Efficient Construction of Identity Based Signcryption Schemes from Identity Based Encryption and Signature Schemes*

Sumit Kumar Pandey and Rana Barua
Indian Statistical Institute
Kolkata -700108
{sumit_r,rana}@isical.ac.in

Abstract

In this paper, we show how to construct an Identity Based Signcryption Scheme (IBSC) using an Identity Based Encryption (IBE) and an Identity Based Signature (IBS) schemes. This we obtain by first extending the An-Dodis-Rabin construction to the Identity Based setting and then instantiating. We then further modify the construction to obtain an efficient construction. We show that the security of the IBSC scheme–indistinguishability as well as unforgeablity–is derived from the security of the underlying IBE and IBS schemes. Moreover, we show that under mild (reasonable) assumptions, the scheme is both space and time efficient compared to the Sign-then-Encrypt approach.

Keywords: Identity Based Encryption, Identity Based Signature, Identity Based Signcryption

1 Introduction

In order to simplify key management, Shamir proposed an Identity Based Cryptosystem [1] in 1984. In this cryptosystem, unambiguous identity of a user(such as email address, social security number etc.) is used as a public key and the secret key corresponding to a user is issued by a third party called the Private Key Generator (PKG). Since 1984, although Shamir proposed the first Identity Based Signature (IBS) scheme in his proposal of Identity Based Cryptosystem, there were several proposals for Identity Based Encryption (IBE) schemes. However none fulfilled the demands posed by IBE until 2000 when Sakai-Ohgishi-Kashara [10]and 2001 when Boneh and Franklin [4] proposed an IBE scheme from bilinear pairing on Elliptic Curves. In 2001 again, Cocks [3] proposed an IBE scheme based on quadratic residuosity problem modulo an RSA composite modulus.

In 1997, Zheng[14] proposed a new primitive *viz* signcryption in which encryption and signature are done simultaneously at a much lower computational cost and communication overhead than the Sign-then-Encrypt approach. The scheme in [14] was not formally proved to be secure since no formal notion of security was proposed then. It was only in PKC 2002 that Baek, Steinfeld and Zheng [6] introduced a formal notion of security for signcryption.

Since the introduction of the primitive, several schemes have been proposed. Very recently, Matsuda-Matsuura-Schuldt [12] gave several simple but efficient contructions of signcryption schemes using existing primitives. In one of the constructions, they introduced the notion of *signcryption composable* and show how, in this case, a signature scheme and an encryption scheme can be combined to achieve higher efficiency than a simple composition. (Also, [12] gives a nice account of some prevous work on signcryption and has an extensive bibliography.)

*Partially supported by Department of Information Technology, Government of India

In this paper, we consider an efficient construction of an *Identity Based* Signcryption scheme. We will show how to construct an Identity Based Signcryption (IBSC) scheme using any Identity Based Encryption (IBE) scheme and Identity Based Signature (IBS) scheme. Our construction differs from those of [12] in the sense that we do not use the sign-then-encrypt or encrypt-then-sign paradigm. In the identity based setting, [11] gives a construction to obtain an IBSC scheme from an IBE scheme and an IBS scheme. However, our construction allows signature and encryption to be done in parallel during signcryption while decryption and verification can be done in parallel during designcryption. Consequently, our construction is more efficienct than the one in [11]. Security of the resulting IBSC scheme is inherited from the security results of the underlying IBE and IBS schemes in the random oracle model.

In the public key setting, An, Dodis and Rabin [5] proposed a generic construction of Signcryption scheme using Commitment then Encryption and Signature algorithm ($\mathscr{C}t\mathscr{E}\&\mathscr{S}$) . Their construction is efficient in the sense that Encryption and Signature can be done in parallel. In this paper, we show that that their construction can easily be lifted to the Identity Based setting to yield an Identity Based Signcryption scheme. We then instantiate this with a particular commitment scheme to obtain a generic construction of an efficient ID based signcryption scheme. However, like the An-Dodis-Rabin construction, we only obtain a *generalized* IND-CCA secure IBSC scheme from an IND-ID-CCA secure IBE. In fact, we show that this scheme is not IND-CCA secure. To obtain an IND-IBSC-CCA secure Identity Based Signcryption scheme, we modify the preceding construction. We show that this modification yields an *IND-IBSC-CCA secure* signcryption scheme, provided the underlying IBE is *IND-ID-CCA secure*. Finally, we show that our construction yields an efficient identity based signcryption schemes when compared with existing ones.

2 Preliminaries

2.1 Formal Model for Commitment Scheme

Throughout this paper, by a Commitment Scheme we mean a non-interactive Commitment Scheme. A Commitment Scheme consists of three algorithms:

- **Setup :** A probabilistic polynomial time algorithm that takes security parameter as input and outputs a Commitment Key CK (possibly empty) and public parameters.
- **Commit :** A probabilistic polynomial time algorithm which takes message m and public parameters and outputs a pair (c,d), where c is the commitment and d is the decommitment.
- **Open :** A deterministic polynomial time algorithm which takes commitment and decommitment pair (c,d) as input and returns m if (c,d) is a valid pair for m, else $\perp$.

For consistency, it is required that Open(Commit(m)) = m for all message $m \in \mathscr{M}$.

