Security Analysis of Aggregate signature and Batch verification signature schemes

S. Sharmila Deva Selvi[1], S. Sree Vivek[⋆,1], J.Shriram [⋆⋆2], S.Kalaivani[2], and C.Pandu Rangan[⋆,1]

[1] {sharmila,svivek}@cse.iitm.ernet.in, prangan@iitm.ac.in
Indian Institute of Technology Madras
Theoretical Computer Science Laboratory
Department of Computer Science and Engineering
Chennai, India

[2] shriram139@gmail.com,kalaivani.siva@gmail.com
National Institute of Technology Trichy
Department of Computer Science and Engineering
Chennai, India

Abstract. An Identity (ID)-based signature scheme allows any pair of users to communicate securely and to verify each others signatures without exchanging public key certificates. An aggregate signature scheme is a digital signature scheme which supports aggregation of signatures. Given n signatures on n distinct messages signed by n distinct users it is possible to aggregate all these signatures into a single signature. This signature will convince the verifier that the n users signed the n messages. Batch verification is a method to verify multiple signatures at once. Aggregate signature is useful in reducing both communication and computation cost. In this paper we show the weakness in some of the aggregate Signature schemes and batch verification scheme.

Keywords: Identity Based Signature, Aggregate Signatures, Batch Verification, Cryptanalysis.

1 Introduction

The concept of an identity based (ID-based) cryptosystem was introduced by Shamir in 1984 [9]. The distinguishing characteristic of identity-based cryptography is the ability to use any string as a public key. The corresponding private key can only be derived by a trusted Private Key Generator (PKG) which uses a master secret key to generate them. An identity-based cryptosystem removes the need for users to look up and verify the public key before verifiing an signature and message. It avoids the overhead of storage of certificates of public keys and also the public key of the user can easily be derived from the identity which uniquely defines the user. Identity based cryptography provides a more convenient alternative to conventional public key infrastructure.

Identity Based signature scheme are designed in such a way that a user can authenticate a message by producing a unique digital signature on the message using his private key and any other user in the system can verify the authenticity of the signature by just knowing the identity of the user who signed

⋆ Work supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation sponsored by Department of Information Technology, Government of India.

⋆⋆ Work supported by IITM Summer Fellowship 2009

it. Various identity based signature construct have been proposed [2] [14] [7]. The major constraints in the field of cryptography are the communication and computational complexity. One tries to reduce both as much as possible to gain maximum advantage. There are many variations of signature schemes. One such variation aggregate signature.

An important factor to be considered for cryptographic schemes to be practically applicable is efficiency. Efficiency includes both communication and computation efficiency. The growth in bandwidth of communication networks, seems to have more constraints, which increases the importance of communication efficiency. Similarly, in wireless devices such as PDAs, battery life is often a bottleneck when compared to the processing speed, which hinders the computations that includes very complex operations and large numbers. In banking services or electronic commerce one server may have to verify many signatures simultaneously. In order to enhance the efficiency of verification and reduce the communication over-head we use aggregate signatures. Aggregate signatures was first introduced by Boneh et al. [1] and a realization was also proposed by them.

Aggregation can be used to reduce the certificate chains in PKI settings. Apart from compactness other advantages of aggregate signatures are also present. An adversary cannot remove a single signature from the collection of signatures without detection. Two kinds of aggregation schemes exist. One is flexible aggregation where one can aggregate n sigantures on n messages in any order. Other is sequential aggregation where the n^{th} signer aggregates his/her signature into the aggregate signature formed by previous $(n-1)$ users. Sequential aggregation is comparitively a weaker model but has many real life applications in corporate scenarios.

Since aggregate signature was presented by Boneh et al., several schemes have been proposed so far [5] [3] [6] [13] [12] [4]. Gentry and Ramzan in [5] presented the most efficient ID-based aggregate scheme which requires only three pairing computations but it has a weakness, a legal user of the system can forge an aggregate signature if given the signatures of some user i on some message m_i. Also all the users have to agree upon a common randomness value which makes it unsuitable for most real life scenarios. Wen et al. in [11] proposed an aggregate signature scheme with constant pairing operation,but this signture scheme has a forgeability attack which we have pointed in our paper. Wang Zhu et al. in [10] proposed an practical aggregate signature scheme with constant pairing operation but was able to achieve only partial aggregation. But that scheme isnt insider secure. A valid user of the system will be able to forge a signature on any message by any user by just seeing a random singned message by the corresponding user. We show the possible attack in our paper.

The rest of the paper is organised as follows. In section 2 we discuss all the preliminaries and the computational assumptions which we take into considerations. In section 3 we discuss the generic model for aggregate signature. In section 4 we discuss the security model for aggregate signatures. In section 5 we review the Wang Zhu scheme [10] and propose a break for it in section 6. In section 7 we review the Wen et al. scheme [11] and show the attack possible in the scheme in section 8. In Section 9, we review Seung et al.'s [8] scheme and we show the forgeabilty attack on this scheme in section 10. In section 11 and 12 we discuss Shi Cui et al.'s [4] and we show how universal forgery is possible in their batch verification construct. Then in section 13 we post the conclusions and the current open problems in this area.

