Identity Based Partial Aggregate Signature Scheme Without Pairing

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

[1]Department of Computer Science and Engineering,
Indian Institute of Technology Madras.
sharmila@cse.iitm.ac.in,svivek@cse.iitm.ac.in, prangan@iitm.ac.in.
[2]National Institute of Technology Trichy, India

Abstract. An identity based signature allows users to sign their documents using their private keys and the signature can be verified by any one, using the identity of the signer and public parameters of the system. An aggregate signature scheme is a digital signature scheme which allows aggregation of different signatures by different users on different messages. The primary objective of aggregate signature scheme is to achieve both computational and communication efficiency. Here, we propose an identity based aggregate signature scheme, which uses a variation of light weight Schnorr type identity based signature scheme, where in the signers need not agree upon a common randomness and the aggregation is done without having any kind of interaction among the signers. The scheme does not involve any pairing operations even for aggregate signature verification. It is computationally efficient since it avoids the costlier operation in elliptic curve groups (Bilinear Pairings). It should be noted that our signature achieves only partial aggregation because the private key of each user is generated by a randomized extract algorithm and hence a random value is to be propagated with each single signature generated.

Keywords: Identity Based Signature, Aggregate Signature, Partial Aggregation, Random Oracle Model, Provable Security.

1 Introduction

Many real-life applications, handle a collection of signed documents together rather than handling them individually. It is not hard to visualize such a scenario in Bank transactions, legal document processing (archiving and communicating) in a legal firm, digital attestation related application and so on. In all the above applications, generating, storing and transmitting a large number of signed documents arise naturally. An *Aggregate Signature Scheme* combines several signed documents, say $\sigma_1, \ldots, \sigma_t$ on messages $m_1, \ldots, m_t$ by users $U_1, \ldots, U_t$ and produces a single signed document σ_{agg} where size of σ_{agg} is substantially smaller than sum of the sizes of σ_i's. This leads to a significant reduction in the communication cost because, it is only required to transmit σ_{agg} instead of transmitting $\sigma_1, \ldots, \sigma_t$ individually. A similar remark holds good even for storage requirements when σ_{agg} is archived instead of $\sigma_1, \ldots, \sigma_t$.

Certificate chains in hierarchical PKI (Public Key Infrastructure) systems consists of various signatures at different levels in the hierarchy. By using aggregate signature one can combine all these signatures and thus reduce the certificate length. Ordered sequential aggregation is used in communication between routers in a network where each router receives

⋆ 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

the data and the signature of the previous router. The current router aggregates its own signature to the previous aggregate signature and routes it to the next router. This aggregated signature can be used to find the path travelled by the data from source to destination by verifying a single aggregate signature. Aggregate signatures can also be used in wireless network scenarios. Since the major constraint in wireless networks is communication complexity, the use of efficient aggregate signature helps in reducing the amount of data to be communicated.

Identity based aggregate signatures are considered to be more effective than PKI based aggregation because in the formar, only identities (which are very short when compared to group elements) of the signers have to be send along with the aggregate signatures, in order to facilitate verification. In PKI based system, efficient aggregation can be done but the identities, public keys and the certificates (issued by the Certifying Authority) of the public keys of the signers has to be send along with the aggregate signature. The signatures and certificates can be easily aggregated in PKI based system but then there is no known technique to aggregate the public keys. Many well known PKI based aggregate signatures are available in the literature [16, 2, 7, 15]. However, as the focus of this paper is on identity based aggregate signature scheme, we will not compare the PKI based schemes with ours. To the best of our knowledge, there are four provably secure identity based aggregate signature schemes [11], [21], [12] and [8]. It should be noted that all of them use bilinear pairing during the verification process. An identity based aggregate signature scheme is claimed to achieve partial aggregation if a part of the signature is aggregated, namely the part with the secret key component of signers is fully aggregated and the randomness part is propagated without aggregation. If both the parts in the signature are fully aggregated then the scheme is said to achieve full aggregation. We provide a brief survey about the efficiency and weakness of various identity based aggregate signature schemes.

Survey of Existing Schemes: Currently, in literature we have number of identity based aggregate signature schemes.

Xu et al. in [21] proposed an identity based aggregate signature scheme. This scheme uses Sakai et al.'s signature construct as the base signature scheme. This achieves partial aggregation but requires linear number of pairings during signature verification.

Xiangguo et al. gave a aggregate signature scheme [8] which uses the BLSR scheme [6] as the base signature scheme. In this scheme all the signers have to broadcast their own random values used for singing to all the cosigners so that everyone agrees upon a common randomness before the generation of aggregate signature. This result in quadratic communication complexity which is a big overhead. Mutual interaction between all the signers is not always a desirable event in aggregate signatures.

Javier Herranz came up with an identity based signature scheme [12] with partial aggregation. His scheme produces deterministic signature where the signature on a message will always be the same. It also uses linear number of pairing operations leading to inefficient verification.

Yiling et al. proposed an efficient aggregate signature scheme with full aggregation and constant pairing operations in [20]. But the scheme in [20] is not secure since universal forgery of the base signature scheme used in [20] is possible as shown in [18].

Wang et al. designed an identity based aggregate signature [19] and it is claimed to be the most efficient scheme. It uses constant pairing operation for signature verification. But the aggregate signature in this scheme [19] is not secure since universal forgery of signature of any user is possible in this scheme. The attack in Wang et al. scheme [19] is shown in [18].

Recently, Boldyreva et al. [4] proposed a sequential aggregate signature scheme, in-fact the security of their original schemes [3] and [5] were flawed due to the assumption used to

prove them was actually not hard to solve in polynomial time, which was pointed out by Hwang et al. [13]. Their new scheme [4] was based on the hardness of a CDH-type problem that raised from their scheme and uses bilinear pairings. One more aggregate signature scheme, which achieves full aggregation is by Gentry and Ramzan [11]. The drawback of the scheme is, it requires interaction between all the signers whose signatures are to be aggregated before the signature generation process. The weakness of the scheme in [11] is briefly reported in the appendix of our paper. The first RSA based identity based aggregate signature scheme was proposed by Bagherzandi et al. [1]. This scheme uses two rounds of communication between the signers to generate a full aggregate signature, where the first round is to commit the random value shares (again by broadcasting the individual commitments as in [8]) and the second round is the aggregate signature generation round. Their scheme uses equivocable commitments and hence looses its generality and becomes less practical because of the overhead involved in broadcasting the commitments.