2.2 Properties of Commitment

1. Hiding Property : There exists no probabilistic polynomial time adversary $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ which can distinguish the commitment to any two messages of its choice with non-negligible probability. Formally,

$$\Pr[b = \hat{b} | CK \leftarrow Setup(1^k), (m_0, m_1, st) \leftarrow \mathscr{A}_1(CK), b \overset{\mathscr{R}}{\leftarrow} \{0,1\}, (c,d) \leftarrow Commit_{CK}(m_b), \bar{b} \leftarrow \mathscr{A}_2(c, st)] = \frac{1}{2} + \varepsilon$$

where ε is a negligible quantity.

2. Binding Property : There exists no probabilistic polynomial time adversary $\mathscr{A}$, having knowledge of CK, can come up with (c,d,d') such that (c,d) and (c,d') are valid commitment pairs for m and m' but $m \neq m'$.

3. Relaxed Binding Property : There exists no probabilistic polynomial time adversary $\mathscr{A}$, having knowledge of CK, can come up with a message m and $\mathscr{A}(c,d,CK)$ produces with non-negligible probability, a value d' for the output of $Commit(m)$, say (c,d), such that (c,d') is a valid commitment to some $m \neq m'$. Namely, $\mathscr{A}$ cannot find a collision using a randomly generated $c(m)$, even for m of its choice.

2.3 Formal Models for Identity Based Encryption

An IBE scheme consists of four algorithms which are the following :

- **Setup :** A probabilistic polynomial time algorithm run by a private key generator (PKG) that takes security parameter as input and outputs a master secret key *msk* and public parameters *Params*.
- **KeyGen :** A probabilistic polynomial time algorithm run by PKG which takes master secret key *msk*, identity *ID* and public parameters *Params* as input and outputs a secret key sk_{ID} associated to the identity *ID*.
- **Encrypt :** A probabilistic polynomial time algorithm that takes a message m, the recipient identity ID_{Rec}, and public parameters *Params* and outputs a ciphertext $C = Encrypt(m, ID_{Rec}, Params)$.
- **Decrypt :** A deterministic polynomial time algorithm that takes a ciphertext C, recipient identity ID_{Rec}, recipient secret key $sk_{ID_{Rec}}$ and public parameters *Params* as input and outputs m if C is the valid ciphertext of message m else $\perp$.

2.4 Security Notion of IBE scheme

An IBE scheme is said to be *indistinguishable against adaptively chosen-ciphertext* (**IND-ID-CCA**) secure if no probabilistic polynomial time algorithm adversary $\mathscr{A}$ has a non-negligible advantage in the following game.

- The Challenger runs the Setup algorithm on input of a security parameter and gives the public parameters *Params* to adversary $\mathscr{A}$.
- Find Stage : $\mathscr{A}$ starts probing the following oracles :
  - KeyGen : returns secret keys for arbitrary identities.
  - Decryption : given an identity ID_R and a ciphertext C, it generates the receiver's secret key $sk_{ID_R} = KeyGen(ID_R)$ and returns either plaintext m if C is the valid encryption of m else $\perp$.

  $\mathscr{A}$ can submit her queries adaptively in the sense that each query may depend on the previous queries. Once $\mathscr{A}$ decides that the above stage is over, she produces two equal length plaintexts m_0, m_1 and an identity ID_R for which she has not extracted the secret key while probing the KeyGen oracle. The Challenger then chooses a random bit $b \in \{0,1\}$ and returns the ciphertext $C_b = Encrypt(m_b, ID_R, Params)$ and sends it to $\mathscr{A}$.

- Guess Stage : $\mathscr{A}$ issues new queries which are same as in the Find stage but she can neither ask for the secret key of ID_R to the KeyGen oracle nor the decryption of C_b corresponding to the receiver's identity ID_R to the Decryption oracle. Finally, $\mathscr{A}$ outputs a bit b' and wins if $b' = b$. $\mathscr{A}$'s advantage is defined as $\mathscr{A}dv(\mathscr{A}) = |\Pr(b' = b) - \frac{1}{2}|$.

An IBE scheme is said to be **IND-ID-gCCA** secure if no probabilistic polynomial time algorithm adversary has a non-negligible advantage in the game which is the same as that of IND-ID-CCA (described above) except that while querying the Decryption oracle, adversary is not allowed to query on ciphertext C on the same identity ID_R such that $Decryption(C_b, ID_R, sk_R, Params) = Decryption(C, ID_R, sk_R, Params)$.

2.5 Formal Models for Identity Based Signature

An IBS scheme consists of four algorithms which are the following :

- **Setup :** A probabilistic polynomial time algorithm run by a private key generator (PKG) that takes a security parameter as input and outputs a master secret key msk and public parameters $Params$.

- **KeyGen :** A probabilistic polynomial time algorithm run by PKG which takes master secret key msk, identity ID and public parameters $Params$ as input and outputs a secret key sk_{ID} associated to the identity ID.

- **Sign :** A probabilistic polynomial time algorithm that takes a message m, the sender's identity & secret key, ID_{Sen} & $sk_{ID_{sen}}$ and public parameters $Params$ and outputs a signature $\sigma = Sign(m, ID_{Sen}, sk_{ID_{Sen}}, Params)$.

- **Verify :** A deterministic polynomial time algorithm that takes a message m, a signature σ and sender's identity ID_{Sen} as input and outputs *accept* if σ is a valid signature of the message m, else outputs *reject*.

