A Suite of Identity Based Aggregate Signatures and a Multi-Signature Scheme from RSA

S. Sharmila Deva Selvi, S. Sree Vivek*, C. Pandu Rangan*

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

Abstract. Fully aggregateable identity based signature schemes without prior communication between the signing parties is an interesting issue in identity based cryptography. On this front, we identify that deterministic identity based signature schemes lead to full aggregation of signatures without the aforementioned overhead. Inspired by Shamir's identity based signature scheme, we propose a deterministic identity based signature scheme which is also based on RSA. Based on this newly proposed deterministic identity based signature scheme, we design a suite of four identity based aggregate signature schemes with different properties. The first two schemes are deterministic identity based aggregation signature schemes, supporting full aggregation for general and ordered sequential aggregation respectively. The third and fourth schemes are non-deterministic aggregate signature schemes, supporting full aggregation for general and ordered sequential aggregation respectively. We formally prove the schemes to be existentially unforgeable in the random oracle model. We also propose an efficient identity based multi-signature scheme which achieves aggregation in one round. In the passing, we also show the weakness in the proof of the only PKI based aggregate signature scheme in the standard model by Lu et al.

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

1 Introduction

The concept of Identity Based Cryptosystem (IBC) was introduced by Adi Shamir [19] in 1984. The distinguishing characteristic of identity based cryptography is the ability to use any string, that uniquely identifies the user in the system as his public key. In particular, this string may be the email address, telephone number, or any publicly available parameter that is unique to that user. The corresponding private key can only be derived by a trusted Private Key Generator (PKG) who uses a master secret key, for deriving the private key of users. Identity based cryptosystem removes the need for senders to verify the receiver's public key before sending out an encrypted message or verifying a signature. It provides a more convenient alternative to conventional Public Key Infrastructure (PKI) based system. Since 1984, several identity based signature (IBS) schemes have been proposed [20] [10] [22] [14]. Different variations such as proxy, group, ring, threshold signatures, have been proposed in the identity based settings depending on various practical applications.

In several real-life situations, it is advantageous to handle a collection of signed documents together rather than handling them in isolation. 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. Thus, the communication cost can be significantly reduced if we transmit σ_{agg} instead of transmitting $\sigma_1, \ldots, \sigma_t$ individually. A similar remark holds good even for storage requirements when we archive σ_{agg} (instead of $\sigma_1, \ldots, \sigma_t$).

* 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

Various Forms of Aggregate Signatures: The underlying signature scheme may be PKI based or identity based. In the PKI based setting aggregation is easily achieved for example [8], [2]. However, aggregation of identity based signatures are far difficult and subtle, we will see the details below.

Aggregation Using Deterministic Signature Schemes: A generic deterministic signature can be represented as $\sigma = \langle M \rangle$, where M is just the encryption of the message digest with the private key of the signer. Let σ_i be the signature generated by user U_i on the message m_i, for $i = 1$ to t. Here, $|\{\sigma_i, \ldots, \sigma_t\}| = t$, but $|U_1, \ldots, U_t| = k$, where $1 \leq k \leq t$. That is, we allow a user to generate several signatures on different messages. In particular, when $k = 1$,the aggregation scheme works on t signatures on t different messages generated by a single user. Let σ_{agg} be the aggregate signature. In general, naive or direct aggregation will lead to σ_{agg}, where $|\sigma_{agg}| = \mathcal{O}(t + k)$. If $|\sigma_{agg}|$ is $\mathcal{O}(k)$, i.e., if the size of σ_{agg} depends only on the number of users and not on the number of messages, then it is called *partial* aggregation. If $|\sigma_{agg}|$ is independent of both t and k and if its size is proportional to the size of a single signature of the scheme, then it is called $full$ aggregation. Till date, [13] is the only identity based signature scheme that produces deterministic signature. Even though the scheme in [13] produces deterministic signatures, the private key in their scheme uses random values and this value is to be sent along with the aggregate signatures; which makes it partially aggregateable according to our classification. Moreover, the scheme in [13] uses bilinear pairing. To the best of our knowledge, there is no deterministic identity based aggregate signature scheme which achieves full aggregation.

Aggregation Using Non-Deterministic Signature Schemes: The generic format of a non-deterministic signature is $\sigma = \langle K, R, M \rangle$, where K is the nonce (random value) used for generating the private key of the signer, R is the random value used for generating the signature and M is the encryption of the message digest with the private key of the signer and the random values. Let $\sigma_i = \langle K_i, R_i, M_i \rangle$, for $1 \leq i \leq t$ be a collection of t signature to be aggregated. The presence of t independent random values pose some specific problems in the aggregation process. Besides, the independent random values (R_i's), K_i's used in the generation of private keys of the users may also challenge the process of aggregation. Thus we need to formulate the aggregation of non-deterministic signatures in (slightly) different flavors than deterministic signatures.

Let $\sigma_{agg} = \langle K_{agg}, R_{agg}, M_{agg} \rangle$ denote the aggregation of t signatures $\sigma_i = \langle K_i, R_i, M_i \rangle$, for $1 \leq i \leq t$, generated by $U_1, \ldots, U_t$ respectively. We let $|\{U_1, \ldots, U_t\}| = k$, where $1 \leq k \leq t$ to allow the possibility of some users generating more than one signed document. We further assume, without loss of generality, that $\{K_1, \ldots, K_t\} = k$ (The number of random values used in generating the private keys of the k distinct users participating in the aggregation process), $\{R_1, \ldots, R_t\} = t$ (the number of random values used in the generation of the signature) and $\{M_1, \ldots, M_t\} = m$ (the number of message digests encrypted with the private key in the aggregate signature).

A naive clubbing of t non-deterministic signatures generated by k signers will lead to a signature of size $\mathcal{O}(k+t+m)$. In this scenario, there is no advantage what so ever. Consider that an aggregate signature σ_{agg} of $\sigma_i = \langle K_i, R_i, M_i \rangle$, for $1 \leq i \leq t$ is in one of the the following forms:

- $\sigma_{agg} = \langle K_1, \ldots K_k, R_1, \ldots, R_t, M_{agg} \rangle$ - The random values K_i's and R_i's are propagated in full without any compression but the third components are all combined to form M_{agg}. The size of the aggregate signature is $|\sigma_{agg}| = \mathcal{O}(k + t)$.
- $\sigma_{agg} = \langle R_1, \ldots, R_t, M_{agg} \rangle$ - There are no K_i's and R_i's are propagated in full without any compression but the third components are all combined to form M_{agg}. The size of the aggregate signature is $|\sigma_{agg}| = \mathcal{O}(t)$.

In both the cases the non-deterministic aggregate signature scheme is said to achieve *partial* aggregation.

If all the three parts are fully aggregated, i.e. $\sigma_{agg} = \langle K_{agg}, R_{agg}, M_{agg} \rangle$ or there is no randomness involved in generation of the private keys of the signers, i.e. $\sigma_{agg} = \langle R_{agg}, M_{agg} \rangle$, then the scheme is said to achieve $full$ aggregation, i.e. the size of σ_{agg} is proportional to the size of a single signature and is independent of number of signers or signatures. To the best of our knowledge, there is one provably secure identity based aggregate signature scheme [21] which achieves partial aggregation and four identity based aggregate signature schemes, which achieve full aggregation, namely [11], [12], [4] and [1].

Ordered Sequential vs Sequential vs General: Apart from the above classification, aggregate signatures can be classified into three further subtypes, namely Ordered Sequential, Sequential and General aggregate signatures. An *Ordered Sequential* aggregation is a type of aggregation where, the signatures from different signers are aggregated one by one and the order of the signer in the list of signers play a role in the

verification of the signature. The aggregate signature will not be valid if the order of the signers is not considered during generation/verification. This type of aggregate signature can be used to find the path travelled by the data packets from source to destination in routing by using a single aggregate signature. In *sequential* aggregation, the signature is aggregated in any order but the aggregation process is carried out by aggregating the signatures one after the other, where each signer aggregates his signature to the previously aggregated signature. Here, the order of the signer have no role to play during the signature generation/verification process. *General* aggregation is the most common way of aggregating signatures, which is done by any user called the aggregator (one of the signers or any third party). The aggregator collects the signature from all the signers and generates the aggregate signature. The second and third type of aggregate signatures find applications in wireless network, where the major constraint is communication complexity, the use of efficient aggregate signature helps in reducing the amount of data to be communicated.

Usually, in sequential aggregation, the number of interactions (rounds) between the signers is linear with the number of signers participating in the aggregation. This is because, the aggregation is done linearly, i.e. the i^{th} signer signs the i^{th} message and aggregates it with the signature aggregated so far by $i-1$ signers and sends the newly formed aggregate signature to the $i+1^{th}$ signer, for $1 \leq i \leq t$. The situation in general aggregation is slightly different. Here, the signature is aggregated by an aggregator after collecting all the individual signatures. The interaction between the signers and the aggregator is not counted to be a round in general aggregation. However, if there is any sort of communication between the signers before generating the individual signature, then they are counted as rounds in the protocol. Since there are no, general aggregation schemes without any communication among the signers in the identity based setting (to the best of our knowledge), it is an interesting issue to look at.

Identity based Multi-Signature: A *Multi-Signature scheme* combines several signed documents, say σ_1, $\ldots$, σ_t on a single message m by different 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.

Related Work: Many well known PKI based aggregate signatures are available in the literature [8, 16, 2, 17, 18]. However, as the focus of this paper is on identity based aggregate signature scheme, we will not compare the PKI based schemes with ours. Most of the efficient aggregate signature schemes in the PKI setting are deterministic [8, 2, 17]. The first identity based aggregate signature scheme that achieves full aggregation was proposed in [11] by Cheng et al. Their scheme uses bilinear pairing and requires large setup cost because the signers essentially broadcast their individual random values to form a single random value. Moreover, the fact that the signature cannot be generated until all of the signers contribute in the first round, makes the scheme less practical. Gentry et al. proposed another scheme in [12], based on bilinear maps and the security of the scheme relies on the Gap Diffie Hellman problem. The weakness of the scheme is that, the signers of a given aggregate signature must agree on a common random value which was never used by any of the users before to generate a signature. If the signer ever re-uses a random value in two different aggregate signatures, a total break (private key of the users are revealed) of the system is possible. Recently, Boldyreva et al. [5] proposed a sequential aggregate signature scheme (in-fact, the security of their original schemes [4] and [6] were flawed due to the assumption used to prove them was actually not hard to solve in polynomial time, as pointed out by Hwang et al. [15]). Their new scheme [5] was based on the hardness of a CDH-type problem that raised from their scheme and uses bilinear pairings. 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 [11]) 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.