2 Preliminaries

2.1 Bilinear Pairing

Let $\mathbb{G}$ be an additive cyclic group generated by P, with prime order q, and $\mathbb{G}_T$ be a multiplicative cyclic group of the same order q. Let $\hat{e}$ be a pairing defined as $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. It satisfies the following properties.

- **Bilinearity** Let $P, Q \in \mathbb{G}$ and $a, b \in \mathbb{Z}_q^*$ then $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
- **Non Denegrate** Let $P \in \mathbb{G}$ then $\hat{e}(P, P) \neq 1$.
- **Easily Computable** Let $P, Q \in \mathbb{G}$ then $\hat{e}(P, Q)$ must be easily and efficiently computable.

2.2 Computational Assumptions

In this section, we review the computational assumptions related to bilinear maps that are relevant to the protocol we discuss.

Bilinear Diffie-Hellman Problem (BDHP)

Given $(P, aP, bP, cP) \in \mathbb{G}^4$ for unknown $a, b, c \in \mathbb{Z}_q^*$, the BDH problem in $\mathbb{G}$ is to compute $\hat{e}(P, P)^{abc}$.

The advantage of any probabilistic polynomial time algorithm A in solving the BDH problem in $\mathbb{G}$ is defined as

$$Adv_{\mathbb{A}}^{BDH} = Pr[\mathbb{A}(P, aP, bP, cP) = \hat{e}(P, P)^{abc} | a, b, c \in \mathbb{Z}_q^*]$$

The BDH Assumption is that, for any probabilistic polynomial time algorithm $\mathbb{A}$, the advantage $Adv_{\mathbb{A}}^{BDH}$ is negligibly small.

Decisional Bilinear Diffie-Hellman Problem (DBDHP)

Given $(P, aP, bP, cP, \alpha) \in \mathbb{G}^4 \times \mathbb{G}_T$ for unknown $a, b, c \in \mathbb{Z}_q^*$, the $DBDH$ problem in $\mathbb{G}$ is to decide $\alpha = \hat{e}(P, P)^{abc}$.

The advantage of any probabilistic polynomial time algorithm A in solving the DBDH problem in $\mathbb{G}$ is defined as

$$Adv_{\mathbb{A}}^{DBDH} = |Pr[\mathbb{A}(P, aP, bP, cP, \hat{e}(P, P)^{abc}) = 1] - Pr[\mathbb{A}(P, aP, bP, cP, \alpha) = 1]|$$

The DBDH Assumption is that, for any probabilistic polynomial time algorithm $\mathbb{A}$, the advantage $Adv_{\mathbb{A}}^{DBDH}$ is negligibly small.

Computation Diffie-Hellman Problem (CDHP)

Given $(P, aP, bP) \in \mathbb{G}^3$ for unknown $a, b \in \mathbb{Z}_q^*$, the $CDHP$ problem in $\mathbb{G}$ is to compute abP.

The advantage of any probabilistic polynomial time algorithm A in solving the CDH problem in $\mathbb{G}$ is defined as

$$Adv_{\mathbb{A}}^{CDH} = Pr[\mathbb{A}(P, aP, bP) = abP | a, b \in \mathbb{Z}_q^*]$$

The CDH Assumption is that, for any probabilistic polynomial time algorithm A, the advantage Adv_{A}^{CDH} is negligibly small.

3 Generic Model

An Identity based aggregate signature scheme(IBAS) consists of five algorithms as follows.

- **Setup :** The private key generator(PKG) provides the security parameter λ as the input to this algorithm and generates the system parameters $params$ and the master private key Msk. PKG publishes the $params$ and keeps the Msk secret.
- **KeyGen :** The user U_i provides his identity ID_i to PKG. The PKG runs this algorithm with identity $ID_i, params$ and Msk as the input and PKG outputs the private key D_i to user U_i through a secure channel.
- **Signing :** For generating a signature on a message m_i, the user U_i provides his ID_i, his private key D_i, $params$ and message m_i as input to this algorithm. This algorithm generates a valid signature σ_i on message m_i by user U_i.
- **Verify :** This algorithm on input of a signature σ on message m by user with identity ID checks whether σ is a valid signature on message m by ID. If true it outputs "$Valid$", else it outputs "$Invalid$".
- **Aggregate :** On receiving the various signatures $(\sigma_i)_{i=1}^n$ from different users $(U_i)_{i=1}^n$, any third party or one of the signers can run this algorithm and generate the aggregate signature σ_{agg} and the message,identity pairs $(m_i, ID_i)_{i=1}^n$.
- **Aggregate Verification :** This algorithm on input of an aggregate signature σ_{agg},the list of message,identity pairs $(m_i, ID_i)_{i=1}^n$ and the $params$ checks whether σ_{agg} is a valid aggregate signature on m_i by ID_i $\forall$ $i = 1$ to n. If true it outputs "$Valid$", else output "$Invalid$".

