Identity Based Deterministic Signature Scheme Without Forking-Lemma

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@cse.iitm.ac.in.

Abstract. Since the discovery of identity based cryptography, a number of identity based signature schemes were reported in the literature. Although, a lot of identity based signature schemes were proposed, the only identity based deterministic signature scheme was given by Javier Herranz. This signature scheme uses Schnorr signature scheme for generating the private key of the users and uses BLS short signature scheme for generating users signature. The security of this scheme was proved in the random oracle model using forking lemma. In this paper, we introduce a new identity based deterministic signature scheme and prove the security of the scheme in the random oracle model, without the aid of forking lemma. Hence, our scheme offers tighter security reduction to the underlying hard problem than the existing identity based deterministic signature scheme.

Keywords: Identity Based Cryptography, Deterministic, Signature, Tight Security, Random Oracle Model, Provable Security, Without Forking-Lemma.

1 Introduction

The concept of using the identity of an entity for deriving the public key is known as Identity Based Cryptography (IBC). This technique was introduced by Adi Shamir in his seminal paper [15] in 1984. This paved way for eliminating the use of certificates for authenticating the public keys of a user (in PKI based system). In identity based system, a trusted authority called Private Key Generator (PKG) generates the private key for the users. The PKG possesses a master public key and master private key and uses the master private key to generate the private key of the users registered with the system. The private key of the user is the signature on the identity of the user (as message) generated by the PKG with the master private key. The user makes use of his/her private key to generate a signature on a message.

Thus the complete description of an identity based signature scheme can be conveniently split as the description of the signature scheme employed by the PKG to create private keys for the users and description of signature generation algorithm executed by a user on a message. The signature schemes used in these two parts may resemble some well known signature schemes or customized signature scheme or they may be deterministic or probabilistic. For example, the identity based signature scheme by Cha and Cheon [5] may be viewed as BLS [4] + a customized scheme, where the BLS signature scheme is used by the PKG to generate private keys of users and users themselves use the customized scheme to produce signed documents. For this scheme, the private key generation is deterministic while the signature generation process is probabilistic. As another example, the scheme by Galindo et al. [6] uses Schnorr signature for private key generation (by PKG) and again a Schnorr signature scheme [14] for the signature generation (by a user). For this scheme, the private key generation as well as signature generation is probabilistic. The scheme by Javier Herranz [9] uses Schnorr signature for private key generation (by PKG) and BLS signature scheme for the signature generation (by a user). Thus in this scheme, the private key generation is probabilistic and the signature generation is deterministic. Table-1 gives a summary of properties of existing identity based signature schemes.

* Currently Head, Indian Statistical Institute, Chennai, India.

Scheme	Private Key	Signing Algorithm	Type of Scheme		Pairing Computation	
			Key	Sign	Sign	Verify
Cha-Cheon [5]	BLS	Custom	D	P	No	Yes
Sakai [13]	BLS	Custom	D	P	No	Yes
Barreto [1]	[16]	Custom	D	P	No	Yes
Galindo [6]	Schnorr	Schnorr	P	P	No	No
Javier [9]	Schnorr	BLS	P	D	No	Yes
Ours	$Basic_{Sign}$	custom	P	D	No	Yes

Table-1: Properties of ID-Based Signatures

P - Probabilistic Signature, D - Deterministic Signature, Custom - Custom designed signing algorithm

Tightness of Security Reduction: In the computational model, proof of security for a signature follows if there does not exist a polynomial time algorithm with the following ability:

The reduction algorithm makes use of a polynomial time algorithm that forges a signature, to construct a polynomial time algorithm that solves the computational hard problem. If there is no polynomial time algorithm for solving the computational hard problem then the existence of such reduction implies that the signature scheme is not breakable in polynomial time.

This security argument is asymptotic. In CDH based signature schemes, forging signatures is infeasible in prime order groups where the size of the security parameter is above some threshold value. For practice, we should exactly know what should be the constraint on security parameter to impose a sufficient infeasible computational bound on the adversary.

Bellare and Rogaway [3] gave the method for exact security analysis that focuses on the computational efficiency of the reduction algorithm. This allows one to quantify the relation between the difficulty of forging a signature and hardness of the underlying hard problem. The relative hardness of forging the signature to that of breaking the computational assumption can be loose, close or tight as pointed out by Micali and Reyzin [10]. In [7], Goh et al. showed that the application of forking lemma [12], for proving security of Fiat-Shamir based signatures makes it inefficient by imposing an increase in the length of the modulus p. In any discrete-log based system of a prime field $\mathbb{Z}_p$, breaking the discrete-log in the index-calculus method works in $\mathcal{O}(exp(\sqrt[3]{|p|}))$. Thus, a factor α increase in the security parameter implies a α^3 increase in the size of the modulus p. This is why, the reduction with forking lemma for Schnorr signature scheme implies that the scheme is secure only with a field modulus 8000 bits, if we consider that discrete-log problem is hard for 1000 bit modulus. In Table-2, we do not consider the schemes reported in [5, 13, 1, 6] because they are all probabilistic signature schemes. We consider the scheme in [9] for comparing with our scheme.

Scheme	Tightness		Implication on size of $\|p\|$		Size of one group element	Type
	Key	Sign	Key Size	Sign Size		
Javier [9]	NT	NT	8*320=2560†	8*320=2560†	2560	D
Ours	T	T	320	320	320	D

Table-2: Tightness Comparison with the Existing Scheme

T - Tight, NT - Not Tight (uses forking-lemma), P - Probabilistic Signature, D - Deterministic Signature, † *- The eight fold increase is due to loose reduction through forking lemma, We consider Elliptic Curve CDH is hard in 320 bits.*

Application: Aggregation of several signatures is an important computation done on several signatures in order to optimize communication, computation and storage costs. Depending on the size of the aggregated output, we refer a particular aggregation scheme as Naive, Partial or Full aggregation. By using the identity based signature scheme by Herranz [9] partial aggregation is possible. His scheme allows a more compact aggregation where the length of the resulting aggregate signature will not depend on the number of signed messages, but on the number of signers. This improvement, is considered to be a major improvement in [9] because in situations where devices have to store many signatures coming from a small set of users, the size of the aggregate signature gets compact. This is because, the key generation is probabilistic and the randomness used to compute the key can be stored by the verifier and since the signature is deterministic, there is no

randomness to be propagated with the signature and hence the aggregate signature is more compact than the aggregate signatures generated by probabilistic identity based signature schemes.

Our Contribution: Our first contribution is a novel probabilistic PKI based signature scheme (and this is of independent interest) described in section 4. PKG uses this to generate the private keys for users. The next section (section 5) contains the details of an identity based deterministic signature scheme, and again this scheme is different from all the existing ones. Our scheme does not use pairing in the generation process. Of course, in the verification process, we employ pairing computations. The significant advantage of our scheme is that it allows a tight reduction to the GDH problem. For all the schemes that are available so far, the reduction is not tight. However, we show a tight reduction of the security of our scheme to the GDH problem. Ours is the first and only system with this property. Due to this property both the key size and signature size are substantially smaller than the best previously known schemes. Since our identity based signature scheme offers tight reduction to the GDH problem, it can be used to generate more compact aggregate signatures using smaller security parameter values.