Our contribution: In this paper, we propose an identity based aggregate signature scheme whose key generation is similar to Javier Herranz's identity based partial aggregate signature scheme [12]. Our scheme does not require bilinear pairing operation during aggregate signature verification. This is achievable because the individual signature is a Schnorr-like signature. As in [12], our schemes do not have interaction among the signers before signature generation which reduces the communication complexity to a large extent. However, in this scheme we are able to achieve only partial aggregation. We are able to achieve only partial aggregation (as defined by Javier Herranz [12]) because the extract algorithm is a randomized algorithm and the signature generation algorithm is also randomized. We formally prove the security of our scheme in the random oracle model. We also point out certain weaknesses in [11], which makes it unsuitable in real life. The weaknesses of the scheme are briefly reported in the appendix.

2 Preliminary

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. For any P, Q, $R \epsilon \mathbb{G}$ and a, $b \epsilon \mathbb{Z}_q^*$

- **Bilinearity:** $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
- **Non Degenerate:** $\hat{e}(P, P) \neq 1$.
- **Easily Computable:** $\hat{e}(P, Q)$ must be easily and efficiently computable.

2.2 Computational Assumptions

In this section, we review the computational assumption that is relevant to the protocol we discuss.

Discrete Logarithm Problem (DLP): Let $(\mathbb{G}, *)$ be a multiplicative group of order p, $g \in \mathbb{G}$ be a generator of $\mathbb{G}$ and $h = g^x \in \mathbb{G}$, where $x \in \mathbb{Z}_p$ be unknown. Given g and h, the discrete logarithm problem is to find x.

An algorithm $\mathcal{A}$ has an advantage ϵ in solving DLP_{G_1} if

$$Pr[\mathcal{A}(g, h) = x] \geq \epsilon.$$

3 Generic Model

An identity based aggregate signature scheme (IBAS) consists of following six algorithms.

- **Setup:** The private key generator (PKG) provides the security parameter κ as the input to this algorithm, generates the system parameters $params$ and the master private key Msk. PKG publishes $params$ and keeps 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 obtains the private key D_i. The private key D_i is sent to user U_i through a secure channel.
- **Sign:** For generating a signature on a message m_i, the user U_i provides his identity ID_i, his private key D_i, $params$ and message m_i as input. 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 U 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 $(\sigma_i)_{i=1\,to\,n}$ from different users $(U_i)_{i=1\,to\,n}$, any third party or one of the signers can run this algorithm and generate the aggregate signature σ_{agg} for the set of $\langle message, identity\rangle$ pairs $(m_i, ID_i)_{i=1\,to\,n}$.
- **AggregateVerify:** This algorithm on input of an aggregate signature σ_{agg}, the list for $(m_i, ID_i)_{i=1\,to\,n}$ and the $params$ checks whether σ_{agg} is a valid aggregate signature on m_i by ID_i for all $i = 1$ to n. If true, it outputs "$Valid$", else outputs "$Invalid$".
- **Token Generation:** In some models of identity based systems, the PKG may generate a random value that corresponds to the registered user at the time of registration / key generation. This value will be made public by the user. We refer this value as *'token'* and tokens are not used for any encryption schemes. These tokens will always be send as a part of the signature. This extended version of identity based signatures is not considered as violation because this token is used only for signing purpose. More on this have been discussed in section 5.

Important Remark: As encryption algorithms only use the publicly known identities of the user alone as public key, tokens are never used in encryption schemes. Since ours is an identity based signature scheme, introduction of token in our cryptosystem is not a violation of the definition of identity based system.

4 Security Model

4.1 Unforgeability

Gentry et al. in [11] proposed a formal model for aggregate signature scheme. Their scheme used a common randomness. We follow the security model proposed by Gentry et al. with slight variations since we do not have a common random value. An IBAS scheme is secure against existential forgery under adaptive-chosen-identity and adaptive-chosen-message attack if no probabilistic polynomial time algorithm $\mathcal{A}$ has non-negligible advantage in the following game.

- **Setup phase:** The challenger $\mathcal{C}$ runs the setup algorithm and generates the $params$ and Msk. Challenger $\mathcal{C}$ gives $params$ to adversary $\mathcal{A}$.
- **Training phase:** After the setup, $\mathcal{A}$ starts interacting with $\mathcal{C}$ by querying the various oracles provided by $\mathcal{C}$ in the following way:
  - **KeyGen oracle:** When $\mathcal{A}$ makes a query with ID_i, $\mathcal{C}$ outputs D_i, the private key of ID_i to $\mathcal{A}$, provided $\mathcal{C}$ knows the private key for the queried identity. Else it aborts.

- **Signing oracle:** When $\mathcal{A}$ makes a signing query with ID_i, message m_i, $\mathcal{C}$ outputs a valid signature σ_i on m_i by ID_i.

– **Forgery phase:** $\mathcal{A}$ outputs an aggregate signature σ_{Agg} for signatures $(\sigma)_{i=1\,to\,n}$ from the users $(ID_i)_{i=1\,to\,n}$ on messages $(m_i)_{i=1\,to\,n}$ where there exists at least one target identity $ID_T \in \{ID_i\}_{i=1\,to\,n}$, for which private key has not been queried for. The adversary $\mathcal{A}$ wins the game if σ_{agg} is a valid aggregate signature and $\mathcal{A}$ has not queried for the signature from the signing oracle for (ID_T, m_T) pair on which it has generated the forgery.