4 Security Model

4.1 Unforgeability

An IBAS scheme is secure against existential forgery under adaptive-chosen-identity and adaptive-chosen-message attack if no probabilistic polynomial time adversary $\mathbb{A}$ has non-negligible advantage in the following game.

- **Setup phase :** The challenger $\mathbb{C}$ runs the setup algorithm and generates the $params$ and Msk. Challenger $\mathbb{C}$ gives the params to adversary $\mathbb{A}$.
- **KeyGen oracle :** When $\mathbb{A}$ makes a query with ID_i, $\mathbb{C}$ outputs D_i, the private key of ID_i to $\mathbb{A}$ provided $\mathbb{C}$ knows the secret for the queried identity. Else it aborts.
- **Signing oracle :** When $\mathbb{A}$ makes a signing query with ID_i, message m_i, $\mathbb{C}$ outputs a valid signature σ_i on m_i by ID_i.
- **Forgery phase :** When the adversary $\mathbb{A}$ finishes all the queries, $\mathbb{A}$ outputs an aggregate signature $(\sigma)_{i=1}^n$ from the users $(ID_i)_{i=1}^n$ on messages $(m_i)_{i=1}^n$ where there exists a $ID_T \in (ID_i)_{i=1}^n$ who is one of the targer identities. The adversary $\mathbb{A}$ wins the game if σ_{agg} is a valid aggregate signature and $\mathbb{A}$ hasn't queried for signing query for (ID_T, m_T) pair for which it had forged.

5 Review of Practical Identiy-Based Aggregate Signature from Bilinear Maps

In this section we review new aggregate signature scheme by Wang Zhu et al.[10]. The paper claim the scheme to be computationally efficient with constant pairings in verification. The scheme is based on weak CDH problem which is defined as follows.

5.1 Definition of wCDH

The wCDH in $\mathbb{G}_1$ is defined as follows : Given (P, aP, bP, b^2P) for unknown a,b ϵ $\mathbb{Z}_l$ compute abP.

5.2 Construction

The scheme consists of six algorithms. The first four algorithms are similar to a ordinary signature and the last two algorithms provide the aggregating capability.

- **Setup** Given a security parameter $l \ \epsilon \ \mathbb{Z}$, the private key generator (PKG) runs the setup algorithm which outputs two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p, a generator P of $\mathbb{G}_1$, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$, master secret key $s \in \mathbb{Z}_p^*$. PKG computes $P_{pub} = sP$ and $P_{pub^2} = s^2P$. PKG also chooses cryptographic hash functions. $H_1 : \{0,1\}^* \longrightarrow \mathbb{G}_1, H_2 : \{0,1\}^* \longrightarrow \mathbb{Z}_p^*$.
  The system's public parameters are
  Param $= < p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, P_{pub^2}, H_1, H_2 >$.

- **Extract** The PKG provides the user's identity ID_i and the master secret key s as input to this algorithm and receives his public key $D_{ID_i} = H_1(ID_i)$, private key $s_{ID_i} = sD_{ID_i}$. The PKG sends D_{ID_i} through a public channel and S_{ID_i} through a secure channel to the user.

- **Standard signature** To sign the message m_i, the user sends his identity ID_i, message m_i as the input to this algorithm. The algorithm follows the steps below:
  - Randomly choose $x_i \in Z_p^*$ and compute $T_i = x_i P_{pub}$.
  - Compute $h_i = H_2(ID_i, m_i, T_i)$
  - Compute its signature $\sigma_i = (S_i, T_i)$ where $S_i = x_i P + h_i s_{ID_i}$.

- **Verify** Given a signature $\sigma_i = (S_i, T_i)$ on message m_i by user with identity ID_i, compute
  $h_i = H_2(ID_i, m_i, T_i)$
  accept if $\hat{e}(S_i, P_{pub}) \stackrel{?}{=} \hat{e}(P, T_i)\hat{e}(h_i D_{ID_i}, P_{pub^2})$.

- **Aggregate Signature** For the aggregating subset of users $\mathcal{U}$, assign to each user an index i, ranging from 1 to $k \ \epsilon \ |U|$. This algorithm takes as input User U_i's identity ID_i, a signature (S_i, T_i) on a message m_i. It computes $S = \Sigma_{i=1}^k S_i$ and outputs the aggregate signature is $\sigma = (S, T_1, T_2, ...T_k)$.

– **Aggregate Signature verify** This algorithm on receiving the inputs aggregate signature $(S, T_1, T_2, ...T_k)$ for an aggregating subset of users U, and the list of <identity,message> pairs $\{ID_i, m_i\}_{i=1}^k$ indexed as before. To verify the aggregate signature it computes $h_i = H_2(ID_i, m_i, T_i)$ and accept if the following holds.
$\hat{e}(S, P_{pub}) \stackrel{?}{=} \hat{e}(P, \Sigma_{i=1}^k T_i)\hat{e}(\Sigma_{i=1}^k h_i D_{ID_i}, P_{pub^2})$.

