Deterministic Identity Based Signature Scheme and its Application for Aggregate Signatures

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. The revolutionary impact offered by identity based cryptography is phenomenal. This novel mechanism was first coined by Adi Shamir in 1984. Since then, several identity based signature schemes were reported. But surprisingly, none of the identity based signature scheme is having the property of determinism and does rely on bilinear pairing. We think positively in answering this long standing question of realizing deterministic identity based signature in composite order groups and we succeed in developing a signature scheme based on RSA assumption and is deterministic. It is indeed helpful in devising variants of signature primitive. Fully aggregateable identity based signature schemes without prior communication between the signing parties is an interesting issue in identity based cryptography. It is easy to see that deterministic identity based signature schemes lead to full aggregation of signatures without the aforementioned overhead. The major contribution of this paper is a novel deterministic identity based signature scheme whose security relies on the strong RSA assumption and random oracles. Based on this newly proposed deterministic identity based signature scheme, we design an identity based aggregate signature scheme which achieves full aggregation in one round. We formally prove the schemes to be existentially unforgeable under adaptive chosen message and identity attack.

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

1 Introduction

The concept of Identity Based Cryptography(IBC) was introduced by Adi Shamir [23] in 1984. The distinguishing characteristic of identity based cryptography is the ability to use any string, that uniquely identifies a user in the system as the public key. In particular, this string may be the email address, telephone number, or combination of any of these 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 [24] [10] [26] [16]. 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. In applications such as e-Banking, 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 expected to be 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 $\sigma_1, \ldots, \sigma_t$). In identity based systems, the PKG generates the private key

[*] 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

Table 1. Existing Identity Based Schemes

Scheme	Master Public and Private Key	Private Key	Signature	Assumption
Cha-Cheon [10]	$MPK = \langle P, sP \rangle$ $MSK = \langle s \rangle$	$D_A = sQ_A$	$U = rQ_A$ $h = H_1(m, U)$ $V = (r + h)D_A$	GDH
Barreto [2]	$MPK = \langle P, sP \rangle$ $MSK = \langle s \rangle$	$D_A = sQ_A$	$U = rP$ $H = H_2(m, U)$ $V = rH + D_A$	GDH
Sakai [21]	$MPK = \langle P, sP \rangle$ $MSK = \langle s \rangle$	$D_A = \frac{1}{s + q_A}$	$U = rP$ $h = H_1(m, U)$ $V = (r + h)D_A$	q-SDH
Galindo [12]	$MPK = \langle P, sP \rangle$ $MSK = \langle s \rangle$	$D_A =$ $\langle X_A = x_A P,$ $d_A = x_A + sq_A \rangle$	$U_1 = rP, U_2 = X_A$ $h = H_1(m, U)$ $V = x_A + d_A h$	DL
Herranz [15]	$MPK = \langle P, sP \rangle$ $MSK = \langle s \rangle$	$D_A =$ $\langle X_A = x_A P,$ $d_A = x_A + sq_A \rangle$	$U_1 = X_A$ $H = H_2(m)$ $V = d_A H$	CDH
Sharmila [22]	$MPK = \langle P, P_1 = s_1 P$ $P_2 = s_2 P \rangle$ $MSK = \langle s_1, s_2 \rangle$	$D_A =$ $\langle Y_A = x_A P_2, H_A = H_3(ID_A, Y_A)$ $X_A = x_A H_A, q_A = H_1(ID_A, Y_A),$ $d_A = s_2 x_A + s_1 q_A \rangle$	$U_1 = X_A$ $U_2 = Y_A$ $H = H_2(m, ID_A)$ $V = d_A H$	CDH
Shamir [23]	$MPK = \langle N, e \rangle$ $MSK = \langle d \rangle$	$H_A = H_3(ID_A)$ $D_A = H_A^d$	$U_1 = R^e$ $H = H_1(m)$ $V = R + D_A^h$	RSA

and hands over the same to the user. Users use the private key to generate signature on messages. In real world scenario, a signature or signed document is the message with sign of the user. Even if the same message is signed multiple times, the signature will be same. This property makes it deterministic. But in the digital scenario, it is not true always because of the random seed involved in generating the signature. BLS short signature using bilinear pairing and based on GDH assumption[9], FDH signature without bilinear pairing based on RSA assumption [4] are examples of deterministic signature in PKI based setting. In the identity based setting, there are several signature schemes using bilinear pairing namely [10], [2], [21], [15]. The only non-pairing based scheme in identity based setting is given by Galindo [12]. Among all identity based signature schemes, only the scheme [15] proposed by Herranz et al. is deterministic. This scheme is based on GDH assumption and uses bilinear pairing. The idea behind the scheme by Herranz is to use Schnorr signature for generation of private key of a user by PKG and BLS for generation of signature on message by user. Because of the randomness involved in private key, this is formally proved to be secure in the random oracle model with the aid of forking lemma. We summarize the properties of existing well known identity based signatures in **Table-1** and **Table-2**. Determinism and tight reduction are among the most desirable properties that a signature scheme should possess so that it can be used efficiently for aggregation. From **Table-1** and **Table-2** it is clear that none of the existing signature schemes offer this property. Also, we give a first realization for achieving this idealistic property without pairing, based on RSA assumption. the private key produced by the PKG itself is a signature of the PKG on the identity (or the hash of the identity) of the user. Thus, if PKG has used some random nonce then the private key will have a generic form (K, P), where K is a random value depending on the nonce and P is a value depending on the identity of the user. When a user uses such a private key and a random value to generate a signature on a message m, then the signature σ may have a generic form (K, R, M), where K is the 'private key' part, R is the 'random' part depending on the randomness used in the generation of the signature and M is the message part depending on the message m that is signed. We shall use the notation discussed above to describe informally on the nature and for of aggregate signatures.