$$Adv_{\mathcal{A}}^{IBAS} = \{Pr[\mathcal{A}(Verify(\sigma_{agg}) = valid)\}$$

5 Identity Based Aggregate Signature scheme Without Pairings (IBAS)

Normally, the public key of a user in identity based cryptography is obtained by hashing the user's identity, which uniquely identifies him. In the identity based signature by Galindo et al. [10], we find an interesting and subtle difference between all existing schemes and [10]. In [10], Galindo et al. have used a Schnorr signature which in turn uses a purely random value chosen by the PKG to generate the private key of the user. This random value can be interpreted as a *'token'* which we discussed in section 3 on generic model of identity based aggregate signature scheme. This token along with the identity of the user is hashed together to obtain the public key corresponding the user. It should be noted that this is not a violation of the property of identity based cryptosystem with respect to digital signature schemes because, in a digital signature scheme all the components of a signature on an arbitrary message are generated by the signer who is in possession of the private key. Hence, the signer has to send the random value obtained with his private key from the PKG along with each signature he generates. The interesting part is that, if the signer or any potential forger tries to alter the random value obtained from the PKG for the signer, both will fail miserably in generating a valid signature because neither signer nor the forger will be able to generate a valid private key corresponding to the altered random value. We emphasize again that tokens can never be used for encryption schemes and can always be used in signature schemes. In Galindo et al.'s [10] paper, the component g^r is send by the PKG to the user. This component is called as *'token'* in our convention.

Important Remark: Similar kind of key constructs for identity based cryptosystem can be seen in [9], [12] and [14]. In [9], an identity based key agreement protocol was proposed by Dario et al., in [12] a deterministic aggregate signature scheme was proposed by Javier Herranz and in [14] an identity based online/offline signature was proposed by Liu et al.. In this section, we describe a new identity based aggregate signature scheme based on the identity based signature scheme by Galindo et al. [10]. This scheme consists of six algorithms which are described below.

– **IBAS.Setup:** Let κ be the security parameter of the system. Let $\mathbb{G}$ be a multiplicative group of order q. Choose a random generator g of $\mathbb{G}$. Choose three cryptographic hash functions which are defined as $H_1 : \{0,1\}^* \times \mathbb{G} \longrightarrow \mathbb{Z}_q^*, H_2 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{Z}_q^*$ and $H_3 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{Z}_q^* \times \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{Z}_q^*$. Let $s \in_R \mathbb{Z}_q^*$ be the master private key and the master public key is set to be g^s. The public parameters are $params = \langle g, g^s, \mathbb{G}, H_1, H_2, H_3 \rangle$ and the master private key s is kept secret.

– **IBAS.KeyGen:** The user U_i provides his identity ID_i to the Private Key Generator (PKG). The PKG runs this algorithm with ID_i, $params$ and master private key s as the input. The algorithm does the following:
  - Choose a random $x_i \in \mathbb{Z}_q$

- Computes $X_i = g^{x_i}$ and $q_i = H_1(ID_i, X_i)\ mod\ q$
- Computes $d_i = (x_i + sq_i)\ mod\ q$
- Outputs $\langle q_i, X_i, d_i \rangle$

The PKG sends $\langle q_i, X_i, d_i \rangle$ securely to the user U_i. The user U_i keeps the d_i as secret and $\langle q_i, X_i \rangle$ as public. Here X_i is called the token.

Remark: It is to be noted that the private key d_i is a Schnorr signature on the identity ID_i and thus a user who is capable of producing another private key d_i' for the same identity ID_i or a private key d_i'' for an arbitrary identity ID_i'' can effectively forge the underlying Schnorr signature. As a consequence, the private key generated by the PKG is secure and cannot be generated by any user by altering the token value, unless he knows the master private key s. Therefore, we do not consider token as a separate entity for the formal proof of unforgeability of our scheme.

– **IBAS.Sign:** The user U_i who wishes to sign a message m_i gives his ID_i, private key d_i and *params* as input to this algorithm. The algorithm does the following to generate the signature:
- Chooses a random $r_i \in \mathbb{Z}_q$.
- Computes $W_i = g^{r_i}$
- Generates $h_{1i} = H_2(m_i, ID_i, W_i, X_i)$
- Generates $h_{2i} = H_3(m_i, ID_i, h_{1i}, W_i, X_i)$
- Computes $v_i = (r_i h_{1i} + h_{2i} d_i)\ mod\ q$
- Outputs $\langle X_i, v_i, h_{1i}, h_{2i} \rangle$ as the signature of ID_i on message m_i.

Remark: Note that the token value X_i is also a component of the signature. Thus, the verifier need to know only the identity of the signer to perform verification.

– **IBAS.Verify:** Any user can run this verification algorithm. The user provides $\langle X_i, v_i, h_{1i}, h_{2i} \rangle$, ID_i, m_i and *params* as input to this algorithm. The verification is done as follows:
- Compute $q_i = H_1(ID_i, X_i)$
- Compute $\overline{W_i} = \left(\frac{g^{v_i}}{(X_i (g^s)^{q_i})^{h_{2i}}} \right)^{h_{1i}^{-1}}$
- Check whether:
  $h_{1i} \stackrel{?}{=} H_2(m_i, ID_i, \overline{W}_i, X_i)$ and
  $h_{2i} \stackrel{?}{=} H_3(m_i, ID_i, h_{1i}, \overline{W}_i, X_i)$
- Output "$Valid$" if both the verifications pass, else output "$Invalid$".