6 Attack on Wang Zhu et al.'s scheme

We will show that universal forgery is possible in the aggregate scheme of Wang Zhu et al. Consider the adversary $\mathbb{A}$ queries for the private key of some identity $ID_{\mathbb{A}}$. Let the identity on whom the forgery is to be performed is ID. Assume the $\mathbb{A}$ in the training phase queries for a signature of ID on message m. The signature is of the form

$$T = xP_{pub} \tag{1}$$

$$S = xP + hs_{ID} \tag{2}$$

for a random unknown x.
The value of h can be calculated as $H_2(ID, m, T)$
The adversary does the following steps to sign a message m_1 on behalf of ID and aggregate with his own signature on some random message.

– Choose a random x_1 and set $T_1 = x_1 P_{pub}$.
– Set $h_1 = H_2(ID, m_1, T_1)$.
– Set $T^* = \frac{x}{h} h_1 P_{pub}$. (Dividing 3 by h and multiplying by h_1)
– Set $S^* = \frac{x}{h} h_1 P + h_1 s_{ID}$. (Dividing 4 by h and multiplying by h_1.)

– Set $S_1 = S^* + x_1 P = \frac{x}{h} h_1 P + h_1 s_{ID} + x_1 P$. The signature on m_1 by ID is $\sigma_1 = (S_1, T_1)$.
– Then the adversary $\mathbb{A}$ uses his private key $s_{\mathbb{A}}$ and signs a random message m_2 as follows.
– $T_2 = x_2 P_{pub} + T^*$
$= x_2 P_{pub} + \frac{x}{h} h_1 P_{pub}$.
– $h_2 = H_2(ID_{\mathbb{A}}, m_2, T_2)$.
– Set $S_2 = x_2 P + h_2 s_{\mathbb{A}}$
– Then aggregate the S_i components of both the signatures and the aggregate signature is of form $\sigma = (S, T_1, T_2)$ on messages m_1 and m_2. We can verify that this is a valid aggregate signature by the verification algorithm.

$$\begin{aligned}
\hat{e}(S, P_{pub}) &= \hat{e}(\frac{x}{h} h_1 P + h_1 s_{ID} + x_1 P + x_2 P + h_2 s_{\mathbb{A}}, P_{pub}) \\
&= \hat{e}(x_1 P, P_{pub})\hat{e}(\frac{x}{h} h_1 + x_2 P, P_{pub})\hat{e}(h_1 s_{ID} + h_2 s_{\mathbb{A}}, P_{pub}). \\
&= \hat{e}(T_1, P)\hat{e}(T_2, P)\hat{e}(\Sigma_{i=1}^2 h_i D_{ID_i}, P_{pub^2}). \\
&= \hat{e}(\Sigma_{i=1}^2 T_i, P_{pub})\hat{e}(\Sigma_{i=1}^2 h_i D_{ID_i}, P_{pub^2}).
\end{aligned}$$

Thus this aggregate signature passes the verification. The adversary can forge signature on any message by any user having just known a single signature of the user. This kind of break is possible only in aggregation. The base signature scheme is secure but its not secure when aggregated. Thus we have proved a universal forgery on Wang Zhu et al.'s [10] aggregate signature scheme.

7 Review of the Wen et al.'s Aggregate Signature Scheme

In this section we discuss the aggregate signature scheme as proposed by Wen et al. [11] which claims to be efficient with constant pairing operations in signature verification. The Wen et al. aggregate signature scheme [11] comprises five algorithms: KeyGen, Sign, Verify, Aggregate and Aggregate Verify. It uses two hash function $\hat{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. There are two generators P and $P^{'}$ of group $\mathbb{G}$. The system public parameters are $\{\mathbb{G}, \mathbb{G}_T, q, \hat{e}, P, P^{'}, \hat{H}, H\}$

- **Key Generation**: The PKG provides the identity of the user ID_i as input to this algorithm. The algorithm selects a random $s_{ID_i} \in \mathbb{Z}_q^*$ and $D_{ID_i} = s_{ID_i}P$. The public key is $D_{ID_i} \epsilon \mathbb{G}$. The private key is s_{ID_i}.

- **Signing**: The user U_i with identity ID_i who wishes to sign message m_i provides his private key $s_{ID_i} \epsilon \mathbb{Z}_q^*$, and a message $m_i \epsilon \{0,1\}_*$ as an input to this algorithm. The algorithm compute $\hat{h} = \hat{H}(m_i \| s_{ID_i})$ and $h = H(m_i)$. $T_i = \hat{h}P, S_i = (\hat{h} + hs_{ID_i})P^{'}$. The algorithm outputs the signature $\sigma_i = (S_i, T_i)$ on message m_i by user with identity ID_i.