Table 2. Properties of Existing Identity Based Schemes

Scheme	PKI Signature Used For Identity Based Private Key	PKI Signature Used For Identity Based Signing	Without Bilinear Pairing	Proof Without Forking Lemma	Deterministic
Cha-Cheon [10]	BLS	Custom	No	No	No
Barreto [2]	BLS	Custom	No	Yes	No
Sakai [21]	Sakai	Custom	No	No	No
Galindo [12]	Schnorr	Schnorr	Yes	No	No
Herranz [15]	Schnorr	BLS	No	No	Yes
Sharmila [22]	Custom [22]	BLS	No	Yes	Yes
Shamir [23]	RSA	Custom	Yes	-	No
Ours	FDH-RSA	Custom	**Yes**	**Yes**	**Yes**

Let $\alpha = |K|$, $\beta = |R|$ and $\gamma = |M|$, where $|X|$ denotes the size of X. Let $\sigma_{agg} = (K_{agg}, R_{agg}, M_{agg})$ denote the aggregation of t signatures $\sigma_i = (K_i, R_i, M_i)$ generated by user U_i for $1 \leq i \leq t$ for the messages $\{m_1, \ldots, m_t\}$ respectively. We let $|\{U_1, \ldots, U_t\}| = u$, where $1 \leq u \leq t$, to allow the possibility of some users generating more than one signed document. We further assume that $|\{K_1, \ldots, K_t\}| = u$ (The number of random values used in generating the private keys of the u distinct users participating in the aggregation process). Note that $|\{R_1, \ldots, R_t\}| = t$ because for generating each signed document a distinct random value might have been used. Further note that $|\{M_1, \ldots, M_t\}| = t$, and $|\{m_1, \ldots, m_t\}| = k$, for $1 \leq k \leq t$, to allow the possibility of same message being signed by different users. For the specific case where $k = 1$, all users produce signature on one single message and this is referred to as *multi-signature*.

A naive and direct clubbing of all these signed documents will result in $\sigma_{agg} = (K_{agg}, R_{agg}, M_{agg})$, where $|K_{agg}| = u\alpha$, $|R_{agg}| = t\beta$ and $|M_{agg}| = t\gamma$ and hence $|\sigma_{agg}| = \mathcal{O}(u\alpha + t\beta + t\gamma)$. Now 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:

- $\sigma_{agg} = \langle K_1, \ldots K_u, R_1, \ldots, R_t, M_{agg} \rangle$ - 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 $|\sigma_{agg}| = \mathcal{O}(u\alpha + t\beta + \gamma)$. That is, the size of the aggregate signature is dependent on the number of users u and the number of signatures aggregated but independent of the number of message parts.
- $\sigma_{agg} = \langle K_{agg}, R_1, \ldots, R_t, M_{agg} \rangle$ - That is, R_i's are propagated in full without any compression but the K_i's as well as M_i's are all combined to form K_{agg} and M_{agg}. The size of the aggregate signature is $|\sigma_{agg}| = \mathcal{O}(\alpha + t\beta + \gamma)$.
- $\sigma_{agg} = \langle K_1, \ldots, K_t, M_{agg} \rangle$ - This form of aggregation will result if users use deterministic signature schemes to produce individual signatures. Thus $R_1, \ldots, R_t$ will not be present in such schemes. Here, $|\sigma_{agg}| = \mathcal{O}(u\alpha + \gamma)$. That is the size of the aggregate signature depends on the number of distinct users and not on the number of signatures or messages. The scheme described in [15] produces this kind of aggregation.

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

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.

If there is no randomness used in private key generation, then a generic form of the signature in an identity based system may be written as (R, M), where R is the random part depending on the random value used during the production of signature and M is the message part depending on the message. If σ_{agg} is aggregation of t signatures $\sigma_i = (R_i, M_i)$, for $1 \leq i \leq t$ and if $|\sigma_{agg}|$ depends either on the number of messages or the number of signatures (or both) then it will be referred as *partial aggregation*. The scheme described in [25] produces this kind of aggregation. Since R_i's are sent without any compression, there is no need for the signers to communicate the individual random values prior to the generation of individual signatures. The notion of round is used to measure the communication done among signers. Thus, the scheme in [25] does not involve any rounds of communication. If $|\sigma_{agg}|$ is independent of the number of messages and number of signatures it will be referred as *full aggregation*. The schemes described in [11], [13], [6] and [1] produces this kind of aggregation. Here since all R_i's are compressed to form R_{Agg}, the signers have to communicate among themselves to agree upon a common randomness, prior to the generation of individual signatures. Thus one round of communication is required before aggregating the signatures. When the aggregation is done in a sequential order, the number of rounds (interaction) between the signers is linear in 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.

As stated before, for deterministic schemes, no random value is used either in private key generation for users or during signature generation by users. Thus, the signature $\sigma = (M)$ consists of just message dependent part. Let σ_{agg} be the aggregation of t signatures $\sigma_i = (M_i)$, for $1 \leq i \leq t$. If $|\sigma_{agg}|$ depends on the number of signatures or number of messages (or both) it is called *partial aggregation* and if $|\sigma_{agg}|$ is independent of both the number of messages and signatures, then it is called *full aggregation*. As there are no random values involved in deterministic signatures, prior communication is not required to generate aggregate signatures.

Related Work: Many well known PKI based aggregate signatures are available in the literature [8, 18, 3, 19, 20]. 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, 3, 19]. 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 [13], 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. [6] proposed a sequential aggregate signature scheme (in-fact, the security of their original schemes [5] and [7] 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. [17]). Their new scheme [6] 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.

Our contribution: The major contribution of this paper is a secure identity based deterministic signature scheme, proved secure in the random oracle model. That is, there is no randomness deployed either in the private key generation process for the users or the signature generation process by the user. Thus, these kinds of signatures can be efficiently aggregated without any prior communication among signers and this settles an open problem posed in [17]. Moreover, our scheme is 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.

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

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

The following table summarizes the current state of the art in the research of identity based aggregate signatures which achieves full aggregation and are provably secure in the random oracle model. The table includes the communication and computational complexity of our new scheme IBAS along with the other existing schemes. The terms used in **Table-3** are explained here. ND - Nondeterministic Signature, D-

Table 3. State of the art survey of IBAS schemes

Schemes	Agg Sign Len	Sign Cost (/ user)	Agg Verify (t users)	Hard Prob	Sign Type (D/ND)	Rounds	Agg Mode (G/S)
[11]	$2\|\mathbb{G}\|$	3[M]	2[P]+t[M]	CDH	ND	2	G
[1]	$2\|\mathbb{Z}_n^*\| + \|\kappa\| + \|log(l)\|$	2[E]	t[E]	RSA	ND	2	G
[6]	$3\|\mathbb{G}\|$	7[M]	6[P]+t[M]	CDH-type	ND	-	S
[13]	$2\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$	5[M]	3[P]+t[M]	GAP-DH	ND	-	S
IBAS	$\|\mathbb{Z}_n^*\|$	2[E]	$(2t+1)$[E]	strong RSA	D	1	G

Deterministic Signature, G- General aggregation, 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 $\mathbb{G}_T$.

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 $c \equiv a^b$ mod 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:

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

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 framework for a deterministic identity based signature scheme (D-IBS) consists of the first four algorithms described below, namely **Setup**, **Extract**, **Sign** and **Verify**. If the signature scheme is deterministic the signature for a message is always the same for every of invocation of the sign algorithm.