There are three well known identity based multi-signature schemes [12] [3] and [1] in the literature. The scheme in [12] is based on bilinear pairing and the security of the scheme relies on the Gap - Diffie Hellman assumption. The schemes in [3] and [1] are based on RSA assumption and constructs IBMS with 3 and 2 rounds of communication (broadcast) among the signers respectively.

Our contribution: It is possible to achieve full aggregation in deterministic identity based signatures because there is no random values involved in generating the signature, provided the private key generation protocol is not a randomized algorithm. It is well known that Boneh et al.'s aggregate signature scheme [8] in the PKI based setting achieves full aggregation. We try to explore the direction of designing a deterministic

identity based signature scheme based on RSA. It is quite natural to design such a scheme from RSA and as a consequence, we propose a deterministic identity based signature scheme based on RSA and extend it to various other aggregate signature schemes in this paper.

We propose a secure deterministic identity based (i.e. the same signature is generated for a given message, even if the signature algorithm is invoked for multiple number of times. Examples are available in the PKI setting: [9], FDH-RSA) signature scheme, which is based on RSA. Based on this signature, we propose four identity based aggregate signature schemes, whose securities are also based on RSA assumption. The first two schemes are deterministic identity based aggregate signature schemes, for general aggregation and ordered sequential aggregation respectively. The third and fourth schemes produce non-deterministic signatures, for general aggregation and ordered sequential aggregation respectively. Our scheme achieves full aggregation in the identity based setting, which makes it more interesting. Our first scheme affirmatively settles the open problem by Hwang et al. in [15], which is to design an identity based aggregate signature scheme without the requirement of a common random value being established before the aggregate signature is generated. Moreover, our schemes are more efficient than the scheme in [1], because aggregation can be done in a single round in our schemes where as the scheme in [1] requires two rounds of communication between the signers. Our schemes are based on RSA and we prove the security of our schemes in the random oracle model. One of the key features of our deterministic signature scheme is that it is possible to convert one aggregate signature scheme with a particular property, to a scheme with another property in the suite, by just altering the definition of a single hash function (namely, $H_2(.)$). We also propose an identity based multi-signature scheme (IBMS), which does not require any prior communication between the signers to generate the multi-signature. Thus our scheme turns out to be more efficient when compared to all the existing schemes. .

Finally in the appendix, we show the weakness in the proof of the only PKI based aggregate signature scheme in the standard model by Lu et al. [16]. This leaves a very important open issue that whether it is possible to prove the scheme by Lu et al. [16] in the standard model.

Summary of State of the Art: In summary, aggregate signature schemes can be classified based on their attributes and properties as follows:

$$\left\{ \begin{array}{c} Deterministic \\ Non-Deterministic \end{array} \right\} \times \left\{ \begin{array}{c} Partial \\ Full \end{array} \right\} \times \left\{ \begin{array}{c} OrderedSequential \\ Sequential \\ General \end{array} \right\}$$

The following table shows the current state of the art in the research of identity based aggregate signatures which achieves full aggregation.

Property ..... Schemes	Type of Sys (PKI/IB)	Security (SO/RO)	Security Status (PS /NK)	Agg Sign Len	Sign Cost (/ user)	Agg Verify (t users)	Hard Prob	Sign Type (D/ND)	Rounds	Agg Mode (G/OS /S)
[11]	IB	RO	PS	$2\|\mathbb{G}\|$	3[M]	2[P]+t[M]	CDH	ND	2	G
[1]	IB	RO	PS	$2\|\mathbb{Z}_n^*\| + \|\kappa\| + \|log(l)\|$	2[E]	t[E]	RSA	ND	2	G
[5]	IB	RO	PS	$3\|\mathbb{G}\|$	7[M]	6[P]+t[M]	CDH-type	ND	-	S
[12]	IB	RO	PS	$2\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$	5[M]	3[P]+t[M]	GAP-DH	ND	-	S
[16]	PKI	SO	NK†	$2\|\mathbb{G}\|$	4[M]+ ($t.k$+5) [A]	2[P]+ $t.k$[A]+ t[$\mathbb{G}_T$M]	WSS	ND	-	S

Table-1: Summary of existing IBAS schemes.

The following table summarizes the current state of the art in the research of identity based multi-signatures.

Property ..... Schemes	Type of Sys (PKI/IB)	Security (SO/RO)	Security Status (PS /NK)	Agg Sign Len	Sign Cost (/ user)	Agg Verify (t users)	Hard Prob	Sign Type (D/ND)	Rounds
[12]	IB	RO	PS	$2\|\mathbb{G}\|$	3[M]	3[P]	GAP-DH	ND	1
[3]	IB	RO	PS	$\|\mathbb{Z}_n^*\| + \kappa$	1[E]	2[E]	RSA	ND	3
[1]	IB	RO	PS	$2\|\mathbb{Z}_n^*\|+\|\kappa\|$	4[E]	5[E]	RSA	ND	2

Table-2: State of the art survey of IBMS schemes

The terms used in the table are explained here. IB- Identity Based System, PKI PKI Based System, SO- Standard Model, RO- Random Oracle Model, PS- Provably Secure, NK- Status Not Known, ND - Nondeterministic Signature, D- Deterministic Signature, G- General aggregation, OS- Ordered Sequential Signature, S- Sequential Signature, κ is the security parameter of the scheme, [E]-Exponentiation in $\mathbb{Z}_n^*$, [M]- Scalar Point Multiplication in $\mathbb{G}$, [P]- Bilinear Pairing Operation, Gap-DH- Gap-Diffie-Hellman, [$\mathbb{G}_T$M]- Exponentiation in $\mathbb{G}_T$ and WSS- Waters Signature Scheme.

Computational Assumption: We review the computational assumption related to the protocols we discuss.

The Strong RSA Problem: Given a randomly chosen RSA modulus n and a random $c \in \mathbb{Z}_n^*$, finding $b > 1$ and $a \in \mathbb{Z}_n^*$, such that $c \equiv a^b \bmod n$ is the strong RSA problem.

The Strong RSA Assumption: The advantage of any probabilistic polynomial time algorithm $\mathcal{F}$ in solving the strong RSA problem in $\mathbb{Z}_n^*$ is defined as:

$$Adv_{\mathcal{F}}^{sRSA} = Pr\left[\mathcal{F}(p, q, n, c) \rightarrow \{a, b\} \mid (a \in \mathbb{Z}_n^*, b > 1) \wedge (c \equiv a^b \, mod \, n)\right]$$

The *strong RSA Assumption* is that, for any probabilistic polynomial time algorithm $\mathcal{F}$, the advantage $Adv_{\mathcal{F}}^{sRSA}$ is negligibly small.

2 Generic Model

In this section, we describe the generic frame work for a identity based signature scheme and an aggregate signature scheme. An identity based aggregate signature scheme (IBAS) consists of following six algorithms. The frame work of a deterministic identity based signature scheme (D-IBS) consists of the first four algorithms described below, namely **Setup**, **KeyGen**, **Sign** and **Verify**. If the signature scheme is deterministic the signature σ will not be randomized, i.e. the signature for a message is always the same.

- **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 a signature σ on message m by user U with identity ID_i checks whether σ is a valid signature on message m by ID_i. If true it outputs "$Valid$", else it outputs "$Invalid$".
- **AggregateSign:** On receiving the various signatures $(\sigma_i)_{i=1\,to\,t}$ from different users $(U_i)_{i=1\,to\,t}$, 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 $(m_i, ID_i)_{i=1\,to\,t}$.

  Note: For sequential aggregation, each user contribute in the generation of aggregate signature by aggregating his own signature to the aggregate signature generated by the signers so far.
- **AggregateVerify:** This algorithm on input of an aggregate signature σ_{agg}, the list for $(m_i, ID_i)_{i=1\,to\,t}$ and the *params* checks whether σ_{agg} is a valid aggregate signature on m_i by ID_i for all $i = 1$ to t. If true, it outputs "$Valid$", else outputs "$Invalid$".

3 Security Model

3.1 Existential Unforgeability

We define the security model for the existential unforgeability of an IBAS scheme under chosen-identity and chosen-message attack in this section. An IBAS scheme is secure against existential forgery under chosen-identity and chosen-message attack if no probabilistic polynomial time algorithm $\mathcal{F}$ has non-negligible advantage in the following game.

- **Setup phase:** The challenger $\mathcal{C}$ runs the setup algorithm and generates the system public parameters $params$ and the master secret key msk. Now, $\mathcal{C}$ gives $params$ to the forger $\mathcal{F}$ and keeps msk secret. $\mathcal{F}$ gives the target identity ID_T and the target message m^* on which $\mathcal{F}$ indents to forge.
- **Training phase:** After the setup is done, $\mathcal{F}$ starts interacting with $\mathcal{C}$ by querying the various oracles provided by $\mathcal{C}$ in the following way:
  - **Random oracles:** Hash functions listed in the $params$ for any arguments, the response will be given by $\mathbb{C}$ treating each hash function as a random function.
  - **KeyGen oracle:** When $\mathcal{F}$ makes a query with ID_i as input, $\mathcal{C}$ outputs D_i, the private key of ID_i to $\mathcal{F}$, provided $\mathcal{C}$ knows the private key for the queried identity. It should be noted that $\mathcal{F}$ should not query the private key corresponding to ID_T
  - **Signing oracle:** When $\mathcal{F}$ makes a signing query with ID_i, message m_i, $\mathcal{C}$ outputs a valid signature σ_i on m_i by ID_i. $\mathcal{F}$ is also allowed to query the signature corresponding to the identity ID_T on any other message m where $m \neq m^*$.