2.6 Security Notion of IBS scheme

An IBS scheme is said to be *strongly unforgeable against chosen message attack* (**SUF-ID-CMA**) if no probabilistic polynomial time adversary has a non-negligible advantage in the following game.

- The Challenger runs the Setup algorithm on input of a security parameter and gives the public parameters $Params$ to adversary $\mathscr{A}$.

- $\mathscr{A}$ starts probing the following oracles :

  - KeyGen : returns secret keys for arbitrary identities.
  - Sign : given an identity ID_S and a message m, it generates the secret key $sk_{ID_S} = KeyGen(ID_S)$ and returns a signature σ on the message m associated to the identity ID_S.

  Once $\mathscr{A}$ decides that this stage is over, $\mathscr{A}$ produces a triple (m^*, σ^*, ID_S^*) where a secret key corresponding to ID_S^* has not been extracted before and such that (m^*, σ^*, ID_S^*) was never obtained from the Sign oracle. $\mathscr{A}$ wins the game if σ^* is a valid signature on message m^* associated to the identity ID_S^*. $\mathscr{A}$'s advantage is defined as the probability of winning the above game.

2.7 Formal Models for Identity Based Signcryption

An Identity Based Signcryption **IBSC** scheme consists of the following four algorithms.

- **Setup :** A probabilistic polynomial time algorithm run by a private key generator (PKG) that takes security parameter as input and outputs a master secret key *msk* and public parameters *Params* that include a system wide public key.
- **KeyGen :** A probabilistic polynomial time algorithm run by the PKG which takes master secret key *msk*, identity *ID* and public parameters *Params* as input and outputs a secret key sk_{ID} associated to the identity *ID*.
- **Signcryption :** A probabilistic polynomial time algorithm that takes a message m, the recipient identity ID_{Rec}, sender's identity ID_{Sen}, sender's secret key $sk_{ID_{Sen}}$ and public parameters *Params* as input and outputs a ciphertext $C = Signcryption(m, ID_{Rec}, ID_{Sen}, sk_{ID_{Sen}}, Params)$.
- **Designcryption :** A deterministic polynomial time algorithm that takes a ciphertext C, the recipient's identity ID_{Rec}, sender's identity ID_{Sen}, receiver's secret key $sk_{ID_{Rec}}$ and public parameters *Params* as input and outputs either a message

  $$m = Designcryption(C, ID_{Rec}, ID_{Sen}, sk_{ID_{Rec}}, Params)$$

  if C is a valid ciphertext of m or an error symbol $\perp$.

For consistency, it is required that if

$$C = Signcryption(m, ID_{Rec}, ID_{Sen}, sk_{ID_{Sen}}, Params)$$

then the output of the $Designcryption(C, ID_{Rec}, ID_{Sen}, sk_{ID_{Rec}}, Params)$ should be m (and sometimes an additional information that allows receiver to convince a third party that the plaintext actually emanated from the sender).

2.8 Security Notion of IBSC scheme

There are several models of security. We shall consider the strongest notion of confidentiality and unforgeability below.

2.8.1 Confidentiality

An IBSC is said to be indistinguishable against chosen ciphertext attack (**IND-IBSC-CCA**) if no probabilistic poynomial time adversary has a non-negligible advantage in the following game between the adversary and a challenger.

- The Challenger runs the Setup algorithm on input of security parameters and sends the public parameters, *Params*, to the adversary $\mathscr{A}$.
- Find Stage : $\mathscr{A}$ starts probing the following oracles:
  - KeyGen : returns secret key for arbitrary identities.
  - Signcryption : given a pair of identities ID_{Rec}, ID_{Sen} and a plaintext m, it returns an encryption under the receiver's identity ID_{Rec} of the message m signed in the name of the sender ID_{Sen}.

– Designcryption : given a pair of identities (ID_{Rec}, ID_{Sen}) and a ciphertext C, it generates the receiver's secret key $sk_{ID_{Rec}} = KeyGen(ID_{Rec})$ and returns either a pair (m,s) made of a plaintext m and a transferable authenticating information for the sender's identity ID_{Sen} or the $\perp$ symbol if C does not properly decrypt under the secret key $sk_{ID_{Rec}}$

Once $\mathscr{A}$ decides that this stage is over, it produces two plaintexts m_0, m_1 and identities ID^*_{Rec} and ID^*_{Sen} and sends to the challenger. It is required that $\mathscr{A}$ should not have extracted the secret key of ID^*_{Rec} before. Challenger then returns $C_b = Signcryption(m_b, ID^*_{Rec}, ID^*_{Sen}, sk_{ID^*_{Sen}}, Params)$ to $\mathscr{A}$ for a random $b \in \{0,1\}$.

- Guess Stage : $\mathscr{A}$ issues new queries. This time $\mathscr{A}$ neither may issue a secret key extraction query on ID^*_{Rec} to the KeyGen oracle nor can issue a designcryption query on C_b corresponding to receiver's identity ID^*_{Rec} and sender's identity ID^*_{Sen} to the Designcryption oracle. Finally, $\mathscr{A}$ outputs a bit b' and wins if $b' = b$.

$\mathscr{A}'s$ advantage is defined as $Adv(\mathscr{A}) = |\Pr(b' = b) - \frac{1}{2}|$.