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.

Extract: The user provides his identity ID to the PKG. The PKG runs this algorithm with identity ID, *params* and *msk* as the input and obtains the private key D. The private key D is sent to user through a secure channel.

Sign: For generating a signature on a message m, the user provides his identity ID, his private key D, *params* and the message m as input. This algorithm generates a valid signature σ on message m by the user.

Verify: This algorithm on input a signature σ on message m by the user with identity ID, checks whether σ is a valid signature on message m by ID. If true it outputs "*Valid*", else it outputs "*Invalid*".

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 of D-IBS

We define the security model for the existential unforgeability of a deterministic identity based signature scheme under adaptive chosen identity and message attack in this section. A D-IBS scheme is secure against existential forgery, under adaptive chosen identity and message attack if no probabilistic polynomial time forger $\mathcal{F}$ has non-negligible advantage in the following game:

Setup phase: The challenger $\mathcal{C}$ runs the setup algorithm and generates the system parameters *params* and the master secret key *msk*. Now, $\mathcal{C}$ gives *params* to the forger $\mathcal{F}$ and keeps *msk* secret.

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:

- **Extract oracle:** When $\mathcal{F}$ makes a query with an identity ID as input, $\mathcal{C}$ outputs D, the private key of ID to $\mathcal{F}$.
- **Signing oracle:** When $\mathcal{F}$ makes a signing query with identity ID and message m, $\mathcal{C}$ outputs a valid signature σ on m by ID.

Forgery phase: $\mathcal{F}$ outputs a signature σ, with ID_S as signer, and on a message m^*. $\mathcal{F}$ wins the game if σ is a valid signature and $\mathcal{F}$ has not queried for the signature corresponding to (ID_S, m^*) pair from the sign oracle and private key corresponding to ID_S. The advantage of $\mathcal{F}$ is given by,

$$Adv_{\mathcal{F}}^{D-IBS} = \{Pr[\mathcal{F}(Verify(\sigma) = valid)\}$$

3.2 Existential Unforgeability of IBAS

We define the security model for the existential unforgeability of an IBAS scheme under adaptive chosen identity and message attack in this section. An IBAS scheme is secure against existential forgery under adaptive chosen identity and 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.

Training phase: After the setup is done, $\mathcal{F}$ starts interacting with $\mathcal{C}$ by querying the oracles provided by $\mathcal{C}$ in the following way:

- **Extract oracle:** When $\mathcal{F}$ makes a query with an identity ID as input, $\mathcal{C}$ outputs D, the private key of ID to $\mathcal{F}$.
- **Signing oracle:** When $\mathcal{F}$ makes a signing query with identity ID and message m, $\mathcal{C}$ outputs a valid signature σ on m by ID.

  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 say $ID_S \in \{ID_i\}_{i=1\,to\,t}$, for which the private key was not queried by $\mathcal{F}$ and let m_S the message corresponding to ID_S. 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_S, m_S) pair from the sign oracle.

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

4 Deterministic Identity Based Signature Scheme (D-IBS)

In this section, we propose a new deterministic identity based signature scheme and also prove the scheme to be existentially unforgeable under adaptive chosen message and adaptive chosen identity attack in the random oracle model. The deterministic identity based signature scheme consists of the following algorithms:

- **Setup(κ):** Given κ as input, the PKG generates $params$ and msk by performing the following:
  - Chooses two primes p and q of size κ, 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 $\phi(n) = (p-1)(q-1)$. Therefore $|n| = 2\kappa$ and $|\phi(n)| = 2\kappa$.
  - Chooses e, such that $|e| = \kappa/4$ and computes d such that $ed \equiv 1 \bmod \phi(n)$.
  - It also chooses three cryptographic hash functions $H_0 : \{0,1\}^* \times \{0,1\} \rightarrow \mathbb{Z}_n^*$, $H_1 : \{0,1\}^{l_m} \times \{0,1\}^{l_1} \times \{0,1\} \rightarrow \{0,1\}^{\kappa/2}$ and $H_2 : \{0,1\}^{l_m} \times \{0,1\}^{l_1} \times \{0,1\} \rightarrow \{0,1\}^{\kappa/2}$. Here l_m is the size of message and l_1 is the size of identity of a user.
  - Now, PKG publicizes the system parameters, $params = \langle \kappa, n, e, H_0, H_1, H_2 \rangle$ and keeps the factors of n, namely p, q and the secret inverse d as the master secret key msk.
- **Extract(ID):** The user provides his identity ID to PKG. The PKG performs the following to find out the private key of the corresponding user:
  - Compute $g_0 = H_0(ID, 0)$ and $g_1 = H_0(ID, 1)$.
  - Compute $d_0 = (g_0)^d \; mod \; n$ and $d_1 = (g_1)^d \; mod \; n$.
  - The private key $D = \langle d_0, d_1 \rangle$ is sent to the corresponding user through a secure and authenticated channel.
- **Sign(m, ID, D):** To generate a deterministic signature on a message m, the user with identity ID performs the following:

- Picks $\beta \in_R \{0,1\}$
- Computes $h_1 = H_1(m, ID, \beta)$ and $h_2 = H_2(m, ID, \beta)$.
- Computes $\sigma = (d_0)^{h_1}(d_1)^{h_2}\ mod\ n$.

Now, $\langle \sigma, \beta \rangle$ is the signature on m by user with identity ID.

It should be noted that β is random from others view but fixed with respect to the signer. As in [14], in order to avoid maintaining a record of all previous message/signature pairs, the signer can generate β as $\beta = PRF(D, ID, m)$, where $PRF()$ is a private random function (private to the signer). Thus, for a corresponding private key D, identity ID and message m, there is only one possibility of β.

– **Verify(m, σ, β, ID):** In order to verify the validity of a signature $\langle \sigma, \beta \rangle$ with respect to the identity ID and message m, the verifier performs the following:
- Computes $g_0 = H_0(ID, 0)$ and $g_1 = H_0(ID, 1)$.
- Computes $h_1' = H_1(m, ID, \beta)$ and $h_2' = H_2(m, ID, \beta)$.
- Checks whether $\sigma^e\ mod\ n \overset{?}{=} (g_0)^{h_1'}(g_1)^{h_2'}\ mod\ n$
- If the above check holds, outputs "$Valid$", otherwise outputs "$Invalid$".