- **Verification**: This algorithms takes as input the public key, D_{ID_i}, message, m_i, and a signature (S_i, T_i) and computes $h_i = H(m_i)$ and checks if $\hat{e}(S_i, P) \stackrel{?}{=} \hat{e}(P^{'}, T_i + h_i D_{ID_i})$. If yes it outputs "$Valid$" else it outputs "$Invalid$".

- **Aggregation**: For the aggregate signature for a set of users U assign to each user an index i, ranging from 1 to $k = |U|$. Each user u_i provides a signature (S_i, T_i) on a message m_i of his choice as an input to this algorithm. The aggregate algorithm computes $S = \Sigma_{i=1}^k S_i$ and $T = \Sigma_{i=1}^k T_i$ and outputs the aggregate signature is (S, T) and the set of message,identity pair $\{m_i, ID_i\}_{i=1}^k$.

- **Aggregate Verification**: This algorithm takes as input an aggregate signature (S, T) for an aggregating subset of users $U = \{u_i\}_{i=1}^k$, indexed as before, and the original messages m_i and public keys $D_I D_i$ for all users u_i. To verify algorithm first computes $h_i = H(m_i)$ and checks if
  $\hat{e}(S, P) \stackrel{?}{=} \hat{e}(P^{'}, T + \Sigma_{i=1}^k h_i D_i)$ holds. If yes it outputs "$Valid$" else it outputs "$Invalid$".

8 Attack on the Wen et al.'s Aggregate Signature Scheme

The signature scheme as proposed by Wen et al.'s[11] can be forged by any party who may not even be a legal user of the system.

- **Forgery of the signature:** The forger chooses a random message $m \epsilon \{0,1\}^*$ and computes the signature for some user U_i in the following manner,
  - Select a random $r \in Z_q^*$

- $h = H(m)$
- $T = rP - hD_{ID_i}$, where D_{ID_i} is public identity of the user whose signature he is forging.
- $S = rP^{'}$.

The forged signature passes the verification $\hat{e}(S,P) = \hat{e}(P^{'}, T + hD_{ID_i})$ because,
$\hat{e}(P, T + hD_{ID_i}) = \hat{e}(P^{'}, rP - hD_{ID_i} + hDID_i) = \hat{e}(P^{'}, rP) = \hat{e}(rP^{'}, P) = \hat{e}(S,P)$

– *Forgery of the Aggregate Signature:*The aggregate signature can also be forged by anyone in the similar manner. The forger can generate an aggregate signature for a set of users U. He can generate (T_i, S_i) on m_i of his choice for all user $u_i \ \epsilon \ U$ as mentioned above. He then aggregates, $S = \Sigma_{i=1}^{k} S_i$ and $T = \Sigma_{i=1}^{k} T_i$.

The forged aggregate signature passes the verification $\hat{e}(S,P) = \hat{e}(P^{'}, T + \Sigma_{i=1}^{k} h_i D_I D_i)$ because,

$$\begin{aligned}\hat{e}(P^{'}, T + \Sigma_{i=1}^{k} h_i D_I D_i) &= \hat{e}(P^{'}, \Sigma_{i=1}^{k} T_i + \Sigma_{i=1}^{k} h_i D_I D_i) \\ &= \hat{e}(P^{'}, \Sigma_{i=1}^{k} r_i P - \Sigma_{i=1}^{k} h_i D_I D_i + \Sigma_{i=1}^{k} h_i D_I D_i) \\ &= \hat{e}(P^{'}, \Sigma_{i=1}^{k} r_i P) = \hat{e}(\Sigma_{i=1}^{k} r_i P^{'}, P) \\ &= \hat{e}(S,P)\end{aligned}$$

Wen et al. in their paper[11] have proved that their signature is unforgeable in the choosen key model as given in [1]. Their signature consists of $(T = \hat{h}P, S = (\hat{h} + hs)P^{'})$ where $\hat{h} = H(m \| s), h = H(m)$, no user can verify whether the sender has send $T = \hat{h}P$ or $T = rP$ where r is a random element belonging to $\mathbb{Z}_q^*$ because no user other than the sender himself can calculate $\hat{h}$. For every other user it just seems like a random value. Therefore their signature is forgeable by anyone.

9 Identity Based universal designated multi-verifiers signature schemes

Seung et al. [8] have proposed the first identity based universal designated multi-verifiers signature scheme which generalizes identity based universal designated verifier signature scheme. Inorder to achieve this they have proposed a new identity based signature scheme which provides batch verification of signatures. They have proved their signature scheme existentially unforgeable in the random oracle model. In this paper we show that their scheme is not secure when considered for batch verification. Following are the review and attack of [8].

9.1 Review of Seung et al.'s Identity Based Signature Scheme and its Batch Verification

In this section we present the Identity-based signature scheme and its Batch Verification as proposed by Seung et al. [8] in their paper *Identity-based universal designated multi-verifiers signature schemes.* This scheme consists of four algorithms, Setup, Extract, IDSign, ID-PV(Identity-Public Verify).