Outsider security, a weaker notion, does not allow the attacker to extract the secret key of ID^*_{Sen} at any time.

We define an IBSC scheme to be **IND-IBSC-gCCA** secure in the insider security model if no probabilistic polynomial time algorithm adversary has a non-negligible advantage in the game which is the same as that of IND-IBSC-CCA in the insider security model (described above) except that while querying the decryption oracle, the adversary is not allowed to query on ciphertext C on the same receiver's identity ID^*_{Rec} and some other sender's identity ID_{Sen} (may be different from ID^*_{Sen}) such that the output of the $Designcryption(C_b, ID^*_{Rec}, ID^*_{Sen}, sk_{ID^*_{Rec}}, Params)$ and the output of the $Designcryption(C, ID^*_{Rec}, ID_{Sen}, sk_{ID^*_{Rec}}, Params)$ is same.

2.8.2 Unforgeability

An IBSC scheme is said to be signature *unforgeable* against adaptive *chosen message attacks* (**ESUF-IBSC-CMA**) if no probabilistic polynomial time adversary has a non-negligible advantage in the following game.

- The Challenger runs the Setup algorithm on input of security parameters and sends the public parameters, *Params*, to the adversary $\mathscr{A}$.
- Find Stage : $\mathscr{A}$ starts probing the following oracles:
  - KeyGen : returns secret key for arbitrary identities.
  - Signcryption : given a pair of identities ID_{Rec}, ID_{Sen} and a plaintext m, it returns an encryption under the receiver's identity ID_{Rec} of the message m signed in the name of the sender ID_{Sen}.
  - Designcryption : given a pair of identities (ID_{Rec}, ID_{Sen}) and a ciphertext C, it generates the receiver's secret key $sk_{ID_{Rec}}$ = KeyGen(ID_{Rec}) and returns either a pair (m,s) made of a plaintext m and a transferable authenticating information for the sender's identity ID_{Sen} or the $\perp$ symbol if C does not properly decrypt under the secret key $sk_{ID_{Rec}}$
- Finally, $\mathscr{A}$ produces a triple $(C^*, ID^*_{Rec}, ID^*_{Sen})$ and wins the game if the secret key of ID^*_{Sen} was not extracted and if the result of Designcryption oracle on the ciphertext C^* under the secret key

corresponding to ID^*_{Rec} is a valid message-signature pair (m^*, s^*) such that no Signcryption oracle query involving m^*, ID^*_{Sen} and some receiver ID'_{Rec} (may be different for ID_{Rec}) gives a ciphertext C' whose designcryption under the secret key corresponding to ID'_{Rec} is the alleged forgery (m^*, s^*, ID^*_{Sen})

The adversary's advantage is its probability of winning the game.

Notation :

- $S_{IBE} \rightarrow$ Identity Based Encrytion Scheme.
  - $ENC.S_{IBE}(.,\dots,.) \rightarrow$ Encryption algorithm of S_{IBE}.
  - $DEC.S_{IBE}(.,\dots,.) \rightarrow$ Decryption algorithm of S_{IBE}.
- $S_{IBS} \rightarrow$ Identity Based Signature Scheme.
  - $SIG.S_{IBS}(.,\dots,.) \rightarrow$ Sign algorithm of S_{IBS}.
  - $VER.S_{IBS}(.,\dots,.) \rightarrow$ Verification algorithm of S_{IBS}.
- $\mathscr{R} \rightarrow$ Random number space.
- $\mathscr{M} \rightarrow$ Message space.
- $\mathscr{C}' \rightarrow$ Ciphertext space of S_{IBE}
- $\mathscr{S}' \rightarrow$ Signature space of S_{IBS} corresponding to message space $\mathscr{M}$.
- $Params_{IBE} \rightarrow$ Public parameters of S_{IBE}.
- $Params_{IBS} \rightarrow$ Public parameters of S_{IBS}.
- $Params \rightarrow$ Public parameters of our signcryption scheme.
- $msk_{IBE} \rightarrow$ Master secret key of S_{IBE}.
- $msk_{IBS} \rightarrow$ Master secret key of S_{IBS}.
- $msk \rightarrow$ Master secret key of our signcryption scheme.
- $KeyGen_{IBE}(.) \rightarrow$ Key generation algorithm of S_{IBE}.
- $KeyGen_{IBS}(.) \rightarrow$ Key generation algorithm of S_{IBS}.

3 Extension of An-Dodis-Rabin Construction

In [5], An-Dodis-Rabin gave a generic construction of signcryption using *Commitment*, *Encryption* and *Signature* algorithms. Their scheme is in the traditional PKC setting. It can easily been seen that their construction can be lifted to construct an Identity Based Signcryption. Using Commitment, Identity Based Encryption and Identity Based Signature in their construction, we show that similar to the construction in [5] one can obtain an Identity Based Signcryption scheme.

3.1 ID-Based An-Dodis-Rabin Construction

Let *Commit* be a Commitment scheme, S_{IBE} be an Identity Based Encryption scheme, S_{IBS} be an Identity Based Signature scheme and m be the message.

- **Setup**(1^k): The public parameter, *Params*, is $(Params_{IBE}, Params_{IBS}, Commit)$ and master secret key is (msk_{IBE}, msk_{IBS}).