2 Preliminaries

Bilinear Pairing: Let $\mathbb{G}_1$ be an additive cyclic group generated by P, with prime order q, and $\mathbb{G}_2$ be a multiplicative cyclic group of the same order q. A bilinear pairing is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties.

- ***Bilinearity.*** For all $P, Q, R \in \mathbb{G}_1$,
  - $\hat{e}(P + Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$
  - $\hat{e}(P, Q + R) = \hat{e}(P, Q)\hat{e}(P, R)$
  - $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ [Where $a, b \in_R \mathbb{Z}_p$]
- ***Non-Degeneracy.*** There exist $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq I_{\mathbb{G}_2}$, where $I_{\mathbb{G}_2}$ is the identity element of $\mathbb{G}_2$.
- ***Computability.*** There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

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

Definition 1. *Computation Diffie-Hellman Problem (CDHP): Given $(P, aP, bP) \in \mathbb{G}_1^3$ for unknown $a, b \in \mathbb{Z}_p$, the CDH problem in $\mathbb{G}_1$ is to compute abP. The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the CDH problem in $\mathbb{G}_1$ is defined as:*

$$Adv_{\mathcal{A}}^{CDH} = Pr\left[\mathcal{A}(P, aP, bP) = abP \mid a, b \in \mathbb{Z}_p\right]$$

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

Definition 2. *Decisional Diffie-Hellman Problem (DDHP): Given $(P, aP, bP, Q) \in \mathbb{G}^4$ for unknown $a, b \in \mathbb{Z}_p$, the DDH problem in $\mathbb{G}$ is to check whether $Q \stackrel{?}{=} abP$. The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the DDH problem in $\mathbb{G}_1$ is defined as:*

$$Adv_{\mathcal{A}}^{CDH} = |Pr\left[\mathcal{A}(P, aP, bP, Q) = 1\right] - Pr\left[\mathcal{A}(P, aP, bP, abP) = 1\right] \mid a, b \in \mathbb{Z}_p|$$

The DDH Assumption is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{DDH}$ is negligibly small. Here $\mathbb{G}$ is a multiplicative group

Definition 3. *Gap Diffie-Hellman Problem (GDHP) [11][5]: We call $\mathbb{G}$ a GDH group if DDHP can be solved in polynomial time but no probabilistic algorithm can solve CDHP with non-negligible advantage within polynomial time.*

3 Identity Based Signature Scheme

In this section, we describe the generic frame work for an identity based signature scheme. The frame work of an identity based deterministic signature scheme consists of the algorithms described below, namely **Setup**, **Extract**, **Sign** and **Verify**. An identity based signature scheme is deterministic if the signature on a message by the same user is always the same.

3.1 Definition:

- ***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, params, checks whether σ is a valid signature on message m by ID. If true it outputs "$Valid$", else it outputs "$Invalid$".*

3.2 Security Model for Existential Unforgeability

An IBDS 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 various oracles provided by $\mathcal{C}$ in the following way:
  - **KeyGen oracle:** When $\mathcal{F}$ makes a query with an identity ID as input, $\mathcal{C}$ outputs D, the private key of ID to $\mathcal{F}$, provided $\mathcal{C}$ knows the private key for the queried identity.
  - **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:** $\mathcal{F}$ identifies an identity, message pair (ID_T, m^*), where
  - $\mathcal{F}$ has not queried the $KeyGen$ query on ID_T and
  - $\mathcal{F}$ has not asked the signature for the pair (ID_T, m^*).

  $\mathcal{F}$ outputs a signature σ, with ID_T as signer, and on message m^*. $\mathcal{F}$ wins the game if σ is a valid signature.

$$Adv_{\mathcal{F}}^{IBDS} = \{Pr[Verify(\sigma) = valid]\}$$

3.3 Existing Identity Based Signatures

Here, we review the most important identity based signature schemes.

Scheme	Master key	Private Key	Signature
Cha-Cheon [5]	$MSK = s$	$D_A = sQ_A \in \mathbb{G}_1$	$r \in_R \mathbb{Z}_q^*$
	$MPK = sP$	$Q_A = \hat{H}(ID_A) \in \mathbb{G}_1$	$U = rQ_A \in \mathbb{G}_1, h = \bar{H}(m, U) \in \mathbb{Z}_q^*$
			$V = (r + h)D_A \in \mathbb{G}_1$, $\sigma = \langle U, V \rangle$
Sakai [13]	$MSK = s$	$D_A = sQ_A \in \mathbb{G}_1$	$r \in_R \mathbb{Z}_q^*$
	$MPK = sP$	$Q_A = \hat{H}(ID_A) \in \mathbb{G}_1$	$U = rP \in \mathbb{G}_1, H = \bar{H}(m, U) \in \mathbb{G}_1$
			$V = rH + D_A \in \mathbb{G}_1$, $\sigma = \langle U, V \rangle$
Barreto [1]	$MSK = s$	$D_A = \frac{1}{s+q_A} P \in \mathbb{G}_1$	$r \in_R \mathbb{Z}_q^*$
	$MPK = sP$	$q_A = \hat{H}(ID_A)$	$U = rP \in \mathbb{G}_1, h = \bar{H}(m, U) \in \mathbb{Z}_q^*$
			$V = (r + h)D_A \in \mathbb{G}_1$, $\sigma = \langle U, V \rangle$
Galindo [6]	$MSK = s$	$x_A \in_R \mathbb{Z}_q^*, X_A = x_A P$	$r \in_R \mathbb{Z}_q^*$
	$MPK = sP$	$d_A = x_A + sq_A \in \mathbb{Z}_q^*$	X_A, $U = rP, h = \bar{H}(m, U) \in \mathbb{Z}_q^*$
		$q_A = \hat{H}(ID_A, X_A)$	$V = r\bar{h} + d_A \in \mathbb{G}_1$, $\sigma = \langle X_A, U, V \rangle$
Javier [9]	$MSK = s$	$x_A \in_R \mathbb{Z}_q^*, X_A = x_A P$	X_A
	$MPK = sP$	$d_A = x_A + sq_A \in \mathbb{Z}_q^*$	$U = d_A \bar{H}(m) \in \mathbb{G}_1$
		$q_A = \hat{H}(ID_A, X_A)$	$\sigma = \langle X_A, U \rangle$

Table-3: Brief Survey of existing schemes

MSK - Master Private Key, MPK - Master Public Key, $\hat{H}, \bar{H}$ - Cryptographic hash functions

4 Basic Signature Scheme ($Basic_{Sign}$)

We now construct a fully secure public key signature scheme in the random oracle model under the GDH assumption and without using forking lemma. This is a PKI based signature scheme and this will be used by the PKG to generate the private key for the users of our identity based system.