Correctness of the computation of $\overline{W}_i$:

$$RHS = \left(\frac{g^{v_i}}{(X_i (g^s)^{q_i})^{h_{2i}}} \right)^{h_{1i}^{-1}} = \left(\frac{g^{(r_i h_{1i} + h_{2i} d_i)}}{(X_i (g^s)^{q_i})^{h_{2i}}} \right)^{h_{1i}^{-1}} = \left(\frac{g^{(r_i h_{1i} + h_{2i}(x_i + sq_i))}}{(X_i (g^s)^{q_i})^{h_{2i}}} \right)^{h_{1i}^{-1}}$$
$$= \left(\frac{g^{r_i h_{1i}} g^{(x_i + sq_i) h_{2i}}}{(X_i (g^s)^{q_i})^{h_{2i}}} \right)^{h_{1i}^{-1}} = \left(\frac{g^{r_i h_{1i}} (X_i (g^s)^{q_i})^{h_{2i}}}{(X_i (g^s)^{q_i})^{h_{2i}}} \right)^{h_{1i}^{-1}}$$
$$= \left(g^{r_i h_{1i}} \right)^{h_{1i}^{-1}} = g^{r_i} = \overline{W}_i = LHS = W_i \text{ (as required)}$$

This shows that the above verification check is valid and consistent. Note that the verification can be done by anyone as it involves only publicly known parameters such as $v_i, W_i, h_{1i}, h_{2i}, g^s, X_i, q_i$

– **IBAS.Aggregate:** This algorithm takes as input a set of n signatures $\{X_i, v_i, h_{1i}, h_{2i}\}_{i=1\ to\ n}$ and the corresponding identity, message pairs $\langle ID_i, m_i \rangle$, such that $\forall i = 1$ to n $\langle X_i, v_i, h_{1i}, h_{2i} \rangle$ is the signature on message m_i by ID_i. The aggregation is done as follows:

$$v_{agg} = \sum_{i=1}^{n} v_i.$$

The algorithm outputs the final aggregate signature $\langle X_i, v_{agg}, h_{1i}, h_{2i}\rangle_{i=1\,to\,n}$ and the corresponding message identity pair $\{m_i, ID_i\}_{i=1\,to\,n}$.

– **IBAS.AggregateVerify:** This algorithm takes the aggregate signature $\langle X_i, v_{agg}, h_{1i}, h_{2i}\rangle_{i=1\,to\,n}$ and the corresponding message identity pair $\{m_i, ID_i\}_{i=1\,to\,n}$ and performs the following to verify the aggregate signature:
  - For $i = 1$ to n, compute $q_i = H_1(ID_i, X_i)$
  - For $i = 1$ to n, compute $\overline{W_i} = \left(\frac{g^{v_i}}{(X_i(g^s)^{q_i})^{h_{2i}}}\right)^{h_{1i}^{-1}}$
  - For $i = 1$ to n, check whether:
    Compute $h_{1i} \stackrel{?}{=} H_2(m_i, ID_i, W_i, X_i)$ and
    Compute $h_{2i} \stackrel{?}{=} H_3(m_i, ID_i, h_{1i}, W_i, X_i)$
  - Check whether $g^{v_{agg}} \stackrel{?}{=} \prod_{i=1}^{n}(\overline{W}_i)^{h_{1i}}.\prod_{i=1}^{n}(X_i)^{h_{2i}}.(g^s)^{\sum_{i=1}^{n} q_i h_{2i}}$. If all the above checks hold good, output "$Valid$" else output "$Invalid$".

Correctness of the IBAS.AggregateVerify algorithm:

$$LHS = g^{v_{agg}} = g^{\sum_{i=1}^{n} r_i h_{1i} + \sum_{i=1}^{n} h_{2i} d_i} = g^{\sum_{i=1}^{n} r_i h_{1i}}.g^{\sum_{i=1}^{n} h_{2i} d_i}$$
$$= \prod_{i=1}^{n}(g^{r_i})^{h_{1i}}.g^{\sum_{i=1}^{n} h_{2i}(x_i + s q_i)} = \prod_{i=1}^{n}(\overline{W}_i)^{h_{1i}}.\prod_{i=1}^{n}(X_i)^{h_{2i}}.(g^s)^{\sum_{i=1}^{n} q_i h_{2i}} = RHS$$

This shows that the aggregate verification test is correct and consistent.

6 Security Proof for IBAS

In this section, we prove the security of our identity based aggregate signature scheme (IBAS). We show that if a polynomial time bounded adversary exists who can break our scheme with non-negligible probability ϵ' then we will be able to solve the discrete logarithm problem with non-negligible probability ϵ_0. We prove that our scheme is secure against existential forgery under adaptive chosen message and adaptive chosen identity attack. We also use the oracle replay attack technique and forking lemma [17] to prove the security of our scheme.

Theorem 1. *Our aggregate signature scheme IBAS is secure against existential forgery under adaptively chosen identity and adaptively chosen message attack, if there exists a polynomially bounded (t, ϵ') adversary $\mathcal{A}$ making q_{H_1}, q_{H_2}, q_{H_3} hash queries, q_S signcryption queries and q_E extraction queries, who can break our scheme with a non-negligible advantage ϵ', then there exists a DL solver $\mathcal{C}$ with a non-negligible advantage,*

$$\epsilon_0 = \frac{1}{9}.\frac{10(q_S+1)(q_S+q_{H_3}+q_{H_2}).(1-\frac{q_E}{q_{H_1}}).n}{2^{k+1}}.\frac{1}{q_{H_1}}\epsilon$$

and in polynomial time t_0.

Proof. The adversary $\mathcal{A}$ in adaptive chosen message and adaptive chosen identity attack has access to all the hash oracles, the signing oracle and the key extract oracle. The Challenger $\mathcal{C}$ uses the forgery by $\mathcal{A}$ to solve the computational hard problem (DLP) with the help of oracle replay technique. $\mathcal{C}$ is given an instance of the DL problem i.e given g,$h = g^x \in G$ for some unknown x, the discrete logarithm problem is to find x. The game between $\mathcal{C}$ and $\mathcal{A}$ is as follows:

– **Setup Phase:** $\mathcal{C}$ chooses a group $\mathbb{G}$ and a generator g for the group $\mathbb{G}$. Then, $\mathcal{C}$ chooses a random $s \in Z_q^*$ and calculates g^s. $\mathcal{C}$ has all the three hash oracles H_1, H_2, H_3 under his control. So he gives the public parameters $params = \langle g, g^s, H_1, H_2, H_3 \rangle$ to the adversary and keeps the master secret s to himself.