    Note: It should be noted that Aggregate sign oracle is not required for the adversary because aggregation is a public process and any third party who has t signatures can combine all the signatures to form an aggregate signature. Thus, the forger $\mathcal{F}$ can always generate the aggregate signature after querying t individual signatures. However, in a sequential aggregation $\mathcal{F}$ sends a so far aggregated signature σ_{agg} along with a message, identity pair (m_i, ID_i) and requests for the aggregate signature. $\mathcal{C}$ generates the current aggregate signature σ_{agg} and sends it to $\mathcal{F}$.
- **Forgery phase:** $\mathcal{F}$ outputs an aggregate signature σ_{agg} for signatures $(\sigma_i)_{i=1\,to\,t}$ from the users $(ID_i)_{i=1\,to\,t}$ on messages $(m_i)_{i=1\,to\,t}$ where, at least one identity in the list of identities is the target identity, i.e. $ID_T \in \{ID_i\}_{i=1\,to\,t}$, for which the private key was not queried by $\mathcal{F}$ and the message corresponding to ID_T is m^*. The forger $\mathcal{F}$ wins the game if σ_{agg} is a valid aggregate signature and $\mathcal{F}$ has not queried for the signature corresponding to (ID_T, m^*) pair from the sign oracle.

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

Note: During the Forgery Phase for D-IBS, $\mathcal{F}$ outputs an identity based signature σ^* which is a valid signature on message m^* by the identity ID_T and $\mathcal{F}$ has not queried for the signature corresponding to (ID_T, m^*) pair from the sign oracle.

4 Deterministic Identity Based Signature Scheme (D-IBS)

Inspired by Shamir's identity based signature scheme, we propose a new deterministic identity based signature scheme. We also prove the scheme to be existentially unforgeable under chosen-message and chosen-identity attack in the random oracle model. We use the notation $\mathbb{Z}_n^{odd}$ to represent the odd numbers from $[3, n]$. Throughout the paper, in order to choose a random odd number from the range $[3, n]$, we randomly pick an element in $\mathbb{Z}_n$ and check whether it is odd, if it is odd, we accept it, else we subtract 1 from the chosen number. These numbers are represented as $\mathbb{Z}_n^{odd}$. The deterministic identity based signature scheme consists of the following algorithms:

- **Setup(κ):** The PKG generates $params$ and msk by performing the following:
  - Chooses two primes p and q, such that $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are also primes.
  - Computes $n = pq$ and the Euler's totient function $\phi(n) = (p-1)(q-1)$.
  - Chooses $e \in_R \mathbb{Z}_n^{odd}$, such that $|e| = |n|/4$ and compute d such that $ed \equiv 1 \bmod \phi(n)$.
  - It also chooses two cryptographic hash functions $H_1 : \{0,1\}^* \times \{0,1\} \rightarrow \mathbb{Z}_n^*$ and $H_2 : \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_n^{odd}$.

- Now, PKG publicizes the system parameters, $params = \langle n, e, H_1, H_2 \rangle$ and keeps the factors of n, namely p, q and d as the master secret key msk.

– **KeyGen(ID_i):** The user U_i provides his identity ID_i to PKG. The PKG performs the following to find out the private key of the corresponding user:
  - Compute $g_{i0} = H_1(ID_i, 0)$ and $g_{i1} = H_1(ID_i, 1)$.
  - Compute $d_{i0} = (g_{i0})^d$ and $d_{i1} = (g_{i1})^d$.
  - The private key $D_i = \langle d_{i0}, d_{i1} \rangle$ is sent to user U_i through a secure channel.

– **Sign(m, ID_i, D_i):** To generate a deterministic signature on a message m, the user U_i performs the following:
  - Computes $h = H_2(m, ID_i)$.
  - Computes $\sigma = d_{i0}(d_{i1})^h$.

  Now, σ is the signature on m by user U_i with identity ID_i.

– **Verify(m, σ, ID_i):** In order to verify the validity of a signature σ with respect to the identity ID_i and message m, the verifier performs the following:
  - Computes $g_{i0} = H_1(ID_i, 0)$ and $g_{i1} = H_1(ID_i, 1)$.
  - Computes $h' = H_2(m, ID_i)$.
  - Checks whether $\sigma^e \stackrel{?}{=} g_{i0}(g_{i1})^{h'}$
  - If the above check holds, outputs "$Valid$", otherwise outputs "$Invalid$".

Correctness of verification:

$L.H.S = \sigma^e = (d_{i0}(d_{i1})^{h'})^e = (g_{i0}^d(g_{i1}^d)^{h'})^e = g_{i0}(g_{i1})^{h'} = R.H.S$

Existential Unforgeability of D-IBS:

Theorem 1. *Our identity based signature scheme (D-IBS) is EUF-D-IBS-CMA secure in the random oracle model under chosen-message and chosen-identity attack, if the strong RSA problem is intractable in $\mathbb{Z}_n^*$, where p, q, $(p-1)/2$ and $(q-1)/2$ are large prime numbers.*

Proof: Suppose a forger $\mathcal{F}$ is capable of breaking the EUF-D-IBS-CMA security of the D-IBS scheme and a challenger $\mathcal{C}$ is challenged with an instance of the strong RSA problem say $\langle n, c \in \mathbb{Z}_n^* \rangle$, where n is a composite number with two big prime factors p and q, also $(p-1)/2$ and $(q-1)/2$ are primes. $\mathcal{C}$ can make use of $\mathcal{F}$ to compute a and b such that $c \equiv a^b \bmod n$, by playing the following interactive game with $\mathcal{F}$.

Setup: $\mathcal{C}$ begins the game by setting up the system parameters as in the D-IBS scheme. $\mathcal{C}$ takes n as in the instance of the strong RSA problem, chooses $e \in_R \mathbb{Z}_n^{odd}$, such that $|e| = |n|/4$ and sends $params = \langle n, e \rangle$ to $\mathcal{F}$. $\mathcal{C}$ also designs the two hash functions H_1 and H_2 as random oracles $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$. $\mathcal{C}$ maintains two lists L_1 and L_2 in order to consistently respond to the queries to the random oracles $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$ respectively. Since, the proof is under selective-identity and selective-message attack, $\mathcal{F}$ provides the target identity ID_T and the target message m^* to $\mathcal{C}$, on which $\mathcal{F}$ $\mathcal{F}$ indents to forge.

Training Phase: $\mathcal{F}$ performs a series of queries to the oracles provided by $\mathcal{C}$. The descriptions of the oracles and the responses given by $\mathcal{C}$ to the corresponding oracle queries by $\mathcal{F}$ are described below. For the sake of simplicity, we assume that $\mathcal{O}_{H_1}(.)$ oracle is queried with ID_i as input, before any other oracle is queried with the corresponding identity, as one of the input parameters.

Oracle $\mathcal{O}_{H_1}(ID_i, l \in \{0,1\})$: For answering the $\mathcal{O}_{H_1}$ query, $\mathcal{C}$ checks whether a tuple of the form $\langle ID_i, l, d_{il}, g_{il} \rangle$ exists in list L_1 if so returns the corresponding g_{il} as the response. If a tuple does not exist, $\mathcal{C}$ checks whether $ID_i \stackrel{?}{=} ID_T$.

– If $ID_i \neq ID_T$, then $\mathcal{C}$ performs the following:
  - Chooses $d_{il} \in_R \mathbb{Z}_n^*$.
  - Computes $g_{il} = (d_{il})^e$.

– If $ID_i = ID_T$, then $\mathcal{C}$ performs the following:
  - If $l = 0$, then $\mathcal{C}$ does the following:
    * Chooses $r \in_R \mathbb{Z}_n^{even}$ (r is stored by $\mathcal{C}$ for access in other oracles.)
    * Sets $g_{il} = c^{-r}$ (Note that c is taken from the strong RSA instance).
    * The corresponding d_{il} is set to be " $-$ ". That is, $g_{T0} = c^{-r}$ and $d_{T0} =$ " $-$ "
  - If $l = 1$, then $\mathcal{C}$ does the following:

    - Sets $g_{il} = c$ (Note that c is taken from the strong RSA instance).
    - The corresponding d_{il} is set to be " $-$ ". That is, $g_{T1} = c$ and $d_{T1} =$ " $-$ "
- Stores the tuple $\langle ID_i, l, d_{il}, g_{il} \rangle$ in list L_1 and returns g_{il} as the response.

Remark: Note that $\mathcal{C}$ has not computed the private keys for ID_T, but this will not cause any problem as $\mathcal{C}$ need not hand over the private keys of ID_T to $\mathcal{F}$. For all other identities, $\mathcal{C}$ has proper private and public key.

Oracle $\mathcal{O}_{H_2}(m_i, ID_i)$*:* To respond to this query, $\mathcal{C}$ checks whether a tuple of the form $\langle m_i, ID_i, r_i, h_i \rangle$ exists in list L_1, if so returns the corresponding h_i as the response. If a tuple does not exist, $\mathcal{C}$ checks whether $m_i \stackrel{?}{=} m^*$ and $ID_i \stackrel{?}{=} ID_T$.

- If $m_i \neq m^*$ and $ID_i \neq ID_T$, then $\mathcal{C}$ performs the following:
  - Chooses $h_i \in_R \mathbb{Z}_n^{odd}$ and sets $r_i =$ " $-$ ".
- If $m_i \neq m^*$ and $ID_i = ID_T$, then $\mathcal{C}$ performs the following:
  - Chooses $r_i \in_R \mathbb{Z}^{odd}$, such that $|r_i| \leq |n|/4$.
  - Computes $h_i = er_i + r$.
- If $m_i = m^*$ and $ID_i = ID_T$, then $\mathcal{C}$ performs the following:
  - Computes $h_i = r + 1$ and sets $r_i =$ " $-$ ". (Note that r is chosen by $\mathcal{C}$ during the $\mathcal{O}_{H_1}$ query)
- $\mathcal{C}$ adds the tuple $\langle m_i, ID_i, r_i, h_i \rangle$ in list L_1 and returns the corresponding h_i as the response.