ID-Based Signature Scheme:

– **Setup:** The PKG chooses a random generator P of $\mathbb{G}$. PKG chooses $s \in_R \mathbb{Z}_p^*$ and computes $P_{pub} = sP$. Then PKG keeps s as the *master secret key* and publishes system parameters $param = \{\hat{e}, \mathbb{G}, \mathbb{G}_T, q, P, P_{pub}, H_1, H_2\}$ where, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ are cryptographic hash functions.

– **Extract:** Given an user identity ID, the PKG compute's its public key $D_{ID} = H_1(ID)$ and the private key $S_{ID} = sD_{ID}$, and returns (D_{ID}, S_{ID}) to the user in a secure way.

– **ID-Sign:** Given a secret key S_{ID} and a message m, pick $r \in_R \mathbb{Z}_q^*$, compute $T = rP$, $h = H_2(m, T)$ and $S = rP_{pub} + hS_{ID}$. Output the signature $\sigma = (T, S)$.

– **ID-PV:** Given the public parameter *param*, a message m, a signature $\sigma = (T, S)$, check if

$$\hat{e}(S, P) \stackrel{?}{=} \hat{e}(T + hD_{ID}, P_{pub})$$

If the equality holds then output accept. Otherwise reject.

Batch Verification:

The above mentioned ID-Based Signature scheme allows batch verification of multiple signatures on different messages. That is, a verifier can check the validity of n signatures $(T_1, S_1), \ldots, (T_n, S_n)$ on n messages $m_1, ..., m_n$ simultaneously by checking the following:

$$\hat{e}(\textstyle\sum_{i=1}^{n} S_i, P) \stackrel{?}{=} \hat{e}(\textstyle\sum_{i=1}^{n}(T_i + h_i D_i), P_{pub})$$

10 Attack on the Seung et al.'s Scheme

We will show that universal forgery is possible in the batch verification of Seung et al. scheme [8]. Consider the adversary $\mathbb{A}$ queries for the private key of some identity $ID_{\mathbb{A}}$. Let the identity on whom the forgery is to be performed is ID. Assume the $\mathbb{A}$ in the training phase queries for a signature of ID on message m. The signature is of the form

$$T = xP \tag{3}$$

$$S = xP_{pub} + hs_{ID} \tag{4}$$

for a random unknown x.
The value of h can be calculated as $H_2(m, T)$
. The adversary does the following steps to sign a message m_1 on behalf of ID and aggregate with his own signature on some random message.

– Choose a random x_1 and set $T_1 = x_1 P_{pub}$.
– Set $h_1 = H_2(m_1, T_1)$.
– Set $T^* = \frac{x}{h} h_1 P$. (Dividing 3 by h and multiplying by h_1)

- Set $S^* = \frac{x}{h}h_1P_{pub} + h_1s_{ID}$. (Dividing 4 by h and multiplying by h_1.)

- Set $S_1 = S^* + x_1P_{pub}$. $=\frac{x}{h}h_1P_{pub} + h_1s_{ID} + x_1P_{pub}$. The signature on m_1 by ID is $\sigma_1 = (S_1, T_1)$.
- Then the adversary $\mathbb{A}$ uses his private key $s_{\mathbb{A}}$ and signs a random message m_2 as follows.
- $T_2 = x_2P + T^*$
  $=x_2P + \frac{x}{h}h_1P$.

- $h_2 = H_2(m_2, T_2)$.
- Set $S_2 = x_2P_{pub} + h_2s_{\mathbb{A}}$
- Then aggregate the S_i components of both the signatures and the aggregate signature is of form $\sigma = (S, T_1, T_2)$ on messages m_1 and m_2. We can verify that this is a valid aggregate signature by the verification algorithm.

$$\begin{aligned}
\hat{e}(S,P) &= \hat{e}(\frac{x}{h}h_1P_{pub} + h_1s_{ID} + x_1P_{pub} + x_2P_{pub} + h_2s_{\mathbb{A}}, P) \\
&= \hat{e}(x_1P_{pub}, P)\hat{e}((\frac{x}{h}h_1 + x_2)P_{pub}, P)\hat{e}(h_1s_{ID} + h_2s_{\mathbb{A}}, P). \\
&= \hat{e}(T_1, P_{pub})\hat{e}(T_2, P_{pub})\hat{e}(\Sigma_{i=1}^{2}h_iD_{ID_i}, P_{pub}). \\
&= \hat{e}(\Sigma_{i=1}^{2}T_i, P_{pub})\hat{e}(\Sigma_{i=1}^{2}h_iD_{ID_i}, P_{pub}). \\
&= \hat{e}(\Sigma_{i=1}^{2}(T_i + h_iD_{ID_i}), P_{pub}).
\end{aligned}$$

Thus this aggregate signature passes the verification. The adversary can forge signature on any message by any user having just known a single signature of the user. This kind of break is possible only in aggregation. The base signatur scheme is secure but its not secure when aggregated. Thus we have proved a universal forgery on Seung et al. signature scheme[8] which is claimed to support batch verification.