- **KeyGen**(*ID*): Let $sk_{IBE} \leftarrow KeyGen_{IBE}(ID)$ and $sk_{IBS} \leftarrow KeyGen_{IBS}(ID)$. The secret key corresponding to the identity *ID* will be (sk_{IBE}, sk_{IBS}).

- **Signcryption**: Given a message m, the sender with identity ID_{Sen}, runs
  - $(c,d) \leftarrow Commit(m)$
  - $e \leftarrow ENC.S_{IBE}(d, ID_{Rec}, Params_{IBE})$ and
  - $s = c||\sigma$ where $\sigma \leftarrow SIG.S_{IBS}(c, ID_{Sen}, sk_{Sen}, Params_{IBS})$;

  outputs signcryption $u = (e,s)$

- **Designcryption**: Receiver with identity ID_{Rec}, validates using $VER.S_{IBS}(c, \sigma, ID_{Sen}, Params_{IBS})$ and decrypts $d \leftarrow DEC.S_{IBE}(e, ID_{Rec}, sk_{Rec}, Params_{IBE})$ (outputting $\perp$ if either fails). The final output is $m' = Open(c,d)$. Clearly, $m' = m$ if everybody is honest.

As in [5], one can prove the following. The proofs are similar and are ommited.

Theorem 1. *If Commitment has hiding property and the Identity Based Encryption scheme is IND-ID-gCCA secure, then the Identity Based Signcryption scheme obtained by the extended An-Dodis-Rabin method is IND-IBSC-gCCA secure in the insider security model.*

Theorem 2. *If Commitment has relaxed binding property and the Identity Based Signature scheme is SUF-ID-CMA secure, then the Identity Based Signcryption scheme constructed above is also ESUF-IBSC-CMA secure in the insider security model.*

Using a particular commitment scheme and the above construction, we now obtain a generic construction of an IBSC scheme using any secure identity based encryption and signature schemes. The construction is as follows.

3.2 An Identity Based Signcryption Scheme using An-Dodis-Rabin Construction

Let m be the message whose bit size is l_1.

1. **Setup**(1^k):
   (a) Choose a Commitment Scheme, say *Commit*.
   (b) Choose an Identity Based Encryption Scheme, say S_{IBE}.
   (c) Choose an Identity Based Signature Scheme, say S_{IBS}.
   (d) Choose two secure Hash functions, $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$.

The public parameter, $Params$, is $(Params_{IBE}, Params_{IBS})$ and master secret key, msk, is (msk_{IBE}, msk_{IBS}).

2. **KeyGen**(ID): Let $sk_{IBE} \leftarrow KeyGen_{IBE}(ID)$ and $sk_{IBS} \leftarrow KeyGen_{IBS}(ID)$. The secret key corresponding to the identity ID will be (sk_{IBE}, sk_{IBS})

3. **Signcryption**($m, ID_{Rec}, ID_{Sen}, sk_{ID_{Sen}}, Params$)
   - **Commit**(m)
     (a) Choose $r \leftarrow \mathscr{R}$
     (b) Compute $h_1 = H_1(m, r)$
     (c) Compute $h_2 = H_2(r)$
     (d) Compute $c = m \oplus h_2$
     (e) Output $(h_1, c||r)$
   - **Encryption**($c||r, ID_{Rec}, Params_{IBE}$)
     (a) $c' \leftarrow ENC.S_{IBE}(c||r, ID_{Rec}, Params_{IBE})$
   - **Signature**($h_1, ID_{Sen}, sk_{ID_{Sen}}, Params_{IBS}$)
     (a) $\sigma \leftarrow SIG.S_{IBS}(h_1, ID_{Sen}, sk_{ID_{Sen}}, Params_{IBS})$

   Ciphertext $\mathscr{C} \equiv (c', h_1||\sigma)$

4. **Designcryption**($\mathscr{C}, ID_{Rec}, ID_{Sen}, sk_{ID_{Rec}}, Params$)
   - **Verify**($h_1, \sigma, ID_{Sen}, Params_{IBS}$)
     (a) $x \leftarrow VER.S_{IBS}(h_1, \sigma, ID_{Sen}, Params_{IBS})$
   - **Decryption**($c', ID_{Rec}, sk_{Rec}, Params_{IBE}$)
     (a) $c''||r' \leftarrow DEC.S_{IBE}(c', ID_{Rec}, sk_{ID_{Rec}}, Params_{IBE})$

     If either of above steps fails, then return $\perp$, else
   - **Open**($c''||r', d$)
     (a) Compute $h'_2 = H_2(r')$
     (b) Compute $m' = c'' \oplus h'_2$
     (c) Compute $h'_1 = H_1(m', r')$
     (d) Check $h_1 \overset{?}{=} H_1(m', r')$
     (e) If the above step is correctly verified, return m', else $\perp$

As defined in [13], we can split the Signcryption algorithm into two algorithms, namely $\overline{\textbf{Encrypt}}$ and $\overline{\textbf{Sign}}$, and also Designcryption algorithm into two algorithms, namely $\overline{\textbf{Decrypt}}$ and $\overline{\textbf{Verify}}$. In this case, then, in Designcryption algorithm, the output of $\overline{\textbf{Decrypt}}$ algorithm will be $(m', r', h_1, \sigma, ID_{Sen})$. $\overline{\textbf{Verify}}$ algorithm then, as input, takes $(m', r', h_1, \sigma, ID_{Sen})$ and returns $true$ if it is a valid tuple else $false$. Final output will be m' if output of $\overline{\textbf{Verify}}$ is true else final output will be $\perp$.