Oracle $\mathcal{O}_{KeyGen}(ID_i)$*:* $\mathcal{C}$ checks whether tuples of the form $\langle ID_i, 0, d_{i0}, g_{i0} \rangle$ and $\langle ID_i, 1, d_{i1}, g_{i1} \rangle$ exists in the list L_1, if so returns the corresponding d_{i0} and d_{i1} as the private keys corresponding to the identity ID_i.
Oracle $\mathcal{O}_{Sign}(m_i, ID_i)$*:* $\mathcal{C}$ checks whether $ID_i \stackrel{?}{=} ID_T$ and performs the following to generate the signature on m_i by ID_i:

- If $ID_i \neq ID_T$, then $\mathcal{C}$ knows the private key corresponding to ID_i, so $\mathcal{C}$ performs the signing algorithm to generate σ_i as per the protocol after querying $\mathcal{O}_{H_2}(m_i, ID_i)$ oracle.
- If $m_i \neq m^*$ and $ID_i = ID_T$, then $\mathcal{C}$ performs the following:
  - Makes an oracle query to the $\mathcal{O}_{H_2}$ oracle as $\mathcal{O}_{H_2}(m_i, ID_i)$. The result obtained from the oracle query is $h_i = er_i + r$.
  - Retrieves the corresponding r_i from the list L_2.
  - Computes $\sigma_i = c^{r_i}$ as the signature.

  Note that if $m_i = m^*$ and $ID_i = ID_T$, then it is an invalid sign oracle query.
- Sends σ_i as the signature on the message m_i by identity ID_i.

The verification of a signature is done by checking whether $\sigma_i^e \stackrel{?}{=} g_{i0}(g_{i1})^{h_i}$. We need to verify only the case when $ID_i \neq ID_T$ because in other cases, the signature is generated as per the protocol. The simulated signature generated by the above oracle passes the verification test as shown below:

$L.H.S = \sigma_i^e = c^{r_i e}$
$R.H.S = g_{i0}(g_{i1})^{h_i} = c^{-r} c^{er_i + r} = c^{r_i e}$

Since $L.H.S = R.H.S$, the simulated signature passes the verification test.

Forgery Phase: At the end of the ***Training Phase***, $\mathcal{F}$ produces a forged signature σ^* on the message m^* as if signed by the user with identity ID_T. $\mathcal{C}$ checks whether it is a valid signature and returns $a = \sigma^*$ and $b = e$ as the solution for the strong RSA problem.
Correctness: Below, we show that the a returned by $\mathcal{C}$ is a valid solution for the strong RSA problem, such that $c = a^b \ mod \ n$.

- The public keys corresponding to ID_T, i.e. $H_1(ID_T, 0)$ and $H_1(ID_T, 1)$ are set to be $\langle c^{-r}, c \rangle$ by $\mathcal{C}$ while $\mathcal{F}$ performed the $\mathcal{O}_{H_1}(ID_T, .)$ queries (We stress that c is taken from the strong RSA problem instance).
- Let d be such that $d \equiv e^{-1} \bmod \phi(n)$, where e is the master public key.

  Note: Now, in terms of the public key values and the value d, the private key values d_{T0} and d_{T1} are c^{-rd} and c^d respectively. However, these values cannot be computed explicitly by $\mathcal{C}$ as $\mathcal{C}$ has no way of computing d. That is why $\mathcal{C}$ has set " $-$ " for these unknowns in $\mathcal{O}_{H_1}$ oracle queries. Thus, the values $d_{T0} = c^{-rd}$ and $d_{T1} = c^d$ used in the proof are only hypothetical.
- Let $c^d = x$. Thus, $d_{T0} = x^{-r}$ and $d_{T1} = x$.

– Recall that $\mathcal{C}$ has set the h_i value corresponding to $\mathcal{O}_{H_2}(m^*, ID_T)$ query to be $r+1$.
– Now, $\sigma^* = d_{T0}(d_{T1})^{H_2(m^*, ID_T)} = x^{-r}x^{r+1} = x$ and $(\sigma^*)^e = x^e = (c^d)^e = c^{de} = c$.
– Thus, $\mathcal{C}$ have solved $c = a^b \ mod \ n$, with $a = \sigma^*$ and $b = e$.

Note: Here, the $e(= b)$ was selected by $\mathcal{C}$ and $\sigma^*(= a)$ is output by $\mathcal{F}$. Hence, the equation $c = a^b \ mod \ n$ can be solved for any odd b by choosing b as e of the public parameter and setting a to be σ^* produced by $\mathcal{F}$. □

Remark: The above proof is with respect to selective-identity and selective-message attacks. However, the unforgeability of the scheme can also be proved with respect to adaptive chosen message and identity attacks.

5 A Suite of Identity Based Aggregate Signature schemes from RSA (IBAS)

We propose the new identity based aggregate signature (IBAS) schemes in this section. Each scheme is followed by the proof for existential unforgeability.

Intuition behind the schemes: Our identity based aggregate signature schemes are motivated from Boneh et al.'s aggregate signature [8]. Their scheme is in the PKI settings and the signature is deterministic in nature. A deterministic signature does not have a randomization component and therefore, the signature becomes short. Gentry et al.'s [12] scheme was non-deterministic and uses random values for each signature. In order to establish a common random value, the signers have to participate in a communication round before generating an aggregate signature. If the signature scheme does not have random values, then the signature can be easily aggregated, which we learn from [8]. Thus, we tried to instantiate a deterministic identity based signature scheme and extend it to aggregate signature scheme to achieve full aggregation. Naturally, RSA based constructs suit well in the design of deterministic identity based signature scheme. We have constructed one in the preceding section.

5.1 Deterministic General IBAS (IBAS-I)

Our first IBAS scheme is a deterministic identity based aggregate signature scheme, which supports general aggregation, i.e. the order of the signer is not explicitly used for verification and hence, the order of signing cannot be traced. The scheme consists of six algorithms, out of which **Setup**, **KeyGen**, **Sign** and **Verify** are identical to that of D-IBS scheme. We explain the **AggregateSign** and **AggregateVerify** algorithms below:

– **AggregateSign:** This algorithm takes as input a set of t signatures $\{\sigma_i\}_{i=1\,to\,t}$ and the corresponding message identity pairs $\langle m_i, ID_i\rangle$, such that $\forall i = 1$ to t, $\langle \sigma_i \rangle$ is the valid signature by the user with identity ID_i on message m_i. The aggregation is done as follows:

$$\sigma_{agg} = \prod_{i=1}^{t} \sigma_i$$

The identity based aggregate signature is σ_{agg} and the corresponding list of message identity pairs is $\mathcal{L} = \{m_i, ID_i\}_{i=1\,to\,t}$.

– **AggregateVerify:** This algorithm takes the identity based aggregate signature σ_{agg} and the corresponding list of message identity pairs, $\mathcal{L} = \{m_i, ID_i\}_{i=1\,to\,t}$ and performs the verification as follows:
  - For all i=1 to t
    Compute $g_{i0} = H_1(ID_i, 0)$
    Compute $g_{i1} = H_1(ID_i, 1)$
    Compute $h'_i = H_2(m_i, ID_i)$
  - If $\sigma_{agg}^e \stackrel{?}{=} \prod_{i=1}^{t} (g_{i0}(g_{i1})^{h'_i})$, then outputs "$Valid$" else outputs "$Invalid$".

  The correctness of verification is straight forward.

Security Proof for IBAS-I:

In this section, we prove the security of our identity based aggregate signature scheme (IBAS-I). We show that if a polynomial time bounded forger exists, who can break our scheme with non-negligible advantage ϵ, then it is possible to construct an algorithm that solves the strong RSA problem with the same advantage.

Existential Unforgeability of IBAS-I:

Theorem 2. *Our identity based aggregate signature scheme (IBAS-I) is EUF-IBAS-I-CMA secure in the random oracle model under chosen-message and chosen-identity attack, if the strong RSA problem is intractable in $\mathbb{Z}_n^*$, where p, q, $(p-1)/2$ and $(q-1)/2$ are large prime numbers.*

Proof: The proof of this scheme is some what similar to that of the EUF-D-IBS-EUF proof. The major difference between the proofs of a deterministic identity based signature scheme and an identity based aggregate signature scheme is the aggregate signature oracle. Aggregate oracle is trivial in the case of general aggregation. The adversary obtains the individual signatures from the sign oracle and aggregates all the signature to form an aggregate signature. Below, we provide the sketch of the proof for IBAS-I scheme.

The EUF-IBAS-I-CMA game is an interactive game between the challenger $\mathcal{C}$ and a forger $\mathcal{F}$. The setup and training phases are same as that of the EUF-D-IBS-EUF proof. At the end of the training phase, the forger $\mathcal{F}$ generates the aggregate signature σ_{agg}^* for signatures $(\sigma_i)_{i=1\,to\,t}$ from the users $(ID_i)_{i=1\,to\,t}$ on messages $(m_i)_{i=1\,to\,t}$ where, at least one identity in the list of identities is the target identity, i.e. $ID_T \in \{ID_i\}_{i=1\,to\,t}$, for which the private key was not queried by $\mathcal{F}$ and the corresponding message is m^*.

It is to be noted that all the signatures except the signature corresponding to ID_T in the aggregate signature σ_{agg}^* can be generated by $\mathcal{C}$, since $\mathcal{C}$ knows the private keys corresponding to those identities. Thus, $\mathcal{C}$ generates all other signatures and divides them from σ_{agg}^*. The resulting value along with the value of master public key set by $\mathcal{C}$ will be the solution for the strong RSA problem similar to that of EUF-D-IBS-EUF proof. □

5.2 Deterministic Ordered Sequential IBAS (IBAS-II)

IBAS-II is a deterministic identity based aggregate signature scheme, which supports ordered sequential aggregation, i.e. the order of the signer is used explicitly for verification and hence, the order of signing could be traced. The scheme consists of six algorithms, out of which **Setup**, **KeyGen**, **Sign** and **Verify** are identical to that of D-IBS scheme. There is a slight modification in the sign and verify algorithms alone which we explain below, where as we present the full description of both **AggregateSign** and **AggregateVerify** algorithms:

- **Sign** and **Verify** are same as that of D-IBS with the modification in the parameters of the hash function $H_2(.)$, which takes the list $\mathcal{L}_i$ of signers who has participated in the aggregation process so far (i.e. $h = H_2(\mathcal{L}_i)$). Thus, we change the definition of this hash function to be $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_n^{odd}$.
- **AggregateSign:** The first user generates the signature σ_1 as per the *Sign* algorithm and passes this signature to the next signer, who aggregates his own signature to the received one and sends it as the aggregate signature to the next signer. This process continues until there are no signers left out. $\mathcal{L}_i$ is the list of message, identity pair of the users, which is upgraded on the fly, each time a new signature is aggregated. Initially, the list is empty ($\mathcal{L} = \langle\rangle$), when the first signer generates his signature, the list is appended with the message, identity pair of the first signer. The description of the sequential aggregate algorithm follows:

  When executed by the first signer this algorithm is same as that of the simple *Sign* algorithm. The output of the first signer will be $\sigma_{agg}^{(1)} = \sigma_1$ and $\mathcal{L}_1 = \langle(m_1, ID_1)\rangle$.