11 Review of Shi Cui et al. Scheme

In this section we review the scheme proposed by Shi Cui et al. in [4] where it is claimed that their identity based signature construct can be used for efficient batch verification. They have used the scheme to do batch verification of 3 types. Of these three type the type-3 batch verification construct is similar to aggregate signature construct where n signers sign in n distinct messages and the signatures on all the messages can be verified in a single verification. The base signature scheme consists of 4 algorithms which are as follows.

- **Setup :** The trusted authority runs this algorithm with the security parameter $l \in \mathbb{Z}_{\shortparallel}^*$ as the input. The algorithm chooses two groups $\mathbb{G}_1, \mathbb{G}_2$ and chooses randomly a P as generator of $\mathbb{G}_1$ and a $s \in Z_q^*$ and computes $P_{pub} = sP$ and also computes $\omega = \hat{e}(P, P)$. It chooses a hash function $H : \{0,1\}^* \times \mathbb{G}_2^* \longleftrightarrow \mathbb{Z}_l^*$ and returns the public parameters $params$ as $< P, P_{pub}, \omega, \mathbb{G}_1, \mathbb{G}_2, H >$ and the master secret key s.

– **Extract :** For a given identity $id_i \in Z_l^*$ the trusted authority runs this algorithm with *params*,msk s, and identity id_i as the input. The algorithm does the folowing.

$$S_{ID_i} = \frac{1}{s + id_i}$$

and returns the private key S_{ID_i} to the user id_i.

– **Sign :** A user with id_i who wishes to sign message m_i runs this algorithm with his private key S_{ID_i}, *params* and message m_i as input. The signing algorithm does the following.

- Choose a random $x_i \in Z_l^*$
- Compute $T_i = \omega_i^x$
- Compute the hash $h_i = H(m_i, T_i)$
- Then compute $S_i = (x_i + h_i)S_{id_i}$

and return the signature $< S_i, T_i >$ as the signature on message m_i by identity id_i.

– **Verify :** Any user can run this algorithm to verify the validity of the signature. This algorithm takes as input the signature $< S_i, T_i >$, the message m_i , the identity id_i and the *params* as input. The algorithm verifies whether

$$\omega^{h_i}.T_i \stackrel{?}{=} \hat{e}(P_{pub} + id_i.P, S_i) \tag{5}$$

If yes it outputs "*Valid*" else it outputs "*Invalid*".

– **Correctness :**

$$\begin{aligned} \hat{e}(P_{pub}, id_i.P, S_i) &= \hat{e}((s + id_i)P, S_{id_i})^{h_i + x_i} \\ &= \hat{e}((s + id_i)P, \tfrac{1}{(s+id_i)}P)^{h_i + x_i} \\ &= \hat{e}(P, P)^{h_i + x_i} \\ &= \omega^{h_i + x_i} \\ &= \omega^{h_i}.T_i \end{aligned}$$

– **Batch verification for type 3 :** Suppose there are n signatures from n users $\{id_i\}_{i=1}^n$ on n messages $\{m_i\}_{i=1}^n$ where the signatures is of the form $< S_1, T_1, id_1 >< S_2, T_2, id_2 > \ldots < S_n, T_n, id_n >$ and user can run this algorithm with these inputs. The algorithm checks whether

$$\omega^{\sum_{i=1}^n h_i}.\prod_{i=1}^n T_i \stackrel{?}{=} \hat{e}(P_{pub}, \sum_{i=1}^n S_i)\hat{e}(P, \sum_{i=1}^n id_i S_i) \tag{6}$$

If yes it outputs "*Valid*" else it outputs "*Invalid*".

12 Universal forgery of Shi Cui et al. Scheme with batch verification-type 3

In this section we propose an attack on type-3 batch verification of Shi Cui et al. scheme [4]. Consider a adversary $\mathbb{A}$ who is going to produce a forgery on the target identity ID. The $\mathbb{A}$ queries for the private key of some identity $ID_{\mathbb{A}}$ other than the target identity ID and the private key is $S_{id_{\mathbb{A}}}$. The $\mathbb{A}$ then queries the signature of the target identity on some message m_1. The signature is of the form

$$T_1 = \omega^{x_1} S_1 = (h_1 + x_1)S_{id_1} \tag{7}$$

for some random x_1 and where $h_1 = H(m_1, T_1)$. The adversary then does the following to create a forgery on message m_2.

Chooses a random x_2 and set $T_2 = \omega^{x_2}$ and $h_2 = H(m_2, T_2)$.
Then sets $S_2 = (h_2 + \frac{x_1.h_2}{h_1})S_{id_1}$.(By dividing 7 by h_1 and multiplying by h_2).
Then chooses another random x_3,message m_3 and set $T_3 = \omega^{x_3}\omega^{-x_2}.\omega^{\frac{x_1.h_2}{h_1}}$ and $h_3 = H(m_3, T_3)$.
Sets $S_3 = (h_3 + x_3)S_{id_{\mathbb{A}}}$.