Correctness of verification:

$$L.H.S = \sigma^e = ((d_0)^{h_1'}(d_1)^{h_2'})^e = ((g_0^d)^{h_1'}(g_1^d)^{h_2'})^e = (g_0)^{h_1'}(g_1)^{h_2'} = R.H.S$$

4.1 Existential Unforgeability of D-IBS:

Theorem 1. *The identity based signature scheme (D-IBS) is EUF-D-IBS-CMA secure in the random oracle model under adaptive chosen message and adaptive chosen identity attack, if the strong RSA problem is assumed to be hard in $\mathbb{Z}_n^*$, where $n = pq$, and 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, such that $(p-1)/2$ and $(q-1)/2$ are also primes. $\mathcal{C}$ can make use of $\mathcal{F}$ to compute a and b such that $c \equiv a^b$ mod n, by playing the following interactive game with $\mathcal{F}$. (Note that if mod operation is not specified then the computation is pure integer computation.)

Setup: $\mathcal{C}$ begins the game by setting up the system parameters as in the D-IBS scheme. $\mathcal{C}$ takes n from the instance of the strong RSA problem, chooses e such that $|e| = \kappa/4$, $e = xy$ for some arbitrary x and y, and $|x| = |y| = \kappa/8$. $\mathcal{C}$ chooses w such that $|w| = \kappa/8$. $\mathcal{C}$ sends the public parameters $params = \langle n, e \rangle$ to $\mathcal{F}$. $\mathcal{C}$ stores w, x and y for future use in $\mathcal{O}_{H_0}$ and $\mathcal{O}_{H_2}$ oracles. $\mathcal{C}$ also designs the three hash functions H_0, H_1 and H_2 as random oracles $\mathcal{O}_{H_0}$, $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$. $\mathcal{C}$ maintains three lists L_{H_0}, L_{H_1} and L_{H_2} in order to consistently respond to the queries to the random oracles $\mathcal{O}_{H_0}$, $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$ respectively.

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 $\mathcal{O}_{H_0}(.)$ oracle is queried with ID and both 0 and 1 as inputs, before any other oracle is queried with the corresponding identity as the input parameters.

Oracle $\mathcal{O}_{H_0}(ID, l \in \{0,1\})$: We make a simplifying assumption that $\mathcal{A}$ queries the $\mathcal{O}_{H_0}$ oracle with distinct inputs in each query. If the oracle was queried with ID as input for $l = 0$ first, the next query with the same identity can be made with $l = 1$. There is no loss of generality due to this assumption, because, if the an identity is repeated with the same l value, by definition the oracle consults the list L_{H_0} and gives the same response. Thus, we assume that $\mathcal{A}$ asks $2q_{H_0}$ distinct queries for q_{H_0} distinct identities. Among these q_{H_0} identities, a random identity has to be selected as target identity and it is done as follows.

$\mathcal{C}$ selects a random index T, where $1 \leq T \leq 2q_{H_0}$. $\mathcal{C}$ does not reveal T to $\mathcal{A}$. When $\mathcal{A}$ generates the T^{th} query on ID_T, $\mathcal{C}$ decides to fix ID_T as target identity for the forgery phase. Moreover, $\mathcal{C}$ responds to $\mathcal{A}$ as follows:

– If a tuple of the form $\langle ID, l, d_l, g_l, * \rangle$ exists in list L_{H_0}, then it returns g_l as response.
– If the tuple of the form $\langle ID, l, d_l, g_l, * \rangle$ does not exist in list L_{H_0}, then it does the following.
  - If it is not the T^{th} query i.e. $i \neq T$, then $\mathcal{C}$ performs the following:
    - $*$ Chooses $d_0, d_1 \in_R \mathbb{Z}_n^*$ and sets $H_0(ID, j) = g_j = (d_j)^e\ mod\ n$ for j=0,1.

    * Stores the tuple $\langle ID_i, j, d_j, g_j, - \rangle$(for j=0,1) to list L_{H_0} and returns g_l as the response.
- If it is the T^{th} query i.e. $i = T$, then $\mathcal{C}$ performs the following:
    * Chooses r_0 and r such that $|r_0| = \kappa/2$ and $|r| = \kappa/4$.
    * Sets $H_0(ID_T, 0) = g_0 = c^{xw} \ mod \ n$
    * Sets $H_0(ID_T, 1) = g_1 = c^{x+re} \ mod \ n$
      Note: c is taken from the strong RSA instance, x is chosen by $\mathcal{C}$ during setup.
    * Here d_0, d_1 are not known to $\mathcal{C}$ and hence $\mathcal{C}$ sets $d_j =$ " $-$ " for j=0,1.
    * Stores the tuples $\langle ID, j, -, g_j, - \rangle$ (for j=0,1) to list L_{H_0} and returns g_l 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 mathematically consistent private and public key.

Note: For $\lambda \in \{0,1\}$, $\bar{\lambda}$ represents the negation of λ.

Oracle $\mathcal{O}_{H_1}(m, ID, \beta)$*:* When this query is made by $\mathcal{F}$, $\mathcal{C}$ does the following:

– If a tuple of the form $\langle m, ID, \lambda, \texttt{useful}, s^{(m)}_{\lambda 1}, t^{(m)}_{\lambda 1}, u^{(m)}, h^{(m)}_{1\lambda} \rangle$, where $\lambda = \beta$ exists in the list L_{H_1} then return $h^{(m)}_{1\lambda}$.
– If a tuple of the form $\langle m, ID, \bar{\lambda}, \texttt{useful}, s^{(m)}_{\bar{\lambda} 1}, t^{(m)}_{\bar{\lambda} 1}, v^{(m)}, h^{(m)}_{1\bar{\lambda}} \rangle$, where $\bar{\lambda} = \beta$ exists in the list L_{H_1} then return $h^{(m)}_{1\bar{\lambda}}$.
– Else, performs the following:
- Chooses $\lambda \in_R \{0,1\}$.
- For λ, perform the following:
    * Choose $s^{(m)}_{\lambda 1}, t^{(m)}_{\lambda 1}, s^{(m)}_{\lambda 2}, t^{(m)}_{\lambda 2}, u^{(m)} \in_R \{0,1\}^{\kappa/4}$.
    * Compute $h^{(m)}_{1\lambda} = u^{(m)} + s^{(m)}_{\lambda 1} e + t^{(m)}_{\lambda 1} y$.
    * Compute $h^{(m)}_{2\lambda} = -u^{(m)} w + s^{(m)}_{\lambda 2} e + t^{(m)}_{\lambda 2} y + 1$.
    * Set $\texttt{useful} = 1$.
    * Store the tuple $\langle m, ID, \lambda, \texttt{useful}, s^{(m)}_{\lambda 1}, t^{(m)}_{\lambda 1}, u^{(m)}, h^{(m)}_{1\lambda} \rangle$ in list L_{H_1}.
    * Store the tuple $\langle m, ID, \lambda, \texttt{useful}, s^{(m)}_{\lambda 2}, t^{(m)}_{\lambda 2}, u^{(m)}, h^{(m)}_{2\lambda} \rangle$ in list L_{H_2}.
- For $\bar{\lambda}$, where $\bar{\lambda} = \neg\lambda$ perform the following:
    * Choose $s^{(m)}_{\bar{\lambda} 1}, t^{(m)}_{\bar{\lambda} 1}, s^{(m)}_{\bar{\lambda} 2}, t^{(m)}_{\bar{\lambda} 2}, v^{(m)} \in_R \{0,1\}^{\kappa/4}$.
    * Compute $h^{(m)}_{1\bar{\lambda}} = v^{(m)} + s^{(m)}_{\bar{\lambda} 1} e + t^{(m)}_{\bar{\lambda} 1} y$.
    * Compute $h^{(m)}_{2\bar{\lambda}} = -v^{(m)} w + s^{(m)}_{\bar{\lambda} 2} e + t^{(m)}_{\bar{\lambda} 2} y$.
    * Set $\texttt{useful} = 0$.
    * Store the tuple $\langle m, ID, \bar{\lambda}, \texttt{useful}, s^{(m)}_{\bar{\lambda} 1}, t^{(m)}_{\bar{\lambda} 1}, u^{(m)}, h^{(m)}_{1\bar{\lambda}} \rangle$ in list L_{H_1}.
    * Store the tuple $\langle m, ID, \bar{\lambda}, \texttt{useful}, s^{(m)}_{\bar{\lambda} 2}, t^{(m)}_{\bar{\lambda} 2}, u^{(m)}, h^{(m)}_{2\bar{\lambda}} \rangle$ in list L_{H_2}.
- If $\beta = \lambda$, output $h_{1\lambda}$.
- If $\beta = \bar{\lambda}$, output $h_{1\bar{\lambda}}$.

Oracle $\mathcal{O}_{H_2}(m, ID, \beta)$*:* When this query is made by $\mathcal{F}$, $\mathcal{C}$ does the following:

– If a tuple of the form $\langle m, ID, \lambda, \texttt{useful}, s^{(m)}_{\lambda 2}, t^{(m)}_{\lambda 2}, u^{(m)}, h^{(m)}_{2\lambda} \rangle$, where $\lambda = \beta$ exists in the list L_{H_2} then return $h^{(m)}_{2\lambda}$.
– If a tuple of the form $\langle m, ID, \bar{\lambda}, \texttt{useful}, s^{(m)}_{\bar{\lambda} 2}, t^{(m)}_{\bar{\lambda} 2}, v^{(m)}, h^{(m)}_{2\bar{\lambda}} \rangle$, where $\bar{\lambda} = \beta$ exists in the list L_{H_2} then return $h^{(m)}_{2\bar{\lambda}}$.
– Else, perform the following:
- Chooses $\lambda \in_R \{0,1\}$.
- For λ, perform the following:
    * Choose $s^{(m)}_{\lambda 1}, t^{(m)}_{\lambda 1}, s^{(m)}_{\lambda 2}, t^{(m)}_{\lambda 2}, u^{(m)} \in_R \{0,1\}^{\kappa/4}$.
    * Compute $h^{(m)}_{1\lambda} = u^{(m)} + s^{(m)}_{\lambda 1} e + t^{(m)}_{\lambda 1} y$.
    * Compute $h^{(m)}_{2\lambda} = -u^{(m)} w + s^{(m)}_{\lambda 2} e + t^{(m)}_{\lambda 2} y + 1$.
    * Set $\texttt{useful} = 1$.

    - $*$ Store the tuple $\langle m, ID, \lambda, \texttt{useful}, s_{\lambda 1}^{(m)}, t_{\lambda 1}^{(m)}, u^{(m)}, h_{1\lambda}^{(m)}\rangle$ in list L_{H_1}.
    - $*$ Store the tuple $\langle m, ID, \lambda, \texttt{useful}, s_{\lambda 2}^{(m)}, t_{\lambda 2}^{(m)}, u^{(m)}, h_{2\lambda}^{(m)}\rangle$ in list L_{H_2}.
- For $\bar{\lambda}$, where $\bar{\lambda} = \neg\lambda$ perform the following:
    - $*$ Choose $s_{\bar{\lambda}1}^{(m)}, t_{\bar{\lambda}1}^{(m)}, s_{\bar{\lambda}2}^{(m)}, t_{\bar{\lambda}2}^{(m)}, v^{(m)} \in_R \{0,1\}^{\kappa/4}$.
    - $*$ Compute $h_{1\bar{\lambda}}^{(m)} = v^{(m)} + s_{\bar{\lambda}1}^{(m)} e + t_{\bar{\lambda}1}^{(m)} y$.
    - $*$ Compute $h_{2\bar{\lambda}}^{(m)} = -v^{(m)} w + s_{\bar{\lambda}2}^{(m)} e + t_{\bar{\lambda}2}^{(m)} y$.
    - $*$ Set $\texttt{useful} = 0$.
    - $*$ Store the tuple $\langle m, ID, \bar{\lambda}, \texttt{useful}, s_{\bar{\lambda}1}^{(m)}, t_{\bar{\lambda}1}^{(m)}, u^{(m)}, h_{1\bar{\lambda}}^{(m)}\rangle$ in list L_{H_1}.
    - $*$ Store the tuple $\langle m, ID, \bar{\lambda}, \texttt{useful}, s_{\bar{\lambda}2}^{(m)}, t_{\bar{\lambda}2}^{(m)}, u^{(m)}, h_{2\bar{\lambda}}^{(m)}\rangle$ in list L_{H_2}.
- If $\beta = \lambda$, output $h_{2\lambda}$.
- If $\beta = \bar{\lambda}$, output $h_{2\bar{\lambda}}$.

Remark: From the definition of H_0, it is clear that the oracle $\mathcal{O}_{H_0}()$ must output a random element in $\mathbb{Z}_n^*$. The $\mathcal{O}_{H_0}()$ oracle indeed returns a random value of $\mathbb{Z}_n^*$ because d_0 and d_1 are randomly chosen from $\mathbb{Z}_n^*$. The definition of the oracles $\mathcal{O}_{H_1}()$ and $\mathcal{O}_{H_2}()$ suggest that, when we model them as random oracles, we must use $\kappa/2$ bits of randomness for the output values. However, in the construction defined above, both $\mathcal{O}_{H_1}()$ and $\mathcal{O}_{H_2}()$ use only $\kappa/4$ bits of randomness. This imperfect and slightly weak simulation would reduce the reliability factor by $1/2^{\kappa/4}$, which, for sufficiently large κ, is negligible. Hence, we ignore this factor even in the probability analysis.