Scheme: Let $\mathbb{G}_1$, $\mathbb{G}_2$ be cyclic prime order groups of order p, where $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ be a multiplicative group. Let $P \in_R \mathbb{G}_1$ be the generator of $\mathbb{G}_1$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear map and $H_1(.), H_2(.)$ be two cryptographic hash functions defined by,

$$H_1\text{: } \{0,1\}^{l_m} \times \mathbb{G}_1 \rightarrow \mathbb{G}_1 \text{ and } H_2\text{: } \{0,1\}^{l_m} \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p, \text{ where } l_m \text{ is length of message}$$

- **User KeyGen**: Let U_A be a user with public key $PK_A = \langle P_1, P_2 \rangle = \langle s_1 P, s_2 P \rangle$, where s_1, s_2 are random elements from $\mathbb{Z}_p$. Here, the private key of user U_A is $SK_A = \langle s_1, s_2 \rangle$.
- **Sign:** To generate the signature on message m, the user U_A executes this algorithm:
  - Pick r randomly from $\mathbb{Z}_p$.
  - Compute $Y_m = rP_2$
  - Compute $X_m = rH_1(m, Y_m)$.
  - Find $q_m = H_2(m, X_m)$.
  - Compute $d_m = q_m s_1 + r s_2 \ mod \ p$.
  - Output the signature $\sigma = \langle X_m, d_m \rangle$

  Important Note: The value Y_m is not sent along with the signature because it can be computed from the second component of σ as follows and the hash value q_m is computable by any one on knowing m and X_m:

  $Y_m = d_m P - q_m P_1 = q_m P_1 + rP_2 - q_m P_1 = rP_2$

  The tuple $\langle P_2, H_1(m, Y_m), Y_m, X_m \rangle = \langle P_2, H_1(m, Y_m), rP_2, rH_1(m, Y_m) \rangle$ is a DH tuple. We verify if $\langle P_2, H_1(m, Y_m), Y_m, X_m \rangle$ is a DH tuple by testing $\hat{e}(X_m, P_2) \stackrel{?}{=} \hat{e}(H_1(m, Y_m), Y_m)$. This suggests the following verification algorithm.
- **Verify:**
  - On receiving $\sigma = \langle X_m, d_m \rangle$, compute $q_m = H_2(m, X_m)$ and $Y_m = d_m P - q_m P_1$.
  - Check if $\hat{e}(X_m, P_2) \stackrel{?}{=} \hat{e}(H_1(m, Y_m), Y_m)$.

  If the above check holds accept the signature as "$Valid$" else return "$Invalid$".

4.1 Security

We prove the security of the signature scheme against existential forgery under adaptive chosen-message attacks in the random oracle model. The following theorem shows that the $Basic_{Sign}$ scheme is secure and the security of the scheme follows from the GDH assumption in $(\mathbb{G}_1, \mathbb{G}_2)$

Theorem 1. *Suppose $(\mathbb{G}_1, \mathbb{G}_2)$ be a (τ, t', ε')-GDH group pair of order p. Then the $Basic_{Sign}$ signature scheme on $(\mathbb{G}_1, \mathbb{G}_2)$ is $(t, q_{Sign}, q_{H_1}, q_{H_2}, \varepsilon)$-secure against existential forgery under adaptive chosen-message attack in the random oracle model, for all t and ε, that satisfies*

$$\varepsilon \leq \varepsilon' \ and \ t \geq t' - (q_{H_1} + q_{H_2} + q_{Sign} + \mathcal{O}(1))$$

Proof: Let us assume, $\mathcal{F}$ is a forger algorithm that $(t, q_{Sign}, q_{H_1}, q_{H_2}, \varepsilon)$-breaks the $Basic_{Sign}$ signature scheme on $(\mathbb{G}_1, \mathbb{G}_2)$. We show how to construct a t'-time algorithm $\mathcal{C}$ that solves GDH on $(\mathbb{G}_1, \mathbb{G}_2)$ with probability at least ε'.

Let P be the generator of $\mathbb{G}_1$. Algorithm $\mathcal{C}$ is provided with the challenge instance $(P, aP, bP) \in \mathbb{G}_1$. The goal of $\mathcal{C}$ is to output $abP \in \mathbb{G}_1$. Algorithm $\mathcal{C}$ simulates the challenger and interacts with $\mathcal{F}$ in the following way:

- **Setup:** Challenger $\mathcal{C}$ starts by giving $\mathcal{F}$ the common reference string $(P, \mathbb{G}_1, \mathbb{G}_2)$ and the public key $(P_1 = aP, P_2 = s_2 P)$, where s_2 is chosen at random from $\mathbb{Z}_p$. The private key corresponding to the public keys $(P_1 = aP, P_2 = s_2 P)$ are (a, s_2). Note that, $\mathcal{C}$ does not know one of the private keys namely a.

– **Training Phase:** During this phase $\mathcal{F}$ has access to the following oracles:
  - *H_1 Queries:* Forger $\mathcal{F}$ is allowed to query the H_1 oracle at any time. To handle these queries $\mathcal{C}$ maintains a list which is defines as $\langle m, Y_m, h, H_m\rangle$ and we refer this list as $\mathcal{L}_1 - list$. Initially, this list is empty and will be updated as explained below. When $\mathcal{F}$ queries the oracle H_1 with $(m \in \{0,1\}^{l_m}, Y_m \in \mathbb{G}_1)$ as input, $\mathcal{C}$ responds as follows:
    * If (m, Y_m) already exists as a tuple of the form $\langle m, Y_m, h, H_m\rangle$ in $\mathcal{L}_1 - List$, then $\mathcal{C}$ responds with $H_1(m, Y_m) = H_m \in \mathbb{G}_1$.
    * Otherwise, $\mathcal{C}$ picks a new random $h \in \mathbb{Z}_p$ and sets $H_m = hbP$.
    * $\mathcal{C}$ stores the tuple $\langle m, Y_m, h, H_m\rangle$ in $\mathcal{L}_1 - List$ and responds with $H_1(m, Y_m) = H_m \in \mathbb{G}_1$.
  - *H_2 Queries:* $\mathcal{F}$ can query this oracle at any time and $\mathcal{C}$ maintains a list of tuples $\langle m, X_m, q_m\rangle$. This list is called $\mathcal{L}_2 - list$. When $\mathcal{F}$ issues a query for (m, X_m) to the H_2 oracle, $\mathcal{F}$ responds in the following way:
    * If (m, X_m) already appears in $\mathcal{L}_2 - list$ as a tuple $\langle m, X_m, q_m\rangle$, then $\mathcal{C}$ responds with $H_2(m, X_m) = q_m \in \mathbb{Z}_p$.
    * Otherwise, $\mathcal{C}$ randomly picks a new $q_m \in \mathbb{Z}_p$, stores the tuple $\langle m, X_m, q_m\rangle$ in $\mathcal{L}_2 - list$ and responds with $H_2(m, X_m) = q_m \in \mathbb{Z}_p$.
  - *Signature Queries:* When a signature query is issued by $\mathcal{F}$ for message m, $\mathcal{C}$ responds as follows:
    * $\mathcal{C}$ randomly picks new $d_m, q_m, h \in \mathbb{Z}_p$.
    * Then, $\mathcal{C}$ sets $H_m = hP$, $Y_m = d_mP - q_mP_1 \in \mathbb{G}_1$ and $X_m = \left(\frac{h}{s_2}\right) Y_m \in \mathbb{G}_1$.
    * If a tuple of the form $\langle m, X_m, q_m\rangle$ appears in the list $\mathcal{L}_1 - list$ or a tuple $\langle m, Y_m, h, H_m\rangle$ appears in the list $\mathcal{L}_2 - list$, then repeat the process by picking new set of random values $d_m, q_m, h \in \mathbb{Z}_p$.
    * $\mathcal{C}$ stores the tuple $\langle m, X_m, q_m\rangle$ in $\mathcal{L}_1 - list$ and $\langle m, Y_m, h, H_m\rangle$ in $\mathcal{L}_2 - list$.
    * $\mathcal{C}$ gives the signature $\sigma = \langle X_m, d_m\rangle$ to $\mathcal{F}$.