– **Training Phase:** $\mathcal{A}$ can ask for different queries to $\mathcal{C}$. $\mathcal{C}$ responds correspondingly to the various queries by $\mathcal{A}$. Let $q_E, q_{H_1}, q_{H_2}, q_{H_3}, q_S$ be the maximum number of queries the adversary $\mathcal{A}$ can query to key extract oracle, three hash oracles and the signing oracle respectively. The various queries are as follows:

- $\mathcal{O}_{H_1}$**:** There are two cases in this query.
  * $\mathcal{A}$ queries the H_1 oracle with ID_i, X_i as input, $\mathcal{C}$ chooses a random q_i and returns it to $\mathcal{A}$.
  * Sometimes $\mathcal{A}$ can query for the public key component corresponding to an identity ID_i since $\mathcal{A}$ might want to know the actual X_i corresponding to ID_i. When $\mathcal{A}$ queries ID_i as the i^{th} query, $\mathcal{C}$ does the following:
    · If i=1, $\mathcal{C}$ sets $ID_T = ID_i$ and $X_i = h$, where $h = g^x$, x is unknown and h is the part of the Discrete logarithm problem that $\mathcal{C}$ wants to solve. Then $\mathcal{C}$ stores $\langle \perp, q_i, ID_i \rangle$ in LH_1 list.
    · If $i \neq 1$, choose a random $x_i, q_i \in Z_q^*$, set $X_i = g^{x_i}$, returns $\langle q_i, X_i \rangle$ to the user such that $q_i = H_1(ID_i, X_i)$ and store $\langle x_i, q_i, ID_i \rangle$ in LH_1 list.

  $\mathcal{C}$ sets the first identity queried by $\mathcal{A}$ for the public key component, as the target identity ID_T and it responds accordingly to $\mathcal{A}$. $\mathcal{A}$ will not know which identity is set as the target identity because $\mathcal{A}$ does not know the strategy for selecting the target identity. $\mathcal{A}$ should have asked for at least one query of this kind since $\mathcal{A}$ has to to know the public parameter X_i of a particular ID_i. The only other way for $\mathcal{A}$ to know the X_i corresponding to ID_i is through the private key extract queries. Let the maximum number of queries of this kind to H_1 oracle be $q_{H_1}^*$.

- $\mathcal{O}_{Extract}$**:** When $\mathcal{A}$ queries for private key of ID_i, $\mathcal{C}$ does the following:
  * $\mathcal{C}$ checks the LH_1 list.
  * If the entry corresponding to ID_i in LH_1 is of the form $\langle x_i, q_i, ID_i \rangle$, then $\mathcal{C}$ sets $d_i = x_i + sq_i$ and returns d_i to $\mathcal{A}$.
  * If the tuple corresponding to ID_i in LH_1 is of the form $\langle \perp, q_i, ID_i \rangle$, then $\mathcal{C}$ knows that ID_i is the target identity and then $\mathcal{C}$ aborts.

- $\mathcal{O}_{H_2}$**:** When $\mathcal{A}$ queries for the hash of $\langle ID_i, m_i, W_i \rangle$, $\mathcal{C}$ checks the LH_2 list. If the tuple is already present it returns h_{1i} to $\mathcal{A}$ else $\mathcal{C}$ chooses a random h_{1i} and returns h_{1i} to $\mathcal{A}$ and stores $\langle h_{1i}, W_i, ID_i, m_i \rangle$ in the LH_2 list.

- $\mathcal{O}_{H_3}$**:** When $\mathcal{A}$ queries for the hash of $\langle ID_i, m_i, W_i, h_{1i} \rangle$, $\mathcal{C}$ checks the LH_3 list. If the tuple is already present then it returns h_{2i} to $\mathcal{A}$ else $\mathcal{C}$ chooses a random h_{2i} and returns h_{2i} to $\mathcal{A}$ and stores $\langle h_{2i}, W_i, ID_i, m_i, h_{1i} \rangle$ in the LH_3 list.

- $\mathcal{O}_{Sign}$**:** When $\mathcal{A}$ requests for the signature on m_i by ID_i, $\mathcal{C}$ does the following:
  * If $ID_i \neq ID_T$ then $\mathcal{C}$ knows the private key of ID_i and can generate the signature as per the sign algorithm in the scheme and return $\langle X_i, v_i, h_{1i}, h_{2i} \rangle$ as the signature of ID_i on m_i. It also updates all the hashing lists correspondingly.
  * If $ID_i = ID_T$ then $\mathcal{C}$ does the following:
    · Chooses a random $h_{1i}, h_{2i}, v_i \in Z_q^*$.
    · Sets $v_i = v_i$.
    · Sets $W_i = (X_T^{-h_{2i}} . g^{-s.q_T h_{2i}} . g^{v_i})^{\frac{1}{h_{1i}}}$.
    · Updates the lists LH_2 and LH_3 with the corresponding tuples $\langle h_{1i}, W_i, ID_i, m_i \rangle$ and $\langle h_{2i}, W_i, ID_i, m_i, h_{1i} \rangle$ respectively. If any entry in the list LH_2 or LH_3

is identical to the tuples generated (i.e. $\langle h_{1i}, W_i, ID_i, m_i\rangle$ and $\langle h_{2i}, W_i, ID_i, m_i, h_{1i}\rangle$ respectively), chooses a different v_i, h_{1i}, h_{2i} and repeat the above process.

· Return $\langle X_i, v_i, h_{1i}, h_{2i}\rangle$ to $\mathcal{A}$ as signature on m_i by ID_i

Although the signature $\langle X_i, v_i, h_{1i}, h_{2i}\rangle$ was not generated using the actual signing algorithm. The signature generated is valid due to the following fact:

$$\begin{aligned}(X_T)^{h_{2i}}.(W_i)^{h_{1i}}.(g^s)^{q_T.h_{1i}} &= (X_T)^{h_{2i}}.((X_T^{-h_2 i}.g^{-s.q_T h_{2i}}.g^{v_i})^{\frac{1}{h_{1i}}})^{h_{1i}}.(g^s)^{q_T.h_{1i}}\\ &= (X_T)^{h_{2i}}.(X_T^{-h_2 i}).g^{-s.q_T h_{2i}}.(g^s)^{q_T.h_{1i}}.g^{v_i}\\ &= g^{v_i}\end{aligned}$$

This shows that $\langle X_i, v_i, h_{1i}, h_{2i}\rangle$ will convince $\mathcal{A}$ as a valid signature. There is no need to provide an aggregate oracle since the user can query the signing oracle repeatedly and do the aggregation himself.