  All subsequent executions (i.e. the signer is not the first signer) are performed as follows. Let i be the index of the current signer:
  - The i^{th} signer updates the list as $\mathcal{L}_i = \mathcal{L}_{i-1} \cup (m_i, ID_i)$.
  - Computes his own signature σ_i on the message m_i (It should be noted that the hash computation of H_2 includes the list $\mathcal{L}_i$ in it, i.e. $H_2(\mathcal{L}_i)$).
  - Checks whether $\sigma_{agg}^{(i-1)}$ is a valid aggregate signature by executing the verification procedure below.
  - If it is valid, computes $\sigma_{agg}^{(i)} = \sigma_{agg}^{(i-1)}\sigma_i$.

If $i = t$ then the identity based aggregate signature σ_{agg} is $\sigma_{agg}^{(t)}$ and the corresponding list of message, identity pair is $\mathcal{L} = \langle (m_1, ID_1), \ldots, (m_t, ID_t) \rangle$.

- **AggregateVerify:** This algorithm takes the identity based aggregate signature σ_{agg} and the corresponding list of message, identity pairs, $\mathcal{L} = \langle (m_1, ID_1), \ldots, (m_t, ID_t) \rangle$ and performs the verification as follows:
  - For all i=1 to t
    Compute $g_{i0} = H_1(ID_i, 0)$
    Compute $g_{i1} = H_1(ID_i, 1)$
    Compute $h'_i = H_2(\mathcal{L}_i)$ (Where, $\mathcal{L}_i = \langle (m_1, ID_1), \ldots, (m_i, ID_i) \rangle$)
  - If $\sigma_{agg}^e \stackrel{?}{=} \prod_{i=1}^{t} (g_{i0}(g_{i1})^{h'_i})$ then outputs "$Valid$" else outputs "$Invalid$".

  The correctness of verification is straight forward.

Security Proof for IBAS-II:

In this section, we prove the security of our identity based aggregate signature scheme (IBAS-II). We show that if a polynomial time forger exists, who can break our scheme with non-negligible advantage ϵ, then it is possible to construct an algorithm which solves the strong RSA problem with the same advantage.

Existential Unforgeability of IBAS-II:

Theorem 3. *Our identity based aggregate signature scheme (IBAS-II) is EUF-IBAS-II-CMA secure in the random oracle model under chosen-message and chosen-identity attack, if the strong RSA problem is intractable in $\mathbb{Z}_n^*$, where p, q, $(p-1)/2$ and $(q-1)/2$ are large prime numbers.*

Proof: This proof is similar to that of EUF-IBAS-I-CMA. The forger $\mathcal{F}$ interacts with the challenger $\mathcal{C}$ as in the EUF-IBAS-II-CMA game. At the end of the interaction, $\mathcal{F}$ produces an aggregate signature σ_{agg}^*. This aggregate signature should be generated with the following constraints:

- σ_{agg}^* should be generated with a list $\mathcal{L}$ which contains (ID_T, m^*) as one of the entries.
- The individual signature corresponding to (ID_T, m^*) should not have been queried to the sign oracle.

If the aggregate signature is generated by $\mathcal{F}$, satisfying the above constraints, then $\mathcal{C}$ can divide the signatures corresponding to all other identities except ID_T and submit the resulting value and e, the master public key as the solution for the strong RSA problem, as in the previous proof. □

5.3 Non-Deterministic General IBAS (IBAS-III)

The third scheme IBAS-III is a non-deterministic identity based aggregate signature scheme, with general aggregation. The scheme consists of six algorithms, out of which **Setup**, **KeyGen**, **Sign** and **Verify** are identical to that of IBAS-II scheme. The scheme is non-deterministic in nature, i.e. different signatures are generated for the same message when the sign algorithm is invoked more than once. Randomization is achieved by applying the technique in [7], where a random value is added to the signature in such a way that it cannot be altered and the random value is send along with the signature. The functionality of IBAS-III is similar to that of Gentry et al.'s [12] scheme, where the first user chooses a random value and this value is propagated to the consequent signers. It should be noted that the order of signers is not preserved in this signature scheme. Yet, the aggregation will be efficient if it is performed sequentially, because the random value should be propagated to each and every user before signing. As mentioned earlier, Gentry et al.'s [12] scheme is not secure if the same random value is used again for signing two different messages. This weakness is not observed in our scheme, making it superior that [12]. We describe the sequential aggregation here, but it should be noted that the sequential operation can be overcome if the first signer broadcasts the random value to all the signers in advance or if the random value is generated by hashing a common session identifier among the signers. There is a slight modification in both **AggregateSign** and **AggregateVerify** algorithms which we describe below:

- **Sign** and **Verify** are same as that of D-IBS. It should be noted that the parameters of the hash function H_2, includes the random value chosen by the first signer in the aggregation process (i.e. $h = H_2(m_i, ID_i, r)$). Therefore, we change the definition of this hash function to be $H_2 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^{odd}$.

– **AggregateSign:** When executed by the first signer, this algorithm generates the signature as follows:
  - Chooses $r \in_R \mathbb{Z}_n^*$.
  - Computes $h_1 = H_2(m_1, ID_1, r)$.
  - Computes $\sigma_{agg}^{(1)} = d_{10}(d_{11})^{h_1}$.
  - The aggregate signature send by the first signer is $\sigma_{agg} = \langle r, \sigma_{agg}^{(1)} \rangle$ and the list $\mathcal{L}_1 = \langle m_1, ID_1 \rangle$.

  All subsequent executions (i.e. the current signer is not the first signer) are performed as follows:
  - The i^{th} signer computes his own signature σ_i on the message m_i (Signature is generated according to the standard signing procedure described for D-IBS. It should be noted that the computation of H_2 hash function includes the random value chosen by the first signer in it, i.e. $H_2(m_i, ID_i, r)$).
  - Checks whether $\sigma_{agg}^{(i-1)}$ is a valid aggregate signature by executing the verification procedure mentioned below.
  - If it is valid, computes $\sigma_{agg}^{(i)} = \sigma_{agg}^{(i-1)}\sigma_i$.
  - Updates the list as $\mathcal{L}_i = \mathcal{L}_{i-1} \cup (m_i, ID_i)$

  If $i = t$ then the identity based aggregate signature is $\sigma_{agg} = \langle r, \sigma_{agg}^{(t)} \rangle$ and the corresponding list of message, identity pair is $\mathcal{L} = \langle (m_1, ID_1), \ldots, (m_t, ID_t) \rangle$.
– **AggregateVerify:** This algorithm takes the identity based aggregate signature $\sigma_{agg} = \langle r, \sigma_{agg}^{(t)} \rangle$ and performs the verification as follows:
  - For all i=1 to t
    Compute $g_{i0} = H_1(ID_i, 0)$
    Compute $g_{i1} = H_1(ID_i, 1)$
    Compute $h_i' = H_2(m_i, ID_i, r)$
  - If $(\sigma_{agg}^{(t)})^e \stackrel{?}{=} \prod_{i=1}^{t} (g_{i0}(g_{i1})^{h_i'})$, then outputs "$Valid$" else outputs "$Invalid$".

  Correctness of verification:

$$L.H.S = (\sigma_{agg}^{(t)})^e = (\prod_{i=1}^{t} d_{i0}(d_{i1})^{h_i'})^e = (\prod_{i=1}^{t} g_{i0}^d (g_{i1}^d)^{h_i'})^e = \prod_{i=1}^{t} (g_{i0}(g_{i1})^{h_i'}) = R.H.S$$

Security Proof for IBAS-III:

The proof of existential unforgeability of the IBAS-III follows from the proof of IBAS-I. At the end of the interaction, $\mathcal{F}$ produces an aggregate signature σ_{agg}^*. This aggregate signature should be generated with the following constraints:

– σ_{agg}^* should be generated with a list $\mathcal{L}$ which contains (ID_T, m^*) as one of the entries.
– The individual signature corresponding to (ID_T, m^*) should not have been queried to the sign oracle with the random value r as input to the $\mathcal{O}_{H_2}$ oracle.

If the aggregate signature is generated by $\mathcal{F}$, satisfying the above constraints, then $\mathcal{C}$ can divide the signatures corresponding to all other identities except ID_T and submit the resulting value and e, the master public key as the solution for the strong RSA problem, as in the previous proofs. □

5.4 Non-Deterministic Ordered Sequential IBAS (IBAS-IV)

The fourth scheme IBAS-IV is a non-deterministic identity based aggregate signature scheme, with ordered sequential aggregation that maintains order. The modified **AggregateSign** and **AggregateVerify** algorithms are describe below:

– **Sign** and **Verify** are same as that of D-IBS but the parameters of the hash function H_2, includes the random value chosen by the first signer in the aggregation process (i.e. $h = H_2(\mathcal{L}_i, r)$). Therefore, we change the definition of this hash function to be $H_2 : \{0,1\}^* \times \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^{odd}$.
– **AggregateSign:** Initially the list $\mathcal{L}$ is empty ($\mathcal{L} = \langle\rangle$). When executed by the first signer, this algorithm generates the signature as follows:
  - Chooses $r \in_R \mathbb{Z}_n^*$.
  - Updates the list $\mathcal{L}_1 = \langle m_1, ID_1 \rangle$
  - Computes $h_1 = H_2(r, \mathcal{L}_1)$.

- Computes $\sigma_{agg}^{(1)} = d_{10}(d_{11})^{h_1}$.
- The aggregate signature send by the first user $\sigma_{agg} = \langle r, \sigma_{agg}^{(1)} \rangle$ and $\mathcal{L}_1 = \langle (m_1, ID_1) \rangle$.