  Correctness: The simulated signature is valid and passes the verification test $\hat{e}(X_m, P_2) \stackrel{?}{=} \hat{e}(H_1(m, Y_m), Y_m)$. The correctness is shown below:

  $\text{LHS} = \hat{e}(X_m, P_2) = \hat{e}((\frac{h}{s_2})Y_m, P_2) = \hat{e}((\frac{h}{s_2})Y_m, s_2P)$
  $= \hat{e}(hY_m, P) = \hat{e}(Y_m, hP) = \hat{e}(Y_m, H_1(m, Y_m)) = \text{RHS}$

– **Forgery:** On getting sufficient training, algorithm $\mathcal{F}$ produces a message-signature pair $(m^*, \sigma^* = \langle X^*_m, d^*_m\rangle)$ such that σ^* is not the output generated by sign oracle for message m^* and σ^* is valid. Now, $\mathcal{C}$ may compute the solution to the hard problem as given below. For the system the public key is $(P_1, P_2) = (aP, s_2P)$ and secret key (a, s_2). Here a is not known to $\mathcal{C}$ but s_2 is chosen by $\mathcal{C}$. For the message m^*, the forgery $({X_m}^*, {d_m}^*)$ is produced by forger $\mathcal{F}$ and given to $\mathcal{C}$.
$({X_m}^*, {d_m}^*)$ is a valid forgery implies that,

$$\hat{e}({X_m}^*, P_2) = \hat{e}(H_1(m^*, {Y_m}^*)) \tag{1}$$

where,

$${Y_m}^* = {d_m}^*P - {q_m}^*P_1 \tag{2}$$

and

$${q_m}^* = H_2(m^*, {X_m}^*) \tag{3}$$

Note that $\mathcal{C}$ can compute ${q_m}^*$ and ${Y_m}^*$ and using this $\mathcal{C}$ can obtain the value,

$$H_1(m^*, {Y_m}^*) = h^*bP \tag{4}$$

for same h^* known to $\mathcal{C}$.
From equation (1), it follows that there exists a value r^* satisfying,

$${X_m}^* = r^*H_1(m^*, {Y_m}^*) \tag{5}$$

and

$${Y_m}^* = r^*P_2 \tag{6}$$

Using equation (4) and (5), $\mathcal{C}$ concludes that,

$${X_m}^* = r^*, h^*bP \tag{7}$$

Note that in equation(7), $\mathcal{C}$ knows the value h^*, but $\mathcal{C}$ does not know the values r^* and b. However, $\mathcal{C}$ knows bPas bP is the input of the GDH problem that $\mathcal{C}$ wants to solve.
Using equation(2) and (6), $\mathcal{C}$ concludes that ,

$$r^*P_2 = {d_m}^*P - {q_m}^*P_1$$

That is,

$$r^*s_2P = {d_m}^*P - {q_m}^*aP$$

which implies that,

$${d_m}^* = a{q_m}^* + s_2r^* \tag{8}$$

Now, $\mathcal{C}$ computes,

$$\delta = \frac{1}{{q_m}^*}\left({d_m}^*(bP) - \frac{s_2}{h^*}{X_m}^*\right) \tag{9}$$

Observe that δ can be computed by $\mathcal{C}$ because $\mathcal{C}$ knows the values ${q_m}^*$, h^*, s_2 and bP, and ${d_m}^*$ and ${X_m}^*$ are the components of the forgery produces by forger $\mathcal{F}$ and given to $\mathcal{C}$.
In fact,

$$\begin{aligned}\delta &= \frac{1}{{q_m}^*}\left({d_m}^*(bP) - \frac{s_2}{h^*}{X_m}^*\right)\\ &= \frac{1}{{q_m}^*}\left(\left(a{q_m}^* + s_2r^*\right)(bP) - \frac{s_2}{h^*}r^*h^*(bP)\right) \text{ from (7) and (8).}\\ &= \frac{1}{{q_m}^*}\left(a{q_m}^*(bP)\right)\\ &= abP.\end{aligned}$$

Thus, we have shown that there is no forgery possible in polynomial time with non negligible advantage.

Now, we have to show that $\mathcal{C}$ solves the GDH problem on $(\mathbb{G}_1, \mathbb{G}_2)$ with probability at least ε'. Note that, in this simulation there is almost no aborting scenario for training phase and forgery phase. Hence $\mathcal{C}$ solving the GDH problem happens almost with the same advantage of $\mathcal{F}$.

The hard problem is solved after q_{H_1} queries to the H_1 oracle, q_{H_2} queries to the H_2 oracle and q_{Sign} sign oracle queries and getting the forged signature. The challenger has to spend $\mathcal{O}(1)$ computation to extract the solution to GDH problem from the forgery generated by the adversary. Therefore the total time t taken for solving the hard problem is given by $t \leq t' + (q_{H_1} + q_{H_2} + q_{Sign} + \mathcal{O}(1))$. If t' were polynomial, that would imply that t is also polynomial, and this is a contradiction to the assumption $(\mathbb{G}_1, \mathbb{G}_2)$ is a GDH group pair. □

5 Identity Based Deterministic Signature Scheme (Det-IBS)

Inspired by the impact of tightness of security reduction for a signature scheme, we present the first identity based deterministic signature scheme with tight security reduction to GDH problem. The only identity based deterministic signature by Herranz [9], employs Schnorr signature scheme for generating the private key of the user and uses BLS short signature scheme for producing signature on the message by the user. This system was shown to be secure under GDH problem on $(\mathbb{G}_1, \mathbb{G}_2)$. The reduction given for the scheme in [9] use forking lemma and hence considered to be loose. We present a signature that works on GDH group pair $(\mathbb{G}_1, \mathbb{G}_2)$. We prove the security of the scheme in the random oracle model and show how it leads to a tight reduction. The scheme uses $Basic_{Sign}$ signature scheme for generating the private key of users and BLS short signature for generating the signature on message.