– **Forgery Phase:** The adversary $\mathcal{A}$ after issuing all the queries finally generates an aggregate signature $\langle X_1, \ldots, X_n, v_{agg}, W_1, \ldots, W_n\rangle$ on messages $\{m_i\}_{i=1\,to\,n}$ by users with identities $\{ID_i\}_{i=1\,to\,n}$ such that ID_i has signed message m_i. $\mathcal{A}$ wins the game if it is a valid aggregate signature and there is at least one signer ID_k where $k\epsilon\{1\ldots n\}$ where $ID_k = ID_T$ and $\mathcal{A}$ has not asked for a signature query on the corresponding $\langle ID_k, m_k\rangle$ pair to the $\mathcal{O}_{Sign}$ oracle (i.e it is not a trivial forgery). The adversary $\mathcal{A}$ will be able to do this forgery with a probability of $\epsilon^{'}$ which is expressed as follows:

Let ϵ the advantage of the adversary breaking the scheme in existential forgery under chosen target identity and adaptive chosen message. The places where the algorithm can abort are

- In extract case if the adversary asks for the extract query for the ID_T then the algorithm aborts. q_E is the maximum number of extract queries which the adversary can ask. Then the probability that he has not asked for any extract query for ID_T is
$$Pr[q_E(ID_i) \neq ID_T] = 1 - \frac{q_E}{q^*_{H_1}} \tag{1}$$
where $q^*_{H_1}$ is the maximum number of H_1 of type 2 allowed for the adversary.

- After the forgery the algorithm may abort if the adversary has not used ID_T as one of the identities of the signers or if $\mathcal{A}$ has used a m_i for the signature of ID_T for which it has already asked the signing query. The probability that $\mathcal{A}$ produces a valid forgery is
$Pr[ID_i = ID_T$ for some $i = 1, \ldots, n$ and $ID_i \neq ID_T \forall j = 1$ to n $j \neq i] = \frac{n}{2.q^*_{H_1}}$

Combining the two probabilities we get the advantage of the adversary breaking our scheme in adaptive chosen message and adaptive chosen identity attack is given by

$$\epsilon^{'} = \epsilon.(1 - \frac{q_E}{q^*_{H_1}})\frac{n}{2.q^*_{H_1}} \tag{2}$$

$\mathcal{A}$ will be able to produce a valid signature without knowing the secret key of the signer with probability

$$\epsilon = \frac{10(q_S+1)(q_S+q_{H_3}+q_{H_2})}{2^\kappa}.$$

as shown in [17]. Our signature scheme has parameters similar to those used in [17]. The W_i component in our signature is a randomness component corresponding to σ_1 in [17]. We use the randomness in our hash H_2 and H_3. The signature component v_i is analogous

to σ_2 in [17] which uses both the hash values and randomness. Since the construct has similar components as pointed out by Pointcheval we can use forking lemma in our proof. By using forking lemma $\mathcal{C}$ plays again with $\mathcal{A}$ with same random tape and H_2 oracle but different H_3 oracle. Then with a probability $\epsilon^{''} \geq \frac{1}{9}$, the adversary will be able to produce another valid aggregate signature for the set of users. That aggregate signature will be of the form $\langle X_1, \ldots, X_n, v_{agg}^{'}, W_1, \ldots, W_n \rangle$ where all W_i's are same as previous signature. $\mathcal{C}$ using the two valid aggregate signatures does the following:

$$
\begin{aligned}
&v_{agg} = \textstyle\sum_{i=1}^{n} r_i h_{1i} + \sum_{i=1}^{n} d_i h_{2i} \\
&v_{agg}^{'} = \textstyle\sum_{i=1}^{n} r_i h_{1i} + \sum_{i=1}^{n} d_i h_{2i}^{'} \\
&v_{agg} - v_{agg}^{'} = \textstyle\sum_{i=1}^{n} d_i (h_{2i} - h_{2i}^{'}) \\
&v_{agg} - v_{agg}^{'} - \textstyle\sum_{i=1_{i \neq T}}^{n} d_i (h_{2i} - h_{21}^{'}) = d_T (h_{2T} - h_{2T}^{'})
\end{aligned}
$$

$\mathcal{C}$ knows all the private keys other than the target identity's private key. $\mathcal{C}$ also know all the hash values. Thus by dividing the final equation by $(h_{2r} - h_{2i}^{'})$ $\mathcal{C}$ get d_T. But $\mathcal{C}$ knows that $d_T = x_T + sq_T$. Thus the value of $x_T + sq_T$ is determined above. $\mathcal{C}$ know s and q_T. Thus from the computed value of $x_T + sq_T$, if $\mathcal{C}$ subtracts sq_T, $\mathcal{C}$ can find the value of x_T which is the solution for the given instance of the discrete logarithm problem, since $x_T = x$ as per definition.
The total probability by which the challenger will be able to solve the Discrete logarithm problem is given by

$$
\begin{aligned}
\epsilon_0 &= \frac{10(q_S+1)(q_S+q_{H_3}+q_{H_2})}{2^{\kappa}} . (1 - \frac{q_E}{q_{H_1}^{*}}) \frac{n}{2.q_{H_1}^{*}} \frac{1}{9} && (3) \\
&= \frac{10(q_S+1)(q_S+q_{H_3}+q_{H_2}).(1-\frac{q_E}{q_{H_1}^{*}}).n}{2.q_{H_1}^{*}.9.2^{\kappa}} && (4) \\
&= \frac{1}{9} . \frac{10(q_S+1)(q_S+q_{H_3}+q_{H_2}).(1-\frac{q_E}{q_{H_1}^{*}}).n}{2^{\kappa+1}} . \frac{1}{q_{H_1}^{*}} && (5)
\end{aligned}
$$

Hence we have proved that if a polynomial time bounded adversary exists who can break our scheme with a non-negligible probability ϵ_0 then $\mathcal{C}$ can solve the discrete logarithm problem with non-negligible probability as shown. Thus our scheme is secure against existential forgery under adaptive chosen message and adaptive chosen identity attack. □

7 Efficiency Comparison:

In this section we compare the efficiency of our schemes with few existing schemes. We also give some remarks on the efficiency and merits of our schemes over others.