Remark: It can be easily shown that the commitment scheme used above has both the relaxed binding and the hiding properties in the random oracle model. Hence the Identity Based Signcryption scheme

constructed above is IND-IBSC-gCCA secure, if the Identiy Based Encryption scheme used is IND-ID-gCCA secure.
We will now show that the IBSC scheme obtained above using the An-Dodis-Rabin construction is *not* IND-IBSC-CCA secure in the insider security model.(This was also observed for the original scheme in [5])

- **Attack on Confidentiality in the Insider Security Model**
  Consider the IND-CCA game between the challenger and the adversary $\mathscr{A}$. Let $(C = (c', h_1||\sigma))$ be the challenge ciphertext obtained during the IND-IBSC-CCA game corresponding to the receiver's identity ID_{Rec} and sender's identity ID_{Sen}. In the insider security model, we may assume that the adversary $\mathscr{A}$ knows the sender's secret key $sk_{ID_{Sen}}$. Hence, since the signature algorithm is probabilistic, $\mathscr{A}$ can obtain a different signature on h_1, say σ'. Now, designcryption of $(C' = (c', h_1||\sigma'))$ corresponding to the receiver's identity ID_{Rec} and the sender's identity ID_{Sen} will yield the same message, say m that is obtained from the designcryption of $(C = (c', h_1||\sigma)$ corresponding to ID_{Rec} and ID_{Sen}. By quering the designcryption oracle, $\mathscr{A}$ easily wins the game.

4 A Modified Scheme

As observed above, the construction in section 3.2 yields only an IND-IBSC-gCCA secure scheme from an IND-ID-CCA secure IBE scheme. Thus to obtain an IND-IBSC-CCA secure IBSC scheme we need to modify the construction in section 3.2. This is done below. The modified scheme is no longer an instantiation of the extended An-Dodis-Rabin construction.

Note that the bit length of message space $\mathscr{M}$, public parameters of S_{IBE}, $Params_{IBE}$, public parameters of S_{IBS}, $Params_{IBS}$, msk_{IBE} and msk_{IBS} depend upon the security parameter. We assume that any element from $\mathscr{R}$, $\mathscr{M}$, $\mathscr{C}'$, $\mathscr{S}'$, $Params_{IBE}$, $Params_{IBS}$, msk_{IBE} and msk_{IBS} can be encoded by a string in $\{0,1\}^*$. Let l_1 be the bit-length of message m. We require that the bit-length of a random number $r \in \mathscr{R}$ should be equal to the bit-length of the message m. Let l_2 be the bit length of signatures $s \in S'$. Moreover, assume that $Params_{IBE} \cap Params_{IBS} = \phi$.

Setup(1^k)

Let $H_1 : \{0,1\}^* \to \{0,1\}^{l_1}$, $H_2 : \{0,1\}^* \to \{0,1\}^{l_1}$ be secure hash functions. The public parameter, *Params*, is $(Params_{IBE}, Params_{IBS}, H_1, H_2)$ and the master key, *msk*, is (msk_{IBE}, msk_{IBS}).

KeyGen(*ID*)

Let $sk_{IBE} \leftarrow KeyGen_{IBE}(ID)$ and $sk_{IBS} \leftarrow KeyGen_{IBS}(ID)$. The secret key corresponding to the identity ID will be (sk_{IBE}, sk_{IBS}).

Signcryption$(m, ID_{Rec}, ID_{Sen}, sk_{ID_{Sen}}, Params)$

1. Choose $r \in \mathscr{R}$
2. $h_1 = H_1(m, r, ID_{Rec}, ID_{Sen})$
3. $c = ENC.S_{IBE}(r, ID_{Rec}, Params_{IBE})$
4. $\sigma = SIG.S_{IBS}(h_1, ID_{Sen}, sk_{ID_{Sen}}, Params_{IBS})$

5. $h_2 = H_2(r, c, h_1, \sigma, ID_{Rec}, ID_{Sen})$
6. $c' = h_2 \oplus m$
7. $C \equiv (c, c', h_1, \sigma)$

Designcryption$(C, sk_{ID_{Rec}}, ID_{Rec}, ID_{Sen}, Params)$

1. $r' = DEC.S_{IBE}(c, ID_{Rec}, sk_{ID_{Rec}}, Params_{IBE})$
2. $VER.S_{IBS}(h_1, \sigma, ID_{Sen}, Params_{IBS})$
3. If the above step is correctly verified, then
4. compute $h_2' = H_2(r', c, h_1, \sigma, ID_{Rec}, ID_{Sen})$
5. compute $m' = h_2' \oplus c'$
6. check $h_1 \overset{?}{=} H_1(m', r', ID_{Rec}, ID_{Sen})$
7. If the above step is correctly verified, then return m'.