Scheme: Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a (t, ε)-GDH group pair with same prime order p and $\hat{e}$ be a bilinear map defined by $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The signature scheme comprises of setup, extract, sign and verify algorithms. The scheme makes use of three cryptographic hash functions $H_1 : \{0, 1\}^{l_1} \times \mathbb{G}_1 \rightarrow \mathbb{G}_1$, $H_2 : \{0, 1\}^{l_1} \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p$ and $H_3 : \{0, 1\}^{l_m+1} \times \{0, 1\}^{l_1} \rightarrow \mathbb{G}_1$, where l_1 is the size of the identity string and l_m is the size of the message.

– **Det-IBS.Setup:** PKG picks at random $s_1, s_2 \in \mathbb{Z}_p$, and $P \in \mathbb{G}_1$, sets $P_1 = s_1P \in \mathbb{G}_1$, and $P_2 = s_2P \in \mathbb{G}_1$. The master public key is (P_1, P_2). The master private key is (s_1, s_2).

– **Det-IBS.Extract:** Given the master private (s_1, s_2), and the user identity $ID_A \in \{0,1\}^{l_1}$, perform the following:

- Pick $r_A \in_R \mathbb{Z}_p$.
- Compute $Y_A = r_A P_2 \in \mathbb{G}_1$.
- Find $H_A = H_1(ID_A, Y_A)$ and set $X_A = r_A H_A \in \mathbb{G}_1$.
- Compute $d_A = s_1 q_A + s_2 r_A \, mod \, p$, where $q_A = H_2(ID_A, X_A)$.
- The private key is $D_A = \langle d_A, X_A, Y_A \rangle$.

Note: However, in our identity based deterministic signature scheme, we provide Y_A explicitly along with the private key How ever Y_A is computable by the user with identity ID_A on knowing d_A and X_A.

– **Det-IBS.Sign:** Given a message m, user identity $ID_A \in \{0,1\}^{l_1}$ and the user private key $D_A = \langle d_A \in \mathbb{Z}_p, X_A \in \mathbb{G}_1, Y_A \in \mathbb{G}_1 \rangle$, choose $\lambda \in_R \{0,1\}$, compute $H_m = H_3(m \| \lambda, ID_A)$ and $V = d_A H_m$. The signature is $\sigma = \langle V, \lambda, X_A, Y_A \rangle \in \mathbb{G}_1 \times \{0,1\} \times \mathbb{G}_1^2$.

Note: λ can be generated using a pseudo-random function with the identity ID_A, message m and the private key of the user as input. This helps to preserve the determinism because each time a message is signed by a user, the bit λ is going to be the same. (Goh et al. [8]).

– **Det-IBS.Verify:** Given an identity $ID_A \in \{0,1\}^{l_1}$, a message $m \in \{0,1\}^{l_m}$, and a signature $\sigma = \langle V \in \mathbb{G}_1, \lambda \in \{0,1\}, X_A \in \mathbb{G}_1, Y_A \in \mathbb{G}_1 \rangle$, compute $q_A = H_2(ID_A, X_A)$, $H_A = H_1(ID_A, Y_A)$, and $H_m = H_3(m \| \lambda, ID_A)$ and check,

$$\hat{e}(V, P) \stackrel{?}{=} \hat{e}(H_m, q_A P_1 + Y_A) \text{ ——(a)}$$
$$\hat{e}(X_A, P_2) = \hat{e}(H_A, Y_A) \text{ ——(b)}$$

If both the check passes, output "*Valid*"; if not, output "*Invalid*"

Theorem 2. *The signature scheme Det-IBS is consistent*

Proof: We need to show that, for all private key tuples , and for all messages, any signature generated by the signing algorithm verifies as a valid signature under the respective user identity. Indeed, we have for equation (a)

$$\begin{aligned} \text{LHS} = \hat{e}(V, P) &= \hat{e}(d_A H_m, P) = \hat{e}((q_A s_1 + r_A s_2) H_m, P) \\ &= \hat{e}(H_m, (q_A s_1 + r_A s_2) P) = \hat{e}(H_m, q_A P_1 + r_A P_2) \\ &= \hat{e}(H_m, q_A P_1 + Y_A) = \text{RHS} \end{aligned}$$

and also, for equation (b)

$$\text{LHS} = \hat{e}(X_A, P_2) \;\; = \hat{e}(r_A H_A, P_2) = \hat{e}(H_A, r_A P_2) = \hat{e}(H_A, Y_A) = \text{RHS}.$$

5.1 Security

We prove the security of the identity based deterministic signature scheme against existential forgery under adaptive chosen-message attacks in the random oracle model. The following theorem shows that the *Det-IBS* scheme is secure and the security of the scheme follows from GDH assumption on the groups $(\mathbb{G}_1, \mathbb{G}_2)$

Theorem 3. *Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a $(\tau_1, t_1, \varepsilon_1)$ GDH group pair of order p then the identity based deterministic signature scheme on $(\mathbb{G}_1, \mathbb{G}_2)$ is $(t_2, q_s, q_{H_1}, q_{H_2}, q_{H_3}, \varepsilon_2)$ - secure against existential forgery under an adaptive chosen message attack in random oracle model, for all t_2 and ε_2 satisfying:*

$$\varepsilon_2 \geq 2 q_{H_1} \varepsilon_1 \, and \, t_2 \leq t_1 - (q_{H_1} + q_{H_2} + q_{H_3} + 2q_s + \mathcal{O}(1))$$

Here, q_H is the total number of identities generated.

Proof: Consider $\mathcal{F}$ to be a forger that is assumed to $(t_2, q_s, q_{H_1}, q_{H_2}, q_{H_3}, \varepsilon_2)$ - break the signature scheme. We show how to construct an algorithm $\mathcal{C}$ that solves GDHP on $(\mathbb{G}_1, \mathbb{G}_2)$ with probability at least ε_1. This will contradict the fact that $(\mathbb{G}_1, \mathbb{G}_2)$ is a GDH group pair.