Few Remarks:

- In [11] all the signers have to agree upon a common value ω in order to produce a valid aggregate signature. That will increase the communication complexity. Further weakness of [11] are explained in the appendix.
- Jing Xu et al. [21] achieve partial aggregation and also requires linear number of pairings.
- Javier Herranz [12] also achieves partial aggregation and more over the scheme uses linear number of pairing computations during verification.

Table 1. Comparing various aggregate signature schemes

Scheme	Per Signature		Agg Verification			Agg Signature Size for	Remark
	Exp	Pt.Mul	Exp	Pt.Mul	Pairing	t users & n Signatures	
Gentry et al. [11]	-	3	-	n	3	$2\|\|\mathbb{G}\|\|$	Appendix-A
Jing Xu et al. [21]	-	2	-	-	n+2	$(n+1)\|\|\mathbb{G}\|\|$	n BP + Long Signatures
Javier Herranz [12]	1	-	n	-	n	$t\|\|\mathbb{G}\|\|+1\|\|\mathbb{G}\|\|$	n BP
Cheng et al. [8]	-	3	-	n	2	$2\|\|\mathbb{G}\|\|$	2 Rounds
Our scheme IBAS	1	-	6n+2	-	-	$(t)\|\|\mathbb{G}\|\|+$ $(2n+1)\|\|\mathbb{Z}_q^*\|\|$	No BP + Relatively Short Signatures + 1 Round

LEGEND: $||\mathbb{G}||$ - Size of one group element, $||\mathbb{Z}_q^*||$ - Size of one $\mathbb{Z}_q^*$ element.

- In Cheng et al. [8] scheme achieves full aggregation and also seems efficient. But in this scheme all the signers have to broadcast their respective randomness to other signers so that all agree upon a common randomness finally. This broadcast technique increases the communication complexity enormously and also it is rather like threshold signature and not like a pure aggregate signature.
- It has to be taken into account that the signing part is for each signer. So if n signers are signing the complexity in signing part will be multiplied by n.
- Our scheme IBAS achieves only partial aggregation but does verification without any pairing operation making it the most efficient scheme of all the above. We used a light weight schnorr based signature as proposed by Galindo et al. [10] which is highly efficient and practically implementable.

8 Conclusion

In this paper, we have considered an identity based signature in which the private key for a user is a Schnorr signature on his identity. This private key is generated by the PKG. Besides, the PKG sends a random *'token'* to every user along with his private key. This token cannot be altered by the user and the token can never be used in any identity based encryption scheme. Since, for encryption schemes, only identities are used as public keys. The presence of tokens in the scheme is not a violation to the definition of identity based scheme. However, the concept of *'token'* can be cleverly deployed to avoid all pairing based computations in aggregate signature schemes. We have demonstrated that Galindo et al's [10] signature scheme which uses the concept of *'tokens'* can be used to design an aggregate signature scheme without pairing.

We have addressed the open problem posed by Hwang et al. in [13], which is to design an identity based aggregate signature scheme where the signers need not have to agree on a fresh nonce in advance and do not require a common nonce. Thus an identity based aggregate signature scheme in a non-interactive environment. We have proposed an identity based aggregate signature schemes which uses a variant of schnorr signature, with no pairing operation, which is the first aggregate scheme without pairing, achieves partial aggregation and satisfies the afore mentioned property. We have formally proved the security of the scheme in the random oracle model.

References

1. Ali Bagherzandi and Stanislaw Jarecki. Identity-based aggregate and multi-signature schemes based on rsa. In *Public Key Cryptography - PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 480–498. Springer, 2010.
2. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007*, volume 4596 of *Lecture Notes in Computer Science*, pages 411–422. Springer, 2007.
3. Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *ACM Conference on Computer and Communications Security, CCS 2007*, pages 276–285. ACM, 2007.
4. Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. Cryptology ePrint Archive, Report 2007/438, 2007, Revised on 21-Feb-2010. http://eprint.iacr.org/.
5. Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. New multiparty signature schemes for network routing applications. *ACM Transactions on Information and System Security (TISSEC)*, vol.12(no.1):1–39, 2008.
6. Dan Boneh. Bls short digital signatures. In *Encyclopedia of Cryptography and Security*. Springer, 2005.
7. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
8. Xiangguo Cheng, Jingmei Liu, and Xinmei Wang. Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. In *Computational Science and Its Applications - ICCSA 2005,*, volume 3483 of *Lecture Notes in Computer Science*, pages 1046–1054. Springer, 2005.
9. Dario Fiore and Rosario Gennaro. Making the diffie-hellman protocol identity-based. Cryptology ePrint Archive, Report 2009/174, 2009. http://eprint.iacr.org/ (An extended abstract of this paper appears in the proceedings of CT-RSA 2010).
10. David Galindo and F. D. Garcia. A schnorr-like lightweight identity-based signature scheme. In *In Proceedings of 2nd African International Conference on Cryptology, AfricaCrypt 2009*, Lecture Notes in Computer Science 5580, pages 135–148, 2009.
11. Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Public Key Cryptography - PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 257–273. Springer, 2006.
12. Javier Herranz. Deterministic identity-based signatures for partial aggregation. *The Computer Journal*, vol-49(no-3):322–330, 2006.
13. Jung Yeon Hwang, Dong Hoon Lee, and Moti Yung. Universal forgery of the identity-based sequential aggregate signature scheme. In *Computer and Communications Security, ASIACCS 2009*, pages 157–160. ACM, 2009.
14. Joseph K. Liu, Joonsang Baek, Jianying Zhou, Yanjiang Yang, and Jun Wen Wong. Efficient online/offline identity-based signature for wireless sensor network. Cryptology ePrint Archive, Report 2010/003, 2010. http://eprint.iacr.org/.
15. Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, 2006.
16. Gregory Neven. Efficient sequential aggregate signed data. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 52–69. Springer, 2008.
17. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, vol-13(no-3):361–396, 2000.
18. 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. Cryptology ePrint Archive, Report 2009/290, 2009. http://eprint.iacr.org/.