Therefore $< S_2, T_2 >$ is a signature on m_2 by target identity id and $< S_3, T_3 >$ is a signature on m_3 by identity $id_{\mathbb{A}}$.

It passes the batch verification as follows.

$$\omega^{\sum_{i=2}^{3} h_i}.\prod_{i=2}^{3} T_i \stackrel{?}{=} \hat{e}(P_{pub}, \sum_{i=2}^{3} S_i)\hat{e}(P, idS_2 + id_{\mathbb{A}}S_3) \tag{8}$$

Correctness of Forgery :
LHS is of the form

$$\omega^{h_2+h_3}.T_2.T_3 = \omega^{h_2+h_3+x_2+x_3-x_2+\frac{x_1.h_2}{h_1}} = \omega^{h_2+h_3+x_3+\frac{x_1.h_2}{h_1}} \tag{9}$$

The RHS is of the form

$$\hat{e}(P_{pub}, \sum_{i=2}^{3} S_i)\hat{e}(P, idS_2 + id_{\mathbb{A}}S_3) = \hat{e}(P_{pub}, S_2)\hat{e}(P_{pub}, S_3)\hat{e}(P, idS_2)\hat{e}(P, id_{\mathbb{A}}S_3) \tag{10}$$

$$= \hat{e}(sP + idP, S_2)\hat{e}(sP + id_{\mathbb{A}}P, S_3) \tag{11}$$

$$= \hat{e}((s+id)P, (h_2 + \frac{x_1.h_2}{h_1})\frac{1}{s+id}P)\hat{e}((s+id_{\mathbb{A}})P, (h_3 + x_3)\frac{1}{s+id_{\mathbb{A}}}P) \tag{12}$$

$$= \hat{e}(P, P)^{h_2+\frac{x_1.h_2}{h_1}}.\hat{e}(P, P)^{h_3+x_3} \tag{13}$$

$$= \omega^{h_2+\frac{x_1.h_2}{h_1}+h_3+x_3} \tag{14}$$

(15)

The equation 9 is equal to equation 14. Thus it is valid forgery where the adversary by knowing just a single users private key can forge the signature of any user in the system on any message by just seeing a signature of that user on someother message. Thus we have proved that the scheme in [4] is universally forgeable in batch verification of type-3.

13 Conclusion

We have analysed two aggregate signature schemes which claimed to have efficient computation complexity. We have shown that universal forgery is possible in such constructs and similar constructs. Currently there is no scheme which achieves constant computation cost in pairings during verfication without any interaction among the signers. The interaction among singers in certain situations is a major disadvantage since each singer has to broadcast his value to all other signers and that makes n broadcast operations thus increasing the communication complexity highly. An efficient aggregate signature with constant pairing computations in verification without any interaction among users remains as an interesting open problem in this field. Also developing an aggregate signature in the standard model is another open problem to look at.

References

1. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
2. Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2003.
3. Xiangguo Cheng, Jingmei Liu, and Xinmei Wang. Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *ICCSA (4)*, volume 3483 of *Lecture Notes in Computer Science*, pages 1046–1054. Springer, 2005.
4. Shi Cui, Pu Duan, and Choong Wah Chan. An efficient identity-based signature scheme with batch verifications. In Xiaohua Jia, editor, *Infoscale*, volume 152 of *ACM International Conference Proceeding Series*, page 22. ACM, 2006.
5. Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 257–273. Springer, 2006.
6. Javier Herranz. Deterministic identity-based signatures for partial aggregation. *Comput. J.*, 49(3):322–330, 2006.
7. Florian Hess. Efficient identity based signature schemes based on pairings. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2002.
8. 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):288–295, 2008.
9. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
10. Zhu Wang and Huiyan Chen. A practical identity-based signature scheme from bilinear map. In Mieso K. Denko, Chi-Sheng Shih, Kuan-Ching Li, Shiao-Li Tsao, Qing-An Zeng, Soo-Hyun Park, Young-Bae Ko, Shih-Hao Hung, and Jong Hyuk Park, editors, *EUC Workshops*, volume 4809 of *Lecture Notes in Computer Science*, pages 704–715. Springer, 2007.
11. Yiling Wen and Jianfeng Ma. An aggregate signature scheme with constant pairing operations. In *CSSE (3)*, pages 830–833. IEEE Computer Society, 2008.
12. Jing Xu, Zhenfeng Zhang, and Dengguo Feng. Id-based aggregate signatures from bilinear pairings. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *CANS*, volume 3810 of *Lecture Notes in Computer Science*, pages 110–119. Springer, 2005.
13. HyoJin Yoon, Jung Hee Cheon, and Yongdae Kim. Batch verifications with id-based signatures. In Choonsik Park and Seongtaek Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 233–248. Springer, 2004.
14. Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 277–290. Springer, 2004.