For doing this, let us assume P be the generator of $\mathbb{G}_1$ and $(P, aP, bP) \in \mathbb{G}_1^3$ be the GDH problem instance given to $\mathcal{C}$. The goal of $\mathcal{C}$ is to find $abP \in \mathbb{G}_1$. $\mathcal{C}$ simulates the challenger and interacts with $\mathcal{F}$ as defined in the EUF-CMA game. The game is viewed as given below:

- **Setup:** $\mathcal{C}$ starts interaction with $\mathcal{F}$ by providing $P \in \mathbb{G}_1$, $P_1 = aP \in \mathbb{G}_1$ and $P_2 = s_2 P$, where $s_2 \in_R \mathbb{Z}_p$. Here, the master private key a is not known to $\mathcal{C}$. $\mathcal{C}$ also chooses $1 \leq T \leq q_H$ randomly and sets the T^{th} unique identity queried to the H_1 hash oracle as the target identity. Without loss of generality, we assume ID_T to be the target identity (not known to $\mathcal{F}$ and $\mathcal{C}$ at the start of the game.)
- **Training Phase:** $\mathcal{C}$ interacts with $\mathcal{F}$ in the following manner:

  $\underline{H_1 \text{ Oracle:}}$ $\mathcal{F}$ queries to this oracle with inputs $\langle ID_i, Y_j \rangle$. $\mathcal{C}$ maintains the list L_{H_1}, consisting of tuples of the form $\langle ID_i, Y_j, H_j, \hat{x}_j \rangle$ and responds to $\mathcal{F}$'s queries in the following way:
  - If the tuple $\langle ID_i, Y_j, H_j, \hat{x}_j \rangle$ is already available in the L_{H_1} list, retrieve and return H_j.
  - If $i \neq T$, choose $\hat{x}_j \in_R \mathbb{Z}_p$ and set $H_j = \hat{x_j} P \in \mathbb{G}_1$. Store the tuple $\langle ID_i, Y_j, H_j, \hat{x}_j \rangle$ to L_{H_1} and return H_j.
  - If $i = T$, choose $\hat{x}_j \in_R \mathbb{Z}_p$ and set $H_j = \hat{x}_j(bP)$. Store the tuple $\langle ID_i, Y_j, H_j, \hat{x}_j \rangle$ to L_{H_1} and return H_j to $\mathcal{F}$.

  $\underline{H_2 \text{ Oracle:}}$ To respond to the queries by $\mathcal{F}$, $\mathcal{C}$ maintains the list L_{H_2}, consisting of tuples of the form $\langle ID_i \in \{0,1\}^{l_1}, X_j \in \mathbb{G}_1, q_j \in \mathbb{Z}_p \rangle$. The list is initially empty. When $\mathcal{F}$ queries with (ID_i, X_j), $\mathcal{C}$ responds as follows:
  - If the tuple $\langle ID_i, X_j, q_j \rangle$ already exists in L_{H_2} list, retrieve and return q_j corresponding to (ID_i, X_j) to $\mathcal{F}$.
  - Else, pick $q_j \in_R \mathbb{Z}_p$ store (ID_i, X_j, q_j) in L_{H_2} list and return q_j to $\mathcal{F}$.

  $\underline{H_3 \text{ Oracle:}}$ The input to this oracle are $m_j \| \lambda, ID_i$, where $\lambda \in \{0,1\}$. To respond to the queries by $\mathcal{F}$, $\mathcal{C}$ maintains the list L_{H_3}, consisting of tuples of the form $\langle m_j, \lambda, ID_i, H_{j,\lambda}, y_{j,\lambda}, \star \rangle$. Here $\star$ is either '$\perp$' or '$\top$', where '$\perp$' represents $H_{j,\lambda} = y_{j,\lambda} P$ and '$\top$' represents $H_{j,\lambda} = y_{j,\lambda} bP$. $\mathcal{C}$ respond to $\mathcal{F}$ in the following way:
  - If the tuple $\langle m_j, \lambda, ID_i, H_{j,\lambda}, y_{j,\lambda}, \star \rangle$ is already there in the list L_{H_3}, retrieve and respond with the corresponding $H_{j,\lambda}$.
  - Else,
    - Pick $y_{j,0}, y_{j,1} \in_R \mathbb{Z}_p$.
    - Flip a coin $c \in \{0,1\}$.
    - Set $H_{j,c} = y_{j,c} bP$ and store the tuple $\langle m_j, c, ID_i, H_{j,c}, y_{j,c}, \top \rangle$ in list L_{H_3}.
    - Set $H_{j,\bar{c}} = y_{j,\bar{c}} P$ and store the tuple $\langle m_j, \bar{c}, ID_i, H_{j,\bar{c}}, y_{j,\bar{c}}, \perp \rangle$ in list L_{H_3}.
    - If $c = \lambda$, return $H_{j,c}$ else, return $H_{j,\bar{c}}$.

  Note that $H_{j,\lambda}$ is uniform in $\mathbb{G}_1$ and is independent of $\mathcal{F}$'s current view as required.

  $\underline{Extract \text{ Oracle:}}$ To respond to this query, $\mathcal{C}$ maintains the L_E list, consisting of tuples of the form $\langle ID_i, d_i, X_i, Y_i \rangle$. When $\mathcal{F}$ makes a query with ID_i as input, $\mathcal{C}$ checks whether $i = T$, if so *aborts*. Otherwise, $\mathcal{C}$ performs the following:
  - Choose $d_i, q_i \in_R \mathbb{Z}_p$.
  - Compute $Y_i = d_i P - q_i P_1 = (d_i - a q_i) P$.
  - If $\langle ID_i, Y_i, H_i, \hat{x}_i \rangle$ in L_{H_1} then
    - Compute $X_i = \frac{d_i \hat{x}_i}{s_2} P - \frac{q_i \hat{x}_i}{s_2} P_1 = (d_i - a q_i) \frac{1}{s_2} H_i$
    - If there is a tuple of the form $\langle ID_i, X_i, \star \rangle$ in list L_{H_2} then start the process again.
    - Store the tuple $\langle ID_i, X_i, q_i \rangle$ in L_{H_2} list, the tuple $\langle ID_i, d_i, X_i, Y_i \rangle$ in L_E list and return (d_i, X_i, Y_i) as the private key corresponding to ID_i.
  - Else,
    - Choose a random $\hat{x}_i \in \mathbb{Z}_p$.
    - Compute $H_i = \hat{x}_i P$
    - Compute $X_i = \frac{d_i \hat{x}_i}{s_2} P - \frac{q_i \hat{x}_i}{s_2} P_1 = (d_i - a q_i) \frac{1}{s_2} H_i$
    - If there is a tuple of the form $\langle ID_i, X_i, \star \rangle$ in list L_{H_2} then start the process again.

* Store the tuple $\langle ID_i, Y_i, H_i, \hat{x}_i \rangle$ in L_{H_1}, the tuple $\langle ID_i, X_i, q_i \rangle$ in L_{H_2} list, the tuple $\langle ID_i, d_i, X_i, Y_i \rangle$ in L_E list and return (d_i, X_i, Y_i) as the private key corresponding to ID_i.

Without loss of generality, we assume that any identity is queried only once to this oracle.