19. Zhu Wang, Huiyan Chen, Ding feng Ye, and Qian Wu. Practical identity-based aggregate signature scheme from bilinear maps. *Journal of Shangai Jiatong University*, vol-13(no-6):684–687, 2008.
20. Yiling Wen and Jianfeng Ma. An aggregate signature scheme with constant pairing operations. In *International Conference on Computer Science and Software Engineering, CSSE 2008*, pages 830–833. IEEE Computer Society, 2008.
21. Jing Xu, Zhenfeng Zhang, and Dengguo Feng. Id-based aggregate signatures from bilinear pairings. In *Cryptology and Network Security, CANS-2005*, volume 3810 of *Lecture Notes in Computer Science*, pages 110–119. Springer, 2005.

APPENDIX-A:

In this section, we show the various weaknesses of [11]. Though it is claimed to be currently the most efficient scheme it has the following weakness.

1. According to the scheme in [11], the signers have to store all the ω they have used previously in a database in order to avoid the re-use of randomness. Every time the signer before signing needs to check whether the current ω was not previously used for any signature generation on any message. This leads to increase in storage cost and checking cost, becoming a huge overhead.
2. Not only this, the common randomness ω chosen by the first signer should satisfy the constraint that it was not used previously for signing any message by any o the other signers participating in the aggregation process. Even if $n-1$ signers agree and n^{th} signer disagrees then they have to run the protocol again by picking up a new ω value. This accounts to a lot of wastage of time and network bandwidth.
3. Any signer, if he/she reuse a ω value even once with or without his/her knowledge then universal forgery of their signature is possible. The universal forgery of signature is as follows:
   - Let $\langle S_1, T_1 \rangle$ be a signature on m_1 by ID using the value ω.
   - Let $\langle S_2, T_2 \rangle$ be a signature on m_2 by ID using the same ω
   - The signature components is of the form

$$S_1 = r_1 P_\omega + D_1 + c_1 D_2 \tag{6}$$
$$T_1 = r_1 P \tag{7}$$
$$S_2 = r_2 P_\omega + D_1 + c_2 D_2 \tag{8}$$
$$T_2 = r_2 P \tag{9}$$

     where r_1, r_2 is unknown random numbers, $P_\omega = H_2(\omega)$, $c_1 = H_3(m_1, ID, \omega)$, $c_2 = H_3(m_2, ID, \omega)$.

   - Subtracting 6 from 8 we get

$$S^* = (r_2 - r_1) P_\omega + (c_2 - c_1) D_2 \tag{10}$$

     Dividing by $(c_2 - c_1)$ we get, $S^{**} = \frac{r_2 - r_1}{c_2 - c_1} P_\omega + D_2$

   - Compute a new hash value $c_3 = H_3(m_3, ID, \omega)$ where m_3 is some random message.

   - Multiply S^{**} by c_3 and we get $S^{'} = \frac{r_2 - r_1}{c_2 - c_1} c_3 P_\omega + c_3 D_2$

– Dividing 6 by c_1 and dividing 8 by c_2 we get the two equations

$$S_1^{'} = \frac{r_1}{c_1}P_\omega + \frac{1}{c_1}D_1 + D_2 \tag{11}$$

$$S_2^{'} = \frac{r_2}{c_2}P_\omega + \frac{1}{c_2}D_1 + D_2 \tag{12}$$

– Subtracting 12 from 11 we get
$S_3^{'} = (\frac{r_1}{c_1} - \frac{r_2}{c_2})P_\omega + (\frac{1}{c_1} - \frac{1}{c_2})D_1$

– Dividing $S_3{'}$ by $(\frac{1}{c_1} - \frac{1}{c_2})$ we get
$S_3^{''} = \frac{(\frac{r_1}{c_1} - \frac{r_2}{c_2})}{(\frac{1}{c_1} - \frac{1}{c_2})}P_\omega + D_1$

$S_3^{''} = \frac{r_1c_2 - r_2c_1}{c_2 - c_1}P_\omega + D_1$

– Adding $S^{''}$ to $S^{'}$ we get $S_3 = (\frac{r_2 - r_1}{c_2 - c_1}c_3 + \frac{r_1c_2 - r_2c_1}{c_2 - c_1})P_\omega + D_1 + c_3D_2$

– $S_3 = r^*P_\omega + D_1 + c_3D_2$ where $r^* = (\frac{r_2 - r_1}{c_2 - c_1}c_3 + \frac{r_1c_2 - r_2c_1}{c_2 - c_1})$

– We can derive $T_3 = r^*P$ also knowing the T_1 and T_2 values as follows:

Multiplying T_1 by $(\frac{c_2 - c_3}{c_2 - c_1})$ and multiplying T_2 by $(\frac{c_3 - c_1}{c_2 - c_1})$ and adding the two we get
$T_3 = (\frac{c_2 - c_3}{c_2 - c_1}r_1 + \frac{c_3 - c_1}{c_2 - c_1}r_2)P$

$= (\frac{r_2 - r_1}{c_2 - c_1}c_3 + \frac{r_1c_2 - r_2c_1}{c_2 - c_1})P$

$T_3 = r^*P$

– Thus S_3 and T_3 is a valid signature on message m_3 since its of the standard signature format of Gentry et al.'s scheme where $c_3 = H_3(m_3, ID, \omega)$.

Thus universal forgery of signature is possible in case of Gentry et al.'s scheme. The signer has to be very careful that he/she does not reuse the ω value. So the storage and checking of all used ω values becomes essential in their scheme. The extension which the authors have stated for using ω repeatedly is to get different private keys equal to number of times ω is reused. That again is not a viable solution since the user will have no idea as to how many times he will reuse ω if he does.

4. Using a single signature on a message one can generate a different signature which is valid on the same message by the same user. Though this is not a flaw in the scheme it is considered as a weakness in certain scenarios (strong unforgeability is not satisfied).