All subsequent executions are performed as follows:

- The i^{th} signer updates the list as $\mathcal{L}_i = \mathcal{L}_{i-1} \cup (m_i, ID_i)$.
- Computes his own signature $\sigma_{agg}^{(i)}$ on the message m_i (Signature is generated according to the standard signing procedure described above).
- Checks whether $\sigma_{agg}^{(i-1)}$ is a valid aggregate signature by executing the verification procedure below.
- Computes $\sigma_{agg}^{(i)} = \sigma_{agg}^{(i-1)} \sigma_i$.

If $i = t$ then the identity based aggregate signature is $\sigma_{agg} = \langle r, \sigma_{agg}^{(t)} \rangle$.

– **AggregateVerify:** This algorithm takes the identity based aggregate signature $\sigma_{agg} = \langle r, \sigma_{agg}^{t} \rangle$, the corresponding list of message, identity pairs, $\mathcal{L} = \langle (m_1, ID_1), \ldots, (m_t, ID_t) \rangle$ and performs the verification as follows:

- For all i=1 to t
  Compute $g_{i0} = H_1(ID_i, 0)$
  Compute $g_{i1} = H_1(ID_i, 1)$
  Compute $h_i' = H_2(r, \mathcal{L}_i)$ (Where, $\mathcal{L}_i = \langle (m_1, ID_1), \ldots, (m_i, ID_i) \rangle$)
- If $(\sigma_{agg}^{(t)})^e \stackrel{?}{=} \prod_{i=1}^{t} (g_{i0}(g_{i1})^{h_i'})$ then outputs "$Valid$" else outputs "$Invalid$".

The correctness of verification is straight forward and is omitted.

Security Proof for IBAS-IV:

The proof of existential unforgeability of the IBAS-IV follows from the proof of IBAS-II. At the end of the interaction, $\mathcal{F}$ produces an aggregate signature σ_{agg}^*. This aggregate signature should be generated with the following constraints:

– σ_{agg}^* should be generated with a list $\mathcal{L}$ which contains (ID_T, m^*) as one of the entries.
– The individual signature corresponding to (ID_T, m^*) should not have been queried to the sign oracle with the random value r as input to the $\mathcal{O}_{H_2}$ oracle.

If the aggregate signature is generated by $\mathcal{F}$, satisfying the above constraints, then $\mathcal{C}$ can divide the signatures corresponding to all other identities except ID_T and submit the resulting value and e, the master public key as the solution for the strong RSA problem, as in the previous proofs. □

6 An Identity Based Multi-Signature scheme from RSA (IBMS)

We propose the new identity based multi-signature (IBMS) schemes in this section.

Intuition behind the scheme: Our basic identity based signature scheme D-IBS is deterministic. A deterministic signature does not have a randomization component and therefore, the signature becomes short and hence to construct a multi-signature scheme, it is not required to establish a common random value. This avoids the communication round before generating an multi-signature. Thus, our IBMS do not have any communication round, where as the schemes in [3] and [1] have 3 and 2 rounds respectively. The IBMS scheme in [12] is a one round protocol but it depends on bilinear pairing, which are still more expensive than RSA exponentiation.

6.1 Deterministic IBMS

Our IBMS scheme is a deterministic scheme. The scheme consists of six algorithms, out of which **Setup**, **KeyGen**, **Sign** and **Verify** are identical to that of D-IBS scheme. We explain the **Multi-Sign** and **Multi-Verify** algorithms below:

– **Multi-Sign:** After fixing the identities participating in the protocol, the list $\mathcal{L} = \langle ID_i \rangle_{i=1\,to\,t}$ is generated. Each signer now computes the individual signature as follows:

- Computes $h = H_2(m, \mathcal{L})$.
- Computes $\sigma = d_{i0}(d_{i1})^h$.

Now, the aggregator collects the set of t signatures $\{\sigma_i\}_{i=1\,to\,t}$, the corresponding message m, the list of identities $\mathcal{L} = \langle ID_i \rangle$, checks whether the signatures $\langle \sigma_i \rangle$ is valid $\forall\, i = 1$ to t and aggregates the signatures as follows:

$$\sigma_{agg} = \prod_{i=1}^{t} \sigma_i$$

The identity based aggregate signature is σ_{agg} and the corresponding list of message identity pairs is $\mathcal{L} = \{m_i, ID_i\}_{i=1\,to\,t}$.

– **Multi-Verify:** This algorithm takes the identity based aggregate signature σ_{agg}, the corresponding message m and the list of identities $\mathcal{L} = \{ID_i\}_{i=1\,to\,t}$ and performs the verification as follows:

- For all i=1 to t

  Compute $g_{i0} = H_1(ID_i, 0)$
  Compute $g_{i1} = H_1(ID_i, 1)$

- Compute $h'_i = H_2(m, \mathcal{L})$
- If $\sigma_{agg}^e \stackrel{?}{=} \prod_{i=1}^{t} (g_{i0}(g_{i1})^{h'_i})$, then outputs "$Valid$" else outputs "$Invalid$".

The correctness of verification is straight forward.

7 Conclusion

In this paper, we have proposed the first deterministic identity based signature scheme, whose security is based on RSA. We have proposed a suite of four identity based aggregate signature schemes with different desirable properties. All the schemes proposed in this paper achieve full aggregation and thus the results are attractive. Our first construct, IBAS-I addressed the open problem posed by Hwang et al. in [15], which is to design an identity based aggregate signature scheme where the signers need not have to agree on a common random value. IBAS-II provides sequential aggregation where the order of the signers is preserved and finds application in tracing the route of packets in routing. Gentry et al.'s identity based aggregate signature scheme in [12] had the weakness where in if a signer tactfully re-uses the random value used for signing a message, a total break of the scheme is possible. Our third scheme IBAS-III is a non-deterministic aggregate signature scheme, which does not show any weakness of [12] even if the signer uses the same random value several times. Finally, our fourth scheme IBAS-IV is a non-deterministic scheme which supports sequential aggregation. We have formally proved the security of our schemes in the random oracle model. We have also proposed an identity based multi-signature scheme, which is more efficient when compared to the IBMS schemes based on RSA presented in [3] and [1], where both these scheme require prior communication between the signers and ours does not. The scheme in [12] is based on bilinear pairing and thus ours turns out to be more efficient than all the three existing schemes.

The following table summarizes the current state of the art in the research of identity based aggregate signatures which achieves full aggregation. The table includes the communication and computational complexity of our new schemes IBAS-I, IBAS-II, IBAS-III and IBAS-IV along with the existing schemes. In the following table the column entries marked with †, represents that the issue is discussed in our paper.

Property ..... Schemes	Type of Sys (PKI/IB)	Security (SO/RO)	Security Status (PS /NK)	Agg Sign Len	Sign Cost (/ user)	Agg Verify (t users)	Hard Prob	Sign Type (D/ND)	Rounds	Agg Mode (G/OS /S)
[11]	IB	RO	PS	$2\|\mathbb{G}\|$	3[M]	2[P]+t[M]	CDH	ND	2	G
[1]	IB	RO	PS	$2\|\mathbb{Z}_n^*\| + \|\kappa\| + \|log(l)\|$	2[E]	t[E]	RSA	ND	2	G
[5]	IB	RO	PS	$3\|\mathbb{G}\|$	7[M]	6[P]+t[M]	CDH-type	ND	-	S
[12]	IB	RO	PS	$2\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$	5[M]	3[P]+t[M]	GAP-DH	ND	-	S
IBAS-I	IB	RO	PS	$\|\mathbb{Z}_n^*\|$	1[E]	$(t+1)$[E]	RSA	D	1	G
IBAS-II	IB	RO	PS	$\|\mathbb{Z}_n^*\|$	1[E]	$(t+1)$[E]	RSA	D	-	OS
IBAS-III	IB	RO	PS	$2\|\mathbb{Z}_n^*\|$	1[E]	$(t+1)$[E]	RSA	ND	1	G
IBAS-IV	IB	RO	PS	$2\|\mathbb{Z}_n^*\|$	1[E]	$(t+1)$[E]	RSA	ND	-	OS
[16]	PKI	SO	NK†	$2\|\mathbb{G}\|$	4[M]+ ($t.k$+5) [A]	2[P]+ $t.k$[A]+ t[$\mathbb{G}_T$M]	WSS	ND	-	S

Table-1: State of the art survey of IBAS schemes

The following table summarizes the current state of the art in the research of identity based multi-signatures.

Property ..... Schemes	Type of Sys (PKI/IB)	Security (SO/RO)	Security Status (PS /NK)	Agg Sign Len	Sign Cost (/ user)	Agg Verify (t users)	Hard Prob	Sign Type (D/ND)	Rounds
[12]	IB	RO	PS	$2\|\mathbb{G}\|$	3[M]	3[P]	GAP-DH	ND	1
[3]	IB	RO	PS	$\|\mathbb{Z}_n^*\| + \kappa$	1[E]	2[E]	RSA	ND	3
[1]	IB	RO	PS	$2\|\mathbb{Z}_n^*\|+\|\kappa\|$	4[E]	5[E]	RSA	ND	2
IBMS	IB	RO	PS	$\|\mathbb{Z}_n^*\|$	1[E]	2[E]	RSA	D	1

Table-2: State of the art survey of IBMS schemes

The terms used in both the above tables are explained here. IB- Identity Based System, PKI PKI Based System, SO- Standard Model, RO- Random Oracle Model, PS- Provably Secure, NK- Status Not Known, ND - Nondeterministic Signature, D- Deterministic Signature, G- General aggregation, OS- Ordered Sequential Signature, S- Sequential Signature, κ is the security parameter of the scheme, [E]-Exponentiation in $\mathbb{Z}_n^*$, [M]-Scalar Point Multiplication in $\mathbb{G}$, [P]- Bilinear Pairing Operation, Gap-DH- Gap-Diffie-Hellman, [$\mathbb{G}_T$M]-Exponentiation in $\mathbb{G}_T$ and WSS- Waters Signature Scheme.

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. Mihir Bellare and Gregory Neven. Identity-based multi-signatures from rsa. In *Topics in Cryptology - CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 145–162. Springer, 2006.
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. In *ACM Conference on Computer and Communications Security, CCS 2007*, pages 276–285. ACM, 2007.
5. 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/.
6. 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.

7. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, vol.21(no.2):149–177, 2008.
8. 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.
9. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, vol.17(no.4):297–319, 2004.
10. Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In *Public Key Cryptography - PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2002.
11. 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.
12. 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.
13. Javier Herranz. Deterministic identity-based signatures for partial aggregation. *The Computer Journal*, vol-49(no-3):322–330, 2006.
14. Florian Hess. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2003.
15. 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.
16. 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.
17. Di Ma. Practical forward secure sequential aggregate signatures. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008*, pages 341–352. ACM, 2008.
18. 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.
19. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO - s4*, Lecture Notes in Computer Science, pages 47–53. Springer, 1984.
20. 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.
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.
22. Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In *Public Key Cryptography - PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 277–290. Springer, 2004.

Appendix:

A Aggregate Signatures in Standard Model

A.1 Sequential Aggregate Signature Scheme by Steve Lu et al. in Standard Model

In Eurocrypt 2006, Steve Lu et al proposed the first PKI based sequential aggregate signature scheme in the standard model. According to the definition given in [16], the sequential aggregate signature scheme proposed by Steve Lu et al. consists of three algorithms. The first algorithm **Kg**, is used to generate the public-private key pairs. The second, **ASig**, takes not only a private key and a message to sign, as does an ordinary signing algorithm, but also an aggregate-so-far by a set l signers on l corresponding messages; it extends the given aggregate-so-far signature by l signers into a new aggregate signature as if signed by the $(l+1)$ signers on $(l+1)$ corresponding messages. The third algorithm, **AVf**, takes a purported aggregate signature, along with l public keys and l corresponding messages, and decides whether the aggregate signature is valid or not.

In this section, we review the security model and the scheme of Lu et al. [16]. We also demonstrate the weakness in the security proof of their scheme.

Security Model for Sequential Aggregate Signature Scheme(Certified-Key model) In this section, we recall the security model as defined by Steve Lu et al. in [16]. The model is restricted in the sense that, the adversary can query for the aggregate involving the challenge key by submitting a aggregate-so-far signature with the constraint that, the public key used for generation of aggregate-so-far signature or forgery should be a certified public key(i.e., adversary should submit the (pk, sk) to challenger or secret key sk is known to challenger). The goal of the adversary is to generate a sequential aggregate signature that frames the challenge user. The adversary is allowed to choose all the keys in the forged sequential aggregate signature but not the challenge key.
Formally, the advantage of the forger $\mathcal{F}$ in their model is the probability that the challenger $\mathcal{C}$ outputs 1 in the following game :

Setup : *Initialize the list of certified public keys* $C \leftarrow \emptyset$. Choose (pk^*, sk^*) $\xleftarrow{R}$ **Kg**. Run $\mathcal{F}$ with pk^* as input.

Certification Queries : *Algorithm* $\mathcal{F}$ *provides a key pair* (pk', sk') *in order to certify* pk'. *Add* pk' *to* C *if* sk' *is its matching private key.*

Signing Queries : *Algorithm* $\mathcal{F}$ *requests a sequential aggregate signature, under the challenge key* pk^*, *on a message* M. *In addition, it also provides an aggregate-so-far signature* σ' *on messages* $\mathcal{M}' = \{M_1, \ldots, M_k\}$ *under public keys* $PK' = \{pk_1{}', \ldots, pk_k{}'\}$. $\mathcal{C}$ *generates and output the aggregate signature* σ *along with* $\mathcal{M} = \mathcal{M}' \cup M$ *and* $PK = PK' \cup pk^*$, *if the input provided by* $\mathcal{F}$ *satisfies the following constraints :*
- σ' *is a valid aggregate-so-far signature with respect to* $\mathcal{M}'$, PK'.
- $pk^* \cap PK' = \emptyset$.
- None of the $pk_i{}' \in PK'$ appear more than once in PK'.

Otherwise, output "*Invalid*".

Output : *Eventually,* $\mathcal{F}$ *halts by producing a forgery* σ^* *on messages* $\mathcal{M}$ *under* PK^*. *Output* 1 *if the following condition holds :*
- $pk^* \in PK^*$
- $PK^* \setminus \{pk^*\} \subseteq C$ *and* $|PK^*| \leq n$.
- *Without loss of generality we assume that* pk^* *appears at position* 1 *and* $\mathcal{M}[1]$ *must not have been queried by* $\mathcal{F}$ *to the sequential aggregate sign oracle.*
- $pk_i \in PK^*$ *should not appear more than once in* PK^*.

Otherwise, output 0.

The sequential aggregate signature scheme is (t, $q_{Certify}$, q_{sign}, n, ϵ) secure if no $t - time$ adversary making $q_{Certify}$ certification queries and q_{sign} signing queries can win the above game with advantage more than ϵ, where n is upper bound on the length of the sequential signatures involved.

Scheme The sequential aggregate signature(WSA) by Steve Lu et al. is based on the Waters signature scheme(WSS). The aggregate signature scheme achieves full aggregation and the aggregate signature consists of two group elements $S_1{}', S_2{}'$. Here S_2' is the "shared random value". The main idea behind the construct is, when a user wants to add his signature to a aggregate-so-far signature will add his signature and will re-randomize it. For this the user should know the discrete log values of all his public keys. The formal description of the scheme is given below :

WSA.Kg :

- *Pick* α, $\bar{y} \in_R \mathbb{Z}_p$.
- *Pick* $\hat{y} = (y_1, y_2, \ldots, y_k) \in_R \mathbb{Z}_p^k$.
- *Compute,*
  $\bar{u} = g^{\bar{y}}$.
  $\hat{u} = (u_1, u_2, \ldots, u_k) \leftarrow (g^{y_1}, g^{y_2}, \ldots, g^{y_k})$.
  $A = \hat{e}(g, g)^{\alpha}$.
- *User's public key is* $pk = (A, \bar{u}, \hat{u})$.
- *User's private key is* $sk = (\alpha, \bar{y}, \hat{y})$.

WSA.Asig(*sk*, *pk* $M = (m_1, \ldots, m_k)$, $\sigma' = (S_1', S_2')$, $\mathcal{M} = \{M_1, M_2, \ldots, M_l\}$, $PK = \{pk_1, pk_2, \ldots, pk_l\}$***) :*** *Parse* $\mathcal{M}[i] = M_i = (m_{i,1}, m_{i,2}, \ldots, m_{i,k})$ *and* $PK[i] = pk_[i] = (A_i, \bar{u}_i, \hat{u}_i)$. *If* ***AVf****(*$\sigma'$, $\mathcal{M}$, PK*) returns "*$Invalid$*", Output* $FAIL$. *Else, Compute the following :*

- $\omega_1 \leftarrow S_1' g^{\alpha} (S_2')^{\bar{y} + \sum_{j=1}^{k} (y_j m_j)}$. (1)
- $\omega_2 \leftarrow S_2'$.
- *Pick* $\tilde{r} \in_R \mathbb{Z}_p$.
- $S_1 \leftarrow \omega_1 \, (\bar{u} \prod_{j=1}^{k} {u_j}^{m_j})^{\tilde{r}} \prod_{i=1}^{l} (\bar{u}_i \prod_{j=1}^{k} {u_{i,j}}^{m_{i,j}})^{\tilde{r}}$. (2)
- $S_2 \leftarrow \omega_2 g^{\tilde{r}}$. (3)
- Update $\mathcal{M}$ by adding M and update PK by adding pk.
- Output $\sigma = (S_1, S_2)$.

WSA.AVf(*$\sigma, \mathcal{M}, PK$****) :*** *Input is a purported sequential signature. Output "*$Valid$*" if the following are satisfied :*

- $\hat{e}(S_1, g)\hat{e}(S_2, \prod_{i=1}^{l} (\bar{u}_i \prod_{j=1}^{k} {u_{i,j}}^{m_{i,j}})) = \prod_{i=1}^{l} A_i$. (4)
- $|PK| = |\mathcal{M}|$.
- All $pk_i \in PK$ appears only once and are certified.

If any one of the above constraint is not satisfied, output "*Invalid*"

Weakness in the Security Proof The security of the sequential aggregate signature scheme in [16] is reduced to the security of the underlying Waters signature scheme(WSS)[]. It is shown that if an adversary $\mathcal{F}$ succeeds in generating a forgery σ^* by playing the game proposed in the security, then the challenger $\mathcal{C}$ can derive a valid WSS signature. Here, in this section we are going to analyze the property exhibited by the aggregate-so-far signature σ' by l signers on corresponding l messages with the aggregate generated σ generated by augmenting the $(l+1)^{th}$ user's signature on the $(l+1)^{th}$ message to the signature σ'. By using that property we show that the security reduction given for [16](reduced to the security of WSS) does not guarantee the relationship: the simulation of aggregate sign oracle is not capturing the real scenario(that can be easily identified by the adversary $\mathcal{F}$). Thereby, we show that the simulation is wrong and the adversary almost always abort during the game. Also, we demonstrate that fixing this weakness in the proof, leads to the assumption that challenger $\mathcal{C}$ himself is solving some hard problem. For doing this first we establish the property that prevails in the aggregate signature in *Lemma* 1.

Closer Look at the Property of Aggregate signature in [16] :

Lemma 1. *Suppose* $\sigma' = (S_1', S_2')$ *be a valid aggregate-so-far signature with respect to* $\mathcal{M} = \{M_i\}_{i=1\,to\,t}, PK = \{pk_i\}_{i=1\,to\,t}$, *where* $S_1' = \prod_{i=1}^{t} g^{\alpha_i} (\prod_{i=1}^{t} (\bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}}))^r$ *and* $S_2' = g^r$, *and let* $\sigma = (S_1, S_2)$ *be the valid aggregate signature generated by the* $WSA.Asig(sk_{\hat{t}} = (\alpha_{\hat{t}}, \bar{y}_{\hat{t}}, \hat{y}_{\hat{t}} = (y_{\hat{t},1}, \ldots, y_{\hat{t},k})), pk_{\hat{t}} = (A_{\hat{t}}, \bar{u}_{\hat{t}}, u_{\hat{t}}), M_{\hat{t}}, \sigma', \mathcal{M}, PK)$ *protocol, where* $S_1 = \prod_{i=1}^{\hat{t}} g^{\alpha_i} (\prod_{i=1}^{\hat{t}} (\bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}}))^{r+\tilde{r}}$ *and* $S_2 = g^{r+\tilde{r}}$. *Then,* $\hat{\sigma} = (\hat{S}_1, S_2)$ *derived from* σ' *and* σ *will be a valid signature of* $M_{\hat{t}}$ *with respect to* $pk_{\hat{t}}$, *where* $\hat{S}_1 = (S_1)(S_1')^{-1} \left(\prod_{i=1}^{t} X^{\bar{y}_i} \prod_{j=1}^{k} (X^{m_{i,j}})^{y_{i,j}} \right)^{-1}$ *(Here* $X = S_2(S_2')^{-1} = g^{\tilde{r}}$*)*

Proof : The component $\hat{S}_1$ of σ^* is equal to $g^{\alpha_{\hat{t}}} (\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j})^{r+\tilde{r}}$.