Signature Oracle: Let (m_j, ID_i) be the message identity pair for which $\mathcal{F}$ request the signature. $\mathcal{C}$ performs the following:

- If there are no entries corresponding to $m_j \| 0, ID_i$ and $m_j \| 1, ID_i$ in the list L_{H_3}, then query the H_3 oracle with input $m_j \| 0, ID_i$.
- Retrieve the entries corresponding to $m_j \| 0, ID_i$ and $m_j \| 1, ID_i$ in list L_{H_3}. Let the two tuples retrieved be $\langle m_j, \lambda, ID_i, H_{j,\lambda}, y_{j,\lambda}, \perp \rangle$ and $\langle m_j, \overline{\lambda}, ID_i, H_{j,\overline{\lambda}}, y_{j,\overline{\lambda}}, \top \rangle$ (Note that according to the definition of the H_3 oracle, one of the entries will have $\perp$ and the other one will have $\top$ as the last entry in the tuple.). Pick the entry corresponding to $\perp$, here $\langle m_j, \lambda, ID_i, H_{j,\lambda}, y_{j,\lambda}, \perp \rangle$ is the required tuple.
- If $i \neq T$, then perform the following:
  * Set $H_m = H_{j,\lambda}$
  * Set $V = d_i H_m$. (Note that $\mathcal{C}$ knows the private key d_i corresponding to ID_i)
- If $i = T$, then perform the following:
  * Set $y_j = y_{j,\lambda}$
  * Set $V = y_j(q_T P_1 + x_T P_2)$. (Notice that $V = y_j(q_T P_1 + x_T P_2) = (q_T s_1 + x_T s_2) y_j P = d_T y_j P = d_T H_m$ and, $X_T = x_T H_T$ and $Y_T = x_T P_2$.)
- Return $\sigma = \langle V, X_i, Y_i, \lambda \rangle$ as the signature on the message m_j.

This is indeed a valid signature and satisfies the verification algorithm. The details are given below,

Correctness of the test $\hat{e}(V, P) \stackrel{?}{=} \hat{e}(H_m, q_T P_1 + Y_T)$

$$\begin{aligned} \text{LHS} = \hat{e}(V, P) &= \hat{e}(y_j(q_T P_1 + x_T P_2), P) = \hat{e}((q_T s_1 + x_T s_2) y_j P, P) \\ &= \hat{e}(y_j P, (q_T s_1 + x_T s_2) P) \\ &= \hat{e}(H_m, (q_T s_1 + r_T s_2) P) = \hat{e}(H_m, q_T P_1 + r_T P_2) \\ &= \hat{e}(H_m, q_T P_1 + Y_T) = \text{RHS of equation (a)} \end{aligned}$$

Correctness of the test $\hat{e}(X_A, P_2) = \hat{e}(H_A, Y_A)$

$$\text{LHS} = \hat{e}(X_T, P_2) = \hat{e}(x_T H_T, P_2) = \hat{e}(H_T, x_T P_2) = \hat{e}(H_T, Y_T) = \text{RHS}.$$

– **Forgery:** Eventually, after getting enough training, $\mathcal{F}$ produces a forgery $m^*, ID_S, \sigma^* = (V, X_S, Y_S, \lambda)$. $\mathcal{C}$ *aborts* if any of the following is true:

- $S \neq T$ (i.e., ID_S is not the target identity set by the challenger).
- The last field of the tuple corresponding to $m^* \| \lambda, ID_S$ in list L_{H_3} is $\perp$ (i.e., $\langle m^*, \lambda, ID_S, H_{j,\lambda}, y_{j,\lambda}, \perp \rangle \in L_{H_3}$).
- σ^* corresponding to m^* is illegal. (Since it is a deterministic signature, m^* should not be queried to the sign oracle with ID_S as the signer.)

Otherwise, $\mathcal{C}$ does the following:

- Find $q_S = H_2(ID_T, X_S)$, $H_S = H_1(ID_T, Y_S)$.
- Retrieve $\hat{x}_S$ corresponding to $\langle ID_T, Y_S, H_S, \hat{x}_S \rangle$ in the list L_{H_1}.
- Compute $\Delta = [q_S^{-1} y_S^{-1} (V - s_2 y_S \hat{x}_S^{-1} X_S)] = abP$.

Note that $\mathcal{C}$ can solve the GDH problem instance irrespective of X_S and Y_S, that is $X_S = X_T$ or $X_S \neq X_T$ and $Y_S = Y_T$ or $Y_S \neq Y_T$

Lemma 1. *Let $\langle m^*, \lambda, ID_S, H_{j,\lambda}, y_{j,\lambda}, \top \rangle$ be the tuple in the list L_{H_3} corresponding to $m^* \| \lambda, ID_S$ and $y_S = y_{j,\lambda}$. If (ID_S, σ^*) is a valid forgery on m^* then $q_S^{-1} y_S^{-1} (V - s_2 y_S \hat{x}_S^{-1} X_S) = abP$ with $P_1 = aP$, $P_2 = s_2 P$, $H_1(ID_S, Y_S) = \hat{x}_S bP$ and $H_{m^*} = H_3(m^*, ID_S) = y_S bP$.*

Proof: The proof is straight forward and is given below:

$$\begin{aligned} LHS &= q_S^{-1} y_S^{-1} (V - s_2 y_S \hat{x}_S^{-1} X_S) \\ &= q_S^{-1} y_S^{-1} (d_S H_{m^*} - s_2 y_S \hat{x}_S^{-1} X_S) \\ &= q_S^{-1} y_S^{-1} ((x_S s_2 + a q_S) H_{m^*} - s_2 y_S \hat{x}_S^{-1} x_S \hat{x}_S bP) \\ &= q_S^{-1} y_S^{-1} (x_S s_2 y_S bP + a q_S y_S bP - s_2 y_S x_S bP) \\ &= q_S^{-1} y_S^{-1} (q_S y_S abP) = abP = RHS \end{aligned}$$

□

Now, we show how $\mathcal{C}$ solves the GDH instance (P, aP, bP) with probability at least ε_1. For showing this we have to analyze the probability related to the following events:

- $E_1 : \mathcal{C}$ does not abort as a result of Extract query
- $E_2 : \mathcal{F}$ generates a valid message - signature forgery (m^*, σ^*) for $ID_S = ID_T$.
- E_3 : This event occurs for $m^* \| \lambda, ID_S$ such that the last field of the tuple corresponding to $m^* \| \lambda, ID_S$ in list L_{H_3} is $\top$ (i.e., $\langle m^*, \lambda, ID_S, H_{j,\lambda}, y_{j,\lambda}, \top \rangle \in L_{H_3}$).

Let q_{H_1} and q_E denote the number or queries made to the H_1 and Extract oracles. The probability of the above events to occur is discussed below:

- Probability of $\mathcal{C}$ aborting during an extract query is $\frac{1}{q_{H_1}}$. There are totally q_E extract queries. Thus the probability that $\mathcal{C}$ does not abort in any of the extract queries is $1 - \frac{q_E}{q_{H_1}}$ (i.e., $Pr[E_1] = \left(1 - \frac{q_E}{q_{H_1}}\right)$)
- There are totally $q_{H_1} - q_E$ identities are the eligible entities for being a valid ID_S and thus $ID_S = ID_T$ happens with probability $\frac{1}{q_{H_1} - q_E}$ (i.e., $Pr[E_2] = \frac{1}{q_{H_1} - q_E}$)
- Assuming E_2 has happened, the probability that the message $m^* \| \lambda$ being a fruitful instance (i.e., $\langle m^*, \lambda, ID_S, H_{j,\lambda}, y_{j,\lambda}, \top \rangle \in L_{H_3}$) is $\frac{1}{2(q_{H_1} - q_E)}$ (i.e.,$Pr[E_3 | E_2] = \frac{1}{2(q_{H_1} - q_E)}$). Note that every message corresponding to ID_S carries the hard problem instance bP with probability $\frac{1}{2}$.