Oracle $\mathcal{O}_{Extract}(ID)$: $\mathcal{C}$ checks whether tuples of the form $\langle ID, 0, d_0, g_0, -\rangle$ and $\langle ID, 1, d_1, g_1, r\rangle$ exists in the list L_{H_0}, if so returns the corresponding d_0 and d_1 as the private keys corresponding to the identity ID. However, if $ID = ID_T$, $\mathcal{C}$ aborts.

Oracle $\mathcal{O}_{Sign}(m, ID)$: $\mathcal{C}$ checks whether $ID \overset{?}{=} ID_T$ and performs the following to generate the signature on m by ID:

- If $ID \neq ID_T$, then $\mathcal{C}$ knows the private key corresponding to ID, so $\mathcal{C}$ chooses $\beta \in_R \{0,1\}$ and performs the signing as per the protocol and generates σ, after querying $\mathcal{O}_{H_1}(m, ID, \beta)$ and $\mathcal{O}_{H_2}(m, ID, \beta)$.
- If $ID = ID_T$, then $\mathcal{C}$ checks whether a tuple corresponding to $(m, ID_T, -)$ is found in L_{H_1} and H_{H_2}. If it does not exist, $\mathcal{C}$ invokes the $\mathcal{O}_{H_1}(m, ID_T, 0)$ and $\mathcal{O}_{H_2}(m, ID_T, 0)$ oracles. Then, $\mathcal{C}$ simulates the signing algorithm as follows: (because, $\mathcal{C}$ does not know the private key corresponding to ID_T):
    - $\mathcal{C}$ retrieves the tuples corresponding to m, ID_T from the lists L_{H_1} and L_{H_2}, for which the flag $\texttt{useful}$=0. Let $\langle m, ID, \gamma, \texttt{useful} = 0, s_{\gamma 1}^{(m)}, t_{\gamma 1}^{(m)}, v^{(m)}, h_{1\gamma}^{(m)}\rangle$ be the tuple in list L_{H_1} and $\langle m, ID, \gamma, \texttt{useful} = 0, s_{\gamma 2}^{(m)}, t_{\gamma 2}^{(m)}, v^{(m)}, h_{2\gamma}^{(m)}\rangle$ be the tuple in list L_{H_2}.
    - Set $\beta = \gamma$.
    - Computes $\sigma = c^{xws_{\gamma 1}^{(m)}} c^{wt_{\gamma 1}^{(m)}} c^{xs_{\gamma 2}^{(m)}} c^{t_{\gamma 2}^{(m)}} c^{-rv^{(m)}w} c^{rs_{\gamma 2}^{(m)}e} c^{rt_{\gamma 2}^{(m)}y}$ *mod* n.
- Sends σ, β as the signature on the message m by identity ID_T.

The verification of a signature is done by checking whether $\sigma^e \overset{?}{=} (g_0)^{h_1}(g_1)^{h_2}$. We need to verify only the case when $ID = ID_T$ because in other cases, the signature is generated as per the protocol.

Since $\mathcal{C}$ choosed the tuple corresponding to m, ID_T from L_{H_1} and L_{H_2} such that $\texttt{useful} = 0$, $h_1 = h_{1\gamma}^{(m)} = v^{(m)} + s_{\gamma 1}^{(m)} e + t_{\gamma 1}^{(m)} y$ and $h_2 = h_{2\gamma}^{(m)} = -v^{(m)} w + s_{\gamma 2}^{(m)} e + t_{\gamma 2}^{(m)} y$. The simulated signature generated by the $\mathcal{O}_{Sign}$ oracle passes the verification test as shown below:

$$
\begin{aligned}
R.H.S &= (g_0)^{h_1}(g_1)^{h_2} = (g_0)^{h_{1\gamma}^{(m)}}(g_1)^{h_{2\gamma}^{(m)}}\\
&= (c^{xw})^{h_{1\gamma}^{(m)}}(c^{x+re})^{h_{2\gamma}^{(m)}}\\
&= c^{xw(v^{(m)}+s_{\gamma 1}^{(m)}e+t_{\gamma 1}^{(m)}y)} c^{x+re(-v^{(m)}w+s_{\gamma 2}^{(m)}e+t_{\gamma 2}^{(m)}y)}\\
&= c^{(xwv^{(m)}+xws_{\gamma 1}^{(m)}e+xwt_{\gamma 1}^{(m)}y)} c^{(-xv^{(m)}w+xs_{\gamma 2}^{(m)}e+xt_{\gamma 2}^{(m)}y)} c^{(-rev^{(m)}w+res_{\gamma 2}^{(m)}e+ret_{\gamma 2}^{(m)}y)}\\
&= c^{(xwv^{(m)}+xws_{\gamma 1}^{(m)}e+wt_{\gamma 1}^{(m)}e)} c^{(-xv^{(m)}w+xs_{\gamma 2}^{(m)}e+t_{\gamma 2}^{(m)}e)} c^{(-rev^{(m)}w+res_{\gamma 2}^{(m)}e+ret_{\gamma 2}^{(m)}y)} \quad \text{(Since } xy = e\text{)}\\
&= c^{(xwv^{(m)}+xws_{\gamma 1}^{(m)}e+wt_{\gamma 1}^{(m)}e-xwv^{(m)}+xs_{\gamma 2}^{(m)}e+t_{\gamma 2}^{(m)}e-rev^{(m)}w+res_{\gamma 2}^{(m)}e+ret_{\gamma 2}^{(m)}y)}\\
&= c^{(xws_{\gamma 1}^{(m)}e+wt_{\gamma 1}^{(m)}e+xs_{\gamma 2}^{(m)}e+t_{\gamma 2}^{(m)}e-rev^{(m)}w+res_{\gamma 2}^{(m)}e+ret_{\gamma 2}^{(m)}y)}\\
&= (c^{(xws_{\gamma 1}^{(m)}} c^{wt_{\gamma 1}^{(m)}} c^{xs_{\gamma 2}^{(m)}} c^{t_{\gamma 2}^{(m)}} c^{-rv^{(m)}w} c^{rs_{\gamma 2}^{(m)}e} c^{rt_{\gamma 2}^{(m)}y)})^e\\
&= \sigma^e = L.H.S
\end{aligned}
$$

Since $R.H.S = L.H.S$, the simulated signature is a valid signature on m by identity ID_T.