Correctness : First we show that $\hat{S}_1 = g^{\alpha_{\hat{t}}} (\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j})^{r+\tilde{r}}$:

$$
\begin{aligned}
\text{LHS} = \hat{S}_1 &= (S_1)(S_1')^{-1} \left(\prod_{i=1}^{t} X^{\bar{y}_i} \prod_{j=1}^{k} (X^{m_{i,j}})^{y_{i,j}} \right)^{-1} \\
&= (S_1)(S_1')^{-1} \left((\prod_{i=1}^{t} g^{\bar{y}_i})^{\tilde{r}} \prod_{j=1}^{k} (g^{m_{i,j} y_{i,j}})^{\tilde{r}} \right)^{-1} \\
&= \left(S_1' g^{\alpha_{\hat{t}}} \left(\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{i,j}^{m_{\hat{t},j}} \right)^{r} \left(\prod_{i=1}^{\hat{t}} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{\tilde{r}} \right) (S_1')^{-1} \left(\prod_{i=1}^{t} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{-\tilde{r}} \\
&= \left(g^{\alpha_{\hat{t}}} \left(\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j}^{m_{\hat{t},j}} \right)^{r} \left(\prod_{i=1}^{\hat{t}} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{\tilde{r}} \right) \left((\prod_{i=1}^{t} \bar{u}_i)^{\tilde{r}} \prod_{j=1}^{k} (u_{i,j}^{m_{i,j}})^{\tilde{r}} \right)^{-1} \\
&= \left(g^{\alpha_{\hat{t}}} \left(\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j}^{m_{\hat{t},j}} \right)^{r} \left(\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j}^{m_{\hat{t},j}} \right)^{\tilde{r}} \left(\prod_{i=1}^{t} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{\tilde{r}} \right) \left(\prod_{i=1}^{t} \bar{u}_i) \prod_{j=1}^{k} (u_{i,j}^{m_{i,j}} \right)^{-\tilde{r}} \\
&= g^{\alpha_{\hat{t}}} \left(\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j}^{m_{\hat{t},j}} \right)^{r+\tilde{r}} \left(\prod_{i=1}^{t} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{\tilde{r}} \left(\prod_{i=1}^{t} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{-\tilde{r}} \\
&= g^{\alpha_{\hat{t}}} \left(\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j}^{m_{\hat{t},j}} \right)^{r+\tilde{r}} \\
&= RHS
\end{aligned}
$$

Next, is to that $\hat{\sigma} = (\hat{S}_1, S_2)$ is a valid signature with respect to $pk_{\hat{t}}$ and message $M_{\hat{t}}$. For this, We shall now verify the validity of the test given in **equation (4)** of **WSA.AVf** algorithm for $\hat{\sigma}$.

$$
\begin{aligned}
LHS \text{ of } \textbf{equation (4)} &= \hat{e}(\hat{S}_1, g) \hat{e}(S_2, \bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j}^{m_{\hat{t},j}})^{-1} \\
&= \hat{e}(g^{\alpha_{\hat{t}}} (\bar{u}_{\hat{t}} \prod_{j=1}^{k} u_{\hat{t},j})^{r+\tilde{r}}, g) \hat{e}(g^{r+\tilde{r}}, \bar{u}_t \prod_{j=1}^{k} u_{t,j}^{m_t,j})^{-1} \\
&= \hat{e}(g^{\alpha_{\hat{t}}}, g) \\
&= \hat{e}(g, g)^{\alpha_{\hat{t}}}
\end{aligned}
$$

$= A_{\hat{t}} = RHS$ of **equation (4)**.

□

Remark 1: *This lemma provides a mechanism for adversary to test if the last aggregation σ is done correctly with respect to the previous aggregation σ'. Specifically, when adversary $\mathcal{F}$ gives σ', PK, M to challenger $\mathcal{C}$ requesting for the signature on $M_{\hat{t}}$ and $\mathcal{C}$ gives σ back to $\mathcal{F}$, $\mathcal{F}$ can check if σ is properly constructed with respect to σ', message $M_{\hat{t}}$ and last user $pk_{\hat{t}}$. This check can be done by constructing $\hat{\sigma}$ and applying $WSA.AVf(\hat{\sigma}, M_{\hat{t}}, pk_{\hat{t}})$. Note that construction of $\hat{\sigma}$ requires the knowledge of all the private keys of all the users involved in the construction of σ'(i.e., all sk_i corresponding to certified $pk_i \in PK$ used for generation of σ'), but that is trivially available for $\mathcal{F}$ as per the security model.*

Description of Sequential Aggregate Sign Oracle in [16] :
Let $pk_{\hat{t}}$ be the challenge key. The sequential aggregate sign oracle demonstrated in the security proof of [16] takes a valid aggregate-so-far signature σ' for $\mathcal{M} = \{M_i\}_{i=1\,to\,t}$ and $PK = \{pk_i\}_{i=1\,to\,t}$, a message $M_{\hat{t}}$ as input. Ignores the aggregate-so-far signature σ' and generates a fresh sequential signature by performing the following :

- Obtains the signature for $(M_{\hat{t}}, pk_{\hat{t}})$ by querying the Waters signature algorithm($\sigma_{\hat{t}} = WSS.Sign(M_{\hat{t}}, pk_{\hat{t}})$
- Set $\sigma_0 = \sigma_{\hat{t}}$, $PK_0 = \{pk_{\hat{t}}\}$ and $\mathcal{M}_0 = M_{\hat{t}}$
- For $i = 1$ to t do
  - $\sigma_i = WSA.Asig(sk_i, pk_i, M_i, \sigma_{i-1}, \mathcal{M}_{i-1}, PK_{i-1})$
  - $PK_i = PK_{i-1} \cup pk_i$.
  - $\mathcal{M}_i = \mathcal{M}_{i-1} \cup M_i$.
- $\mathcal{C}$ outputs $\sigma = \sigma_t$ as the sequential aggregate generated from σ' with respect $\mathcal{M} \cup M_{\hat{t}}$ and $PK \cup pk_{\hat{t}}$.

Recall from the **Remark 1** of **Lemma 1** given above, that adversary can now construct $\hat{\sigma}$ from σ' and σ and checks whether $WSA.AVf(\hat{\sigma}, M_{\hat{t}}, pk_{\hat{t}})$ returns "$Valid$". As $\hat{\sigma}$ is not properly or correctly constructed from σ' by $\mathcal{C}$, $\hat{\sigma}$ will fail to pass the test in **equation (4)** of $WSA.Avf$ and expose the challenger. Thus, the adversary $\mathcal{F}$ would abort the security game rather than proceeding further. This clearly shows that the adversary $\mathcal{F}$ is almost always aborted. Note that σ sent by $\mathcal{C}$ to $\mathcal{F}$ would pass the aggregate verify test. But still $\mathcal{F}$ can determine that σ is not properly formed with respect to σ' by using **Lemma 1**

Impossibility of Fixing the Above Flaw : If one tries to fix the issue and reduce the security of WSA scheme to the underlying Waters signature scheme (WSS) by providing the aggregate sign oracle for WSA, then it is equivalent to assuming that the challenger himself is solving the CDHP(which cannot be TRUE for a polynomial algorithm $\mathcal{C}$)

claim 1 :*Let us assume that the challenger $\mathcal{C}$ simulates the sequential aggregate signature oracle successfully such that the input and output to that oracle satisfy Lemma 1, then $\mathcal{C}$ is solving $CDHP$.*

Proof : Note that, had we worked with properly constructed aggregate signature, $\langle \hat{\sigma} = (S_1, S_2), \mathcal{M} = \{M_1, \ldots M_{\hat{t}}\}, PK = \{pk_1, \ldots, pk_{\hat{t}}\}\rangle$, where $S_1 = \prod_{i=1}^{\hat{t}} g^{\alpha_t} \left(\prod_{i=1}^{\hat{t}} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{r+\tilde{r}}$ and $S_2 = g^{r+\tilde{r}}$, generated from $\langle \sigma' = (S_1', S_2'), \mathcal{M}' = \{M_1, \ldots M_t\} PK' = \{pk_1, \ldots, pk_t\}\rangle$, where $S_1' = \prod_{i=1}^{t} g^{\alpha_t} \left(\prod_{i=1}^{t} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{r}$ and $S_2' = g^r$. That means $\hat{\sigma} = (\hat{S}_1, \hat{S}_2 = S_2)$(generated

as per **Lemma 1**) is a valid signature on $M_{\hat{t}}$ using $pk_{\hat{t}}$. For achieving, the challenger must know r used for the generation S_2'(generated by $\mathcal{F}$) or $sk_{\hat{t}}$. Unfortunately, $\mathcal{C}$ is not aware of both r and $sk_{\hat{t}}t$. In other words, without knowing r and $sk_{\hat{t}}$ but using only $S_2' = g^r$ and $pk_{\hat{t}}$ and the values (S_1, S_2) are computed, where $S_1 = \prod_{i=1}^{\hat{t}} g^{\alpha_i} \left(\prod_{i=1}^{\hat{t}} \bar{u}_i \prod_{j=1}^{k} u_{i,j}^{m_{i,j}} \right)^{r+\tilde{r}}$ and $S_2 \leftarrow \omega_2 g^{\tilde{r}}$. (Performing **equation (1),(2)** and **(3)** (achievable only if r or $sk_{\hat{t}}$ is known) without knowing r and $sk_{\hat{t}}$). As these are all random values, this means $\mathcal{C}$ can solve CDHP. This clearly shows that WSA scheme proposed by Steve Lu et al. cannot be reduced to the underlying Waters signature scheme(WSS) $\square$