In this scheme also, as defined in [13], we can split the Signcryption algorithm into two algorithms, namely $\overline{\textbf{Encrypt}}$ and $\overline{\textbf{Sign}}$, and also Designcryption algorithm into two algorithms, namely $\overline{\textbf{Decrypt}}$ and $\overline{\textbf{Verify}}$. In this case, then, in Designcryption algorithm, the output of $\overline{\textbf{Decrypt}}$ algorithm will be $(m', r', h_1, \sigma, ID_{Sen})$. $\overline{\textbf{Verify}}$ algorithm then takes $(m', r', h_1, \sigma, ID_{Sen})$ as input and returns $true$ if it is a valid tuple else $false$. Final output will be m' if output of $\overline{\textbf{Verify}}$ is true else final output will be $\perp$.

Remark: Note that, $ENC.S_{IBE}(.)$ and $SIG.S_{IBS}(.)$ can be run in parallel in Signcryption(.) and $DEC.S_{IBE}$ and $VER.S_{IBS}$ can be run in parallel in Designcryption(.). As a result our construction is more efficient than that obtained in [11]

5 Security of the Modified Scheme

5.1 Message Confidentiality

Theorem 3. *Let $\mathscr{A}$ be a probabilistic polynomial time (PPT) adversary which can break our scheme in the IND-IBSC-CCA game with an advantage ε in the random oracle model. Then there exists a PPT adversary $\mathscr{B}$ which can break S_{IBE} (Identity Based Encryption scheme used) in the IND-ID-CCA game with an advantage at least $\frac{\varepsilon}{2}$.*

Proof. Let there be two PPT challengers $Challenger_1$ and $Challenger_2$. $Challenger_1$ first runs the setup algorithm of S_{IBE} and gives the public parameters $Params_{IBE}$ to $Challenger_2$. $Challenger_2$ here acts as a PPT adversary $\mathscr{B}$ also which will break the IND-ID-CCA security of S_{IBE}. $Challenger_2$ then runs the setup algorithm of S_{IBS} (Identity Based Signature scheme used in our scheme). Let $Params_{IBS}$ be the public parameters of S_{IBS}. $Challenger_2$ then gives the public parameters $Params \equiv (Params_{IBE}, Params_{IBS})$ to the PPT adversary $\mathscr{A}$. Without loss of generality, we assume that $Params_{IBE} \cap Params_{IBS} = \phi$. $\mathscr{B}$ maintains two lists L_1 and L_2 for queries on the hash functions H_1 and H_2. Besides these, $\mathscr{B}$ maintains two other lists $\mathscr{S}_1$ and $\mathscr{S}_2$ for queries on the secret keys of different identities corresponding to S_{IBE} and S_{IBS}.

We now explain how requests from $\mathscr{A}$ are treated by $\mathscr{B}$ who plays the role of a challenger to $\mathscr{A}$.

- H_1 queries : For inputs $m_i \in \mathscr{M}$, $r_i \in \mathscr{R}$, $ID_{Rec_i} \in \{0,1\}^*$, $ID_{Sen_i} \in \{0,1\}^*$, $\mathscr{B}$ searches the list L_1 for the tuple $(m_i, r_i, ID_{Rec_i}, ID_{Sen_i}, h_{1i})$. If such a tuple exists, $\mathscr{B}$ returns h_{1i} to $\mathscr{A}$, else $\mathscr{B}$ randomly selects h_{1i} from $\{0,1\}^{l_1}$ and adds the tuple $(m_i, r_i, ID_{Rec_i}, ID_{Sen_i}, h_{1i})$ to the list L_1 and returns h_{1i} to $\mathscr{A}$.

- H_2 queries : For inputs $r_i \in \mathscr{R}$, $c_i \in \mathscr{C}'$, $h_{1i} \in \{0,1\}^{l_1}$, $\sigma_i \in \mathscr{S}'$, $ID_{Rec_i} \in \{0,1\}^*$, $ID_{Sen_i} \in \{0,1\}^*$, $\mathscr{B}$ searches the tuple $(r_i, c_i, h_{1i}, \sigma_i, ID_{Rec_i}, ID_{Sen_i}, h_{2i})$ in the list L_2. If such a tuple exists, $\mathscr{B}$ returns h_{2i} to $\mathscr{A}$, else $\mathscr{B}$ randomly selects h_{2i} from $\{0,1\}^{l_1}$ and adds the tuple $(r_i, c_i, h_{1i}, \sigma_i, ID_{Rec_i}, ID_{Sen_i} h_{2i})$ to the list L_2. $\mathscr{B}$ then returns h_{2i} to $\mathscr{A}$.

- Secret key queries : For an input ID_i from $\mathscr{A}$, algorithm $\mathscr{B}$ responds to $\mathscr{A}$ in two steps:

  1. $\mathscr{B}$ sends ID_i to $Challenger_1$. $Challenger_1$ then run the Key Generation algorithm of S_{IBE}. Let S_{IBE} return the corresponding secret key sk_{IBE_i}. $\mathscr{B}$ then adds (ID_i, sk_{IBE_i}) into the list $\mathscr{S}_1$.
  2. As the constituent Identity Based Signature scheme, S_{IBS}, is chosen by $\mathscr{B}$, so $\mathscr{B}$ generates the secret key sk_{IBS_i} corresponding to ID_i. $\mathscr{B}$ then adds (ID_i, sk_{IBS_i}) into the list $\mathscr{S}_2$.

  $\mathscr{B}$ finally returns (sk_{IBE_i}, sk_{IBS_i}) to $\mathscr{A}$.

