secure. But it does not allow for concurrent signing, and requires the first signer chooses a one-time global string. All signers need to store the global string as a state to check whether other chosen strings are unique. Our scheme can be concurrently executed and does not require maintaining history states. In addition, our scheme has comparable computation performance to Gentry's.

7. CONCLUSIONS

In this paper, the strong security model for identity-based aggregate signature schemes is presented as well as its efficient construction. Our scheme needs only constant pairing operations during signature verification and can be concurrently used without number restriction. The individual ones in an aggregate signature can be used independently. The security of our scheme is proved in the random oracle model, and the performance is comparable to others. Therefore, the model and constructions will provide strong building blocks for some practical applications.

REFERENCES

[1] Jung Yeon Hwang, Dong Hoon Lee, and Moti Yung. Universal forgery of the identity-based sequential aggregate signature scheme. ASIACCS 2009, pp. 157-160, 2009.

[2] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. Public Key Cryptography, volume 3958 of Lecture Notes in Computer Science, pp. 257-273, 2006.

[3] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. J. Cryptology, 13(3): pp. 361-396, 2000.

[4] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. EUROCRYPT, volume 2656 of Lecture Notes in Computer Science, pp. 416-432, 2003.

[5] HyoJin Yoon, Jung Hee Cheon, and Yongdae Kim. Batch verifications with id-based signatures. ICISC, volume 3506 of Lecture Notes in Computer Science, pp. 233–248, 2004.

[6] Xiangguo Cheng, Jingmei Liu, and XinmeiWang. Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. ICCSA (4), volume 3483 of Lecture Notes in Computer Science, pp. 1046-1054, 2005.

[7] Jing Xu, Zhenfeng Zhang, and Dengguo Feng. Identity-based aggregate signatures from bilinear pairings. CANS, volume 3810 of Lecture Notes in Computer Science, pp. 110-119, 2005.

[8] Javier Herranz. Deterministic identity-based signatures for partial aggregation. Comput. J., 49(3), pp. 322–330, 2006.

[9] Shi Cui, Pu Duan, and Choong Wah Chan. An efficient identity-based signature scheme with batch verifications. Infoscale, volume 152 of ACM International Conference Proceeding Series, pp. 22, 2006.

[10] Yiling Wen and Jianfeng Ma. An aggregate signature scheme with constant pairing operations. CSSE, pp. 830-833, 2008.

[11] S. Sharmila Deva Selvi, S. Sree Vivek, J.Shriram, S.Kalaivani, and C.Pandu Rangan. Security analysis of aggregate signature and batch verification signature schemes. http://eprint.iacr.org/2009/290, 2009.

[12] Zhu Wang, Huiyan Chen, Ding feng Ye, and Qian Wu. Practical identity-based aggregate signature scheme from bilinear maps. Shangai Jiao Tong University Press, volume 13(6), pp. 684-687 2008.

[13] Lei Zhang and Futai Zhang. A new certificateless aggregate signature scheme. Computer Communications, pp. 1079-1085, 2009.

[14] S. Sharmila Deva Selvi, S. Sree Vivek, J.Shriram, and C.Pandu Rangan. Efficient and provably secure identity based aggregate signature schemes with partial and full aggregation. http://eprint.iacr.org/2010/461, 2010.

[15] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In Proc. of CRYPTO'01, volume 2139, pp. 213-229, 2001.

[16] P. Barreto, H. Kim, B. Bynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In Proc. CRYPTO'02, pp. 354-368, 2002.

[17] Seung-Hyun Seo, Jung Yeon Hwang, Kyu Young Choi, and Dong Hoon Lee. Identity-based universal designated multi-verifiers signature schemes. Comput. Stand. Interfaces, 30(5), pp. 288–295, 2008.

[18] Yike Yu, Xuefeng Zheng, Hua Sun. A new ID-based aggregate signature with constant pairing operations. Networks Security Wireless Communications and Trusted Computing (NSWCTC 2010), pp. 188-191, 2010.

Table 1. Performance Comparison

Scheme	Sign cost	Verify cost	Signature Length	Const Pairing	Security
Our Scheme	$2n\,t_s$	$n\,t_s+3\,t_p$	$(n+1)\lvert G_1\rvert+\lvert Z_p^*\rvert$	Yes	Provable
Our Full Scheme	$2n\,t_s$	$2n\,t_s+(n+3)\,t_p$	$\lvert G_1\rvert+2\lvert Z_p^*\rvert$	No but with verifiable single ones	Provable
Zhang's CLAS	$3n\,t_s$	$(n+3)\,t_p$	$(n+3)\lvert G_1\rvert$	No	Provable
Yu's IBAS	$3n\,t_s$	$n\,t_s+3\,t_p$	$(n+1)\lvert G_1\rvert$	Yes, need Interaction	No
Wang Zhu's	$3n\,t_s$	$n\,t_s+3\,t_p$	$(n+1)\lvert G_1\rvert$	Yes	No
Wen's	$2n\,t_s$	$n\,t_s+2\,t_p$	$2\lvert G_1\rvert$	Yes	No
Seung's	$3n\,t_s$	$n\,t_s+2\,t_p$	$2n\lvert G_1\rvert$	Yes	No
Shi Cui's	$n\,t_s+n\,t_e$	$n\,t_s+n\,t_e+3\,t_p$	$n\lvert G_1\rvert+n\lvert G_2\rvert$	Yes	No
Gentry's	$3n\,t_s$	$n\,t_s+3\,t_p$	$2\lvert G_1\rvert$	Yes, but with non-verifiable single ones	Provable