Now, the probability of $\mathcal{C}$ solving the GDH is $\varepsilon_1 \leq \varepsilon_2 \left(\frac{1}{2(q_{H_1} - q_E)}\right) \left(1 - \frac{q_E}{q_{H_1}}\right) = \varepsilon_2 \left(\frac{1}{2q_{H_1}}\right)$.

Important Remark: The generic method given by Bellare et al. [2] to construct an identity based signature scheme is to use certificate for the public key PK and the identity ID of the user, and then use the certificate and the private key SK corresponding to the public key PK for signing the message. Using this approach, to construct a deterministic identity based signature we require two invocations of BLS signature [4]. The first application of BLS signature is by the PKG to sign ID and PK, and the second BLS signature is by the user to sign the message m using SK. Let the scheme described above be denoted as Γ. While proving the security of the scheme Γ, the advantage of $\mathcal{C}$ in solving the GDH problem is given as $\varepsilon_2 \leq \varepsilon_1 \frac{1}{q_{H_1}^2}$, where ε_1 is the advantage of the forger in breaking the signature scheme Γ. This bound can be further improved with the help of Goh et al.'s [8] technique which leads to a boost in the advantage, which is $\varepsilon_2 \leq \varepsilon_1 \frac{1}{2q_{H_1}}$. Thus the security bounds for the scheme Γ and for our new scheme are equivalent, our scheme achieves this bound without certificates. Also, the size of the signature generated are the same. The signature size of both the schemes will be $\mathbb{G}^3 + |ID|$.

6 Application to Efficient Signature Aggregation

The aggregate signatures generated using our identity based deterministic signature scheme produces an aggregate signature similar to the aggregate signatures described in [9]. The size of the aggregate signature depends on the number of distinct signers and not the number of messages signed. The scheme in [9] uses multiple forking-lemma and hence the size of the security parameter should be increased to achieve sufficient security. Since our scheme does not require forking-lemma in the reduction, the security parameter need not be blown-up as in [9] and as a result we have signature schemes with small size. The details of our aggregate signature scheme follows:

The Setup, Extract, Sign algorithm are the same as Det-IBS.Setup, Det-IBS.Extract and Det-IBS.Sign algorithms. The algorithms aggregate sign and aggregate verify are explained below:

- **Det-IBS.AggSign:** Given n signatures $\sigma_1 = \langle V_1, X_1, Y_1 \rangle, \ldots, \sigma_n = \langle V_n, X_n, Y_n \rangle$ on n messages $m_1, \ldots, m_n$ by users with identities $ID_1, \ldots, ID_t$, where $t < n$ and a list $\mathcal{L}$ which provides the details about which message is signed by whom, the aggregate signature σ_{Agg} is computed as follows:
  - Computes $V_{Agg} = \sum_{i=1}^{n} V_i$ and sets $\sigma_{Agg} = \langle V_{Agg}, X_1, \ldots, X_t, Y_1, \ldots, Y_t \rangle$

– **Det-IBS.AggVerify:** Given an aggregate signature σ_{Agg} on n messages $m_1, \ldots, m_n$ by users with identities $ID_1, \ldots, ID_t$, where $t < n$ and a list $\mathcal{L}$, σ_{Agg} is verified as follows:

Perform the following checks:

$$\hat{e}(V_{Agg}, P) \stackrel{?}{=} \prod_{i=1}^{n} (\hat{e}(H_3(m_i, ID_i), q_i P_1 + Y_i)) \text{ ——(3)}$$

$$\hat{e}(\sum_{j=1}^{t} X_j, P_2) = \prod_{j=1}^{t} \hat{e}(H_1(ID_j, Y_j), Y_j) \text{ ——(4)}$$

If the checks in (3) and (4) pass, output "$Valid$"; if not, output "$Invalid$"

Note that in the first verification, if a signer has signed more than one message (This will be documented in the list $\mathcal{L}$), the corresponding Y_i and identity ID_i will be reused along with the corresponding message m_i. The second verification needs to be done only for t values as the number of distinct signers is only t. If a single signer, say U_A with identity ID_A signs more than one message, our aggregate signature scheme can be used to generate very efficient aggregate signatures with only three group elements namely, V_{Agg}, X_A and Y_A. This optimizes the storage and communication complexity of signatures generated and communicated by a single user. This cannot be achieved by any of the probabilistic identity based aggregate signature schemes.

We argue the proof of security of the aggregate signature scheme informally. Consider the forger sends an aggregate signature (with one of the signatures that is aggregated as a signature by the target identity and the sign oracle was not queried by the adversary with the corresponding message with the target identity as the signer), as the forgery in the EUF-CMA (Existential Unforgeability) game. The challenger can remove all other signatures except the one corresponding to the target identity from the aggregate signature by generating them using the values the challenger has obtained from the random oracle lists. Now, the resulting signature is a valid individual signature by the target identity on the target message and that will be a forgery to the basic identity based signature scheme. This is a contradiction since the basic identity based signature scheme is EUF-CMA secure the aggregate signature is also secure.

7 Conclusion

In this paper, we have designed an identity based deterministic signature scheme that has tight reduction to GDH problem. Our scheme is completely different from all the existing schemes. The PKG uses a novel PKI based signature scheme to generate the private keys for users and the PKI based signature scheme itself is of independent interest. We have also proposed a novel scheme for the users to generate signed documents. Both the schemes allow tight reductions and this results in substantially smaller keys and signatures than the ones of the only other identity based deterministic signature scheme by Javier Herranz [9]. Improving the tightness by avoiding the abort scenario during the extract phase in the deterministic identity based signature scheme is considered to be an open challenge and an interesting direction to proceed.

Acknowledgement: We would like to thank the anonymous referees of IWSEC-2011 for their insightful remarks, which helped in improving our paper greatly. Special thanks goes to Prof. Willy Susilo for shephard heading our paper and helping in improving the paper.

References

1. 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.
2. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. *Journal of Cryptology*, vol.22(no.1):1–61, 2009.
3. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT-1996*, Lecture Notes in Computer Science, pages 399–416. Springer, 1996.
4. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, vol.17(no.4):297–319, 2004.

5. 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.
6. 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*, Lecture Notes in Computer Science 5580, pages 135–148, 2009.
7. Eu-Jin Goh and Stanislaw Jarecki. A signature scheme as secure as the diffie-hellman problem. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 401–415. Springer, 2003.
8. 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.
9. Javier Herranz. Deterministic identity-based signatures for partial aggregation. *The Computer Journal*, vol-49(no-3):322–330, 2006.
10. Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. *Journal of Cryptology*, Vol.15(No.1):1–18, 2002.
11. Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2001.
12. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *EUROCRYPT - 1996*, pages 387–398, 1996.
13. 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.
14. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.
15. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO - 84*, Lecture Notes in Computer Science, pages 47–53. Springer, 1984.
16. 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.