Forgery Phase: At the end of the ***Training Phase***, $\mathcal{F}$ produces a forged signature σ^*, β^* on the message m^* as if signed by the user with identity ID_S. If σ^* is a valid signature on m^* and if σ^* satisfies all the constraints given below, then $\mathcal{C}$ can solve the hard problem.

- $ID_S = ID_T$
- Private key of ID_T is not queried to Extract oracle.
- Signature on m^*, ID_T is not queried to Sign Oracle. (This is forbidden in the model because it is a deterministic signature scheme).
- The tuple $\langle m^*, ID_T, \beta^*, \texttt{useful}, s_{\beta^*1}^{(m^*)}, t_{\beta^*1}^{(m^*)}, v^{(m^*)}, h_{1\beta^*}^{(m^*)} \rangle$ corresponding to m^*, ID_T, in list L_{H_1} and the tuple $\langle m^*, ID_T, \beta^*, \texttt{useful}, s_{\beta^*2}^{(m^*)}, t_{\beta^*2}^{(m^*)}, v^{(m^*)}, h_{2\beta^*}^{(m^*)} \rangle$ corresponding to m^*, ID_T, in list L_{H_2} has the flag "`useful`=1".

Now, if the above constraints are satisfied, then $\mathcal{C}$ obtains a and b such that $c = a^b \ mod \ n$ by performing the following:

- The public keys corresponding to ID_T, i.e. $H_0(ID_T, 0)$ and $H_0(ID_T, 1)$, are set to be $\langle g_0 = c^{xw}$ and $g_1 = c^{x+re} \rangle$ by $\mathcal{C}$ while $\mathcal{F}$ performed the $\mathcal{O}_{H_0}(ID_T, .)$ queries (Note that c was 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 keys and the value d, the private keys corresponding to ID_T are $d_0 = g_0^d = c^{xwd}$ and $d_1 = g_1^d = c^{xd+r}$, **implicitly**. However, these values cannot be computed explicitly by $\mathcal{C}$, because $\mathcal{C}$ has no way of computing d. That is why $\mathcal{C}$ has set " $-$ " for these unknowns in $\mathcal{O}_{H_0}$ oracle queries. Thus, the values d_0 and d_1 used in the proof are only hypothetical.
- Since we have the condition that "`useful`=1", $\mathcal{C}$ should have set the $h_{1\beta^*}^{(m^*)} = \mathcal{O}_{H_2}(m^*, ID_T, \beta^*) = u^{(m^*)} + s_{\beta^*1}^{(m^*)} e + t_{\beta^*1}^{(m^*)} y$ and $h_{2\beta^*}^{(m^*)} = \mathcal{O}_{H_2}(m^*, ID_T, \beta^*) = u^{(m^*)} w + s_{\beta^*2}^{(m^*)} e + t_{\beta^*2}^{(m^*)} y + 1$ respectively.
- Now, $\sigma^* = (d_0)^{h_{1\beta^*}^{(m^*)}} (d_1)^{h_{2\beta^*}^{(m^*)}} = (c^{xwd})^{h_{1\beta^*}^{(m^*)}} (c^{xd+r})^{h_{2\beta^*}^{(m^*)}}$ and hence the signature verification holds good for a proper forgery, because $(\sigma^*)^e = ((c^{xwd})^{h_{1\beta^*}^{(m^*)}} (c^{xd+r})^{h_{2\beta^*}^{(m^*)}})^e = (c^{xw})^{h_{1\beta^*}^{(m^*)}} (c^{x+re})^{h_{2\beta^*}^{(m^*)}} = g_0^{h_{1\beta^*}^{(m^*)}} g_1^{h_{2\beta^*}^{(m^*)}}$.
- Hence, the forgery σ^*, submitted by $\mathcal{F}$ is of the following form. Now, for the sake of simplicity, we rename the values $v^{(m^*)} = v$, $s_{\beta^*1}^{(m^*)} = s_1$, $t_{\beta^*1}^{(m^*)} = t_1$, $s_{\beta^*2}^{(m^*)} = s_2$ and $t_{\beta^*1}^{(m^*)} = t_2$ in the following equations.

$$
\begin{aligned}
\sigma^* &= (d_0)^{h_{1\beta^*}^{(m^*)}} (d_1)^{h_{2\beta^*}^{(m^*)}} = (c^{xwd})^{h_{1\beta^*}^{(m^*)}} (c^{xd+r})^{h_{2\beta^*}^{(m^*)}} \\
&= (c^{xwd})^{(v^{(m^*)} + s_{\beta^*1}^{(m^*)} e + t_{\beta^*1}^{(m^*)} y)} (c^{xd+r})^{(-v^{(m^*)} w + s_{\beta^*2}^{(m^*)} e + t_{\beta^*2}^{(m^*)} y + 1)} \\
&= (c^{xwd})^{(v + s_1 e + t_1 y)} (c^{xd})^{(-vw + s_2 e + t_2 y + 1)} (c^r)^{(-vw + s_2 e + t_2 y + 1)} \\
&= c^{(xwdv + xwds_1 e + xwdt_1 y)} c^{(-xdvw + xds_2 e + xdt_2 y + xd)} c^{(-rvw + rs_2 e + rt_2 y + r)} \\
&= c^{(xwdv + xwds_1 e + wdt_1 e)} c^{(-xwdv + xds_2 e + dt_2 e + xd)} c^{(-rvw + rs_2 e + rt_2 y + r)} \text{ (Since } xy = e) \\
&= c^{(xwds_1 e + wdt_1 e + xds_2 e + dt_2 e - rvw + rs_2 e + rt_2 y + r)} c^{xd} \\
&= c^{(xws_1 + wt_1 + xs_2 + t_2 - rvw + rs_2 e + rt_2 y + r)} c^{xd} \text{ (Since } ed \equiv 1 \ mod \ \phi(n))
\end{aligned}
$$

- Let, $z = c^{(xws_1 + wt_1 + xs_2 + t_2 - rvw + rs_2 e + rt_2 y + r)}$ and z can be computed by $\mathcal{C}$ because $\mathcal{C}$ knows the values $r, s_1, s_2, t_1, t_2, v, w, x$ and y.
- Therefore, $\sigma^* = zc^{xd} \Rightarrow c^{xd} = \sigma^*/z \Rightarrow c^{xd} = c^{x(x^{-1}y^{-1})} = c^{y^{-1}} \Rightarrow \sigma^* = zc^{y^{-1}}$. Thus, $\mathcal{C}$ can obtain the solution for the equation $c = a^b \ mod \ n$ with $a = \sigma^*/z$ and $b = y$.

Probability Analysis: The following are the three events during which $\mathcal{C}$ aborts the game.

1. $\mathcal{E}_1$ - The event in which $ID_S \neq ID_T$.
2. $\mathcal{E}_2$ - The event in which the private key of ID_T is queried to the Extract oracle.
3. $\mathcal{E}_2$ - The event in which the tuples in L_{H_1} and L_{H_2} corresponding to the pair (m^*, ID_T) has an entry for the flag `useful` $= 0$.

Let us consider that $\mathcal{F}$ has made q_{H_0}, H_0 oracle queries and q_e extract oracle queries. Then the probability of the three aborting events $\mathcal{E}_1, \mathcal{E}_2$ and $\mathcal{E}_3$ are:

$Pr[\mathcal{E}_1] = \left(1 - \frac{1}{q_{H_0} - q_e}\right)$, $Pr[\mathcal{E}_2] = \left(\frac{q_e}{q_{H_0}}\right)$ and $Pr[\mathcal{E}_2] = \frac{1}{2}$

Let ϵ' be the advantage of $\mathcal{A}$ in breaking the unforgeability of the scheme. We compute ϵ' as the probability of $\mathcal{C}$ proceeding the game with out aborting. Therefore,

$$\begin{aligned}\epsilon' &= \neg\mathcal{E}_1 \vee \neg\mathcal{E}_2 \vee \neg\mathcal{E}_3 = \left(1 - \left(1 - \frac{1}{q_{H_0} - q_e}\right)\right)\left(1 - \frac{q_e}{q_{H_0}}\right)\left(1 - \frac{1}{2}\right)\\ &= \left(1 - 1 + \frac{1}{q_{H_0} - q_e}\right)\left(1 - \frac{q_e}{q_{H_0}}\right)\left(\frac{1}{2}\right) = \left(\frac{1}{q_{H_0} - q_e}\right)\left(\frac{q_{H_0} - q_e}{q_{H_0}}\right)\left(\frac{1}{2}\right)\\ &= \left(\frac{1}{2q_{H_0}}\right)\end{aligned}$$

Thus, the forgery σ^* submitted by the forger $\mathcal{F}$ can be used by $\mathcal{C}$ to successfully solve the strong RSA problem with an advantage $\epsilon \geq \left(\frac{\epsilon'}{2q_{H_0}}\right)$. □

5 Identity Based Aggregate Signature scheme from RSA (IBAS)

We propose the new identity based aggregate signature (IBAS) scheme in this section and prove the existential unforgeability of the scheme.

Intuition behind the scheme: Our identity based aggregate signature scheme is 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 [13] 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.

Deterministic General IBAS: Our scheme is a deterministic identity based aggregate signature scheme, which supports full aggregation, i.e. the size of the aggregate signature is one group element along with the message and the list of identities. The scheme consists of six algorithms, out of which **Setup**, **Extract**, **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 $\{\sigma_i, \beta_i\}_{i=1\,to\,t}$ and the corresponding message identity pairs $\langle m_i, ID_i\rangle$, such that $\forall i = 1$ to t, $\langle\sigma_i, \beta_i\rangle$ 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 and β's is $\mathcal{L} = \{m_i, ID_i, \beta_i\}_{i=1\,to\,t}$.

– **AggregateVerify:** This algorithm takes the identity based aggregate signature σ_{agg} and the corresponding list of message identity pairs, $\mathcal{L} = \{m_i, ID_i, \beta_i\}_{i=1\,to\,t}$ and performs the verification as follows:
  - For all i=1 to t
    Compute $g_{i0} = H_0(ID_i, 0)$
    Compute $g_{i1} = H_0(ID_i, 1)$
    Compute $h'_{i1} = H_1(m_i, ID_i, \beta_i)$ and
    Compute $h'_{i2} = H_2(m_i, ID_i, \beta_i)$
  - If $\sigma_{agg}^{e} \overset{?}{=} \prod_{i=1}^{t}((g_{i0})^{h'_{i1}}(g_{i1})^{h'_{i2}})$, then outputs "*Valid*" else outputs "*Invalid*".

The correctness of verification is straight forward.

Security Proof for IBAS:

In this section, we prove the security of our identity based aggregate signature scheme. 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:

Theorem 2. *Our identity based aggregate signature scheme (IBAS) is EUF-IBAS-CMA secure in the random oracle model under adaptive chosen message and adaptive chosen identity attack, if the strong RSA problem is assumed to be hard 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 scheme.

The EUF-IBAS-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. □

6 Conclusion

In this paper, we have proposed the first deterministic identity based signature scheme, whose security is based on strong RSA assumption. Our scheme proposed in this paper achieves full aggregation and thus the result is attractive. Our IBAS addressed the open problem posed by Hwang et al. in [17], which is to design an identity based aggregate signature scheme where the signers need not have to agree on a common random value. Gentry et al.'s identity based aggregate signature scheme in [13] 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 scheme does not show the weakness of [13]. We have formally proved the security of our scheme in the random oracle model. We have summarized the current state of the art in the research of identity based aggregate signatures which achieves full aggregation and showed the communication and computational complexity of our new scheme IBAS along with the other existing schemes in **Table-3**.

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. Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532. Springer, 2005.
3. 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.
4. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT*, Lecture Notes in Computer Science, pages 399–416. Springer, 1996.
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. In *ACM Conference on Computer and Communications Security, CCS 2007*, pages 276–285. ACM, 2007.

6. 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/.
7. 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.
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. David Galindo and Flavio D. Garcia. A schnorr-like lightweight identity-based signature scheme. In *In Proceedings of 2nd African International Conference on Cryptology, AfricaCrypt 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 2009.
13. 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.
14. Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the diffie-hellman problems. *Journal of Cryptology*, Vol.20(No.4):493–514, 2007.
15. Javier Herranz. Deterministic identity-based signatures for partial aggregation. *The Computer Journal*, vol-49(no-3):322–330, 2006.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, Okinawa, Japan, January, pages 135–148, 2000.
22. S. Sharmila Deva Selvi, S. Sree Vivek, and C. Pandu Rangan. Identity-based deterministic signature scheme without forking-lemma. In *Advances in Information and Computer Security, IWSEC-2011*, volume 7038 of *Lecture Notes in Computer Science*, pages 79–95. Springer, 2011.
23. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO - s4*, Lecture Notes in Computer Science, pages 47–53. Springer, 1984.
24. 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.
25. 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.
26. 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.