- Signcryption queries : The response to signcryption query for message $m_i \in \mathscr{M}$ corresponding to the receiver's identity ID_{Rec_i} and sender's identity $ID_{Sen_i} \in \{0,1\}^*$ is as follows :

  1. $\mathscr{B}$ searches the list $\mathscr{S}_2$ for the secret key corresponding to the identity ID_{Sen_i}. If it does not exist, $\mathscr{B}$ generates the secret key corresponding to ID_{Sen_i} using S_{IBS}. Let sk_{IBS_i} be the corresponding secret key.
  2. $\mathscr{B}$ then runs $Signcryption(m_i, ID_{Rec_i}, ID_{Sen_i}, sk_{IBS_i}, Params)$. Let $C_i \equiv (c_i, c'_i, h_{1i}, \sigma_i)$ be the output of the *Signcryption* algorithm. Then $\mathscr{B}$ returns C_i to $\mathscr{A}$.

- Designcryption queries : For an input $C = (c_i, c'_i, h_{1i}, \sigma_i)$, where $c_i \in \mathscr{C}'$ and $h_{1i} \in \{0,1\}^{l_1}$, σ'_i, ID_{Rec_i}, $ID_{Sen_i} \in \{0,1\}^*$ (receiver's and sender's identities are ID_{Rec_i} and ID_{Sen_i} respectively), $\mathscr{B}$ first verifies whether σ_i is a valid signature of h_{1i} or not corresponding to the identity ID_{Sen_i}. If it is so, $\mathscr{B}$ sends (c_i, ID_{Rec_i}) to $Challenger_1$. $Challenger_1$ then runs the decryption algorithm of S_{IBE} to decrypt (c_i, ID_{Rec_i}), else returns $\perp$. Let r_i be the output from the decryption algorithm of S_{IBE}. Then $\mathscr{B}$ searches the list L_2 for the tuple $(r_i, c_i, h_{1i}, \sigma_i, ID_{Rec_i}, ID_{Sen_i}, h_{2i})$. If such a tuple does not exist, $\mathscr{B}$ chooses uniformly at random a string h_{2i} from $\{0,1\}^{l_1}$ and adds $(r_i, c_i, h_{1i}, \sigma_i, ID_{Rec_i}, ID_{Sen_i}, h_{2i})$ to the list L_2. $\mathscr{B}$ then computes $m_i = h_{2i} \oplus c'$. Now, $\mathscr{B}$ searches the list L_1 for the tuple $(m_i, r_i, ID_{Rec_i}, ID_{Sen_i}, h'_{1i})$. If such a tuple exists and $h'_{1i} = h_{1i}$, then $\mathscr{B}$ returns m_i else $\perp$. If such a tuple does not exist, then $\mathscr{B}$ chooses a string h'_{1i} uniformly at random from $\{0,1\}^{l_1}$ and adds $(m_i, r_i, ID_{Rec_i}, ID_{Sen_i}, h'_{1i})$ to the list L_1. If $h'_{1i} = h_{1i}$, then $\mathscr{B}$ returns m_i, else $\perp$.

Once $\mathscr{A}$ decides to enter the challenge phase, it chooses two messages m_0, m_1 of same length and two indentities ID_R and ID_S corresponding to the receiver's and sender's identities respectively and sends them to $\mathscr{B}$. $\mathscr{B}$ then chooses two random strings $r_0, r_1 \in \{0,1\}^{l_1}$ and the receiver's identity ID_R and sends them to $Challenger_1$. $Challenger_1$ then chooses a bit, say b, uniformly at random from $\{0,1\}$ and computes the ciphertext c_b corresponding to receiver's identity by running the encryption algorithm of S_{IBE}. $Challenger_1$ then sends the ciphertext c_b to $\mathscr{B}$. $\mathscr{B}$ then searches the list L_1 for the tuple $(m_0, r_0, ID_R, ID_S, h_{1_0})$. If such a tuple doesn't exist, $\mathscr{B}$ chooses a string h_{1_0} uniformly at random from $\{0,1\}^{l_1}$ and adds the tuple $(m_0, r_0, ID_R, ID_S, h_{1_0})$. $\mathscr{B}$ then computes a signature on h_{1_0}, say σ_0 corresponding to identity ID_S. $\mathscr{B}$ then searches the tuple $(r_0, c_b, h_{1_0}, \sigma_0, ID_R, ID_S, h_{2_0})$

in the list L_2. If such a tuple doesn't exist. $\mathscr{B}$ chooses a string uniformly at random, say h_{2_0} from $\{0,1\}^{l_1}$ and adds the tuple $(r_0, c_b, h_{1_0}, \sigma_0, ID_R, ID_S, h_{2_0})$ to the list L_2. $\mathscr{B}$ then computes $c'_0 = h_{2_0} \oplus m_0$. Let $C_0 \equiv (c_b, c'_0, h_{1_0}, \sigma_0)$. $\mathscr{B}$ then searches the list L_1 for the tuple $(m_1, r_1, ID_R, ID_S, h_{1_1})$. If such a tuple doesn't exist, $\mathscr{B}$ chooses a string h_{1_1} uniformly at random from $\{0,1\}^{l_1}$ and adds the tuple $(m_1, r_1, ID_R, ID_S, h_{1_1})$. $\mathscr{B}$ then computes a signature on h_{1_1}, say σ_1 corresponding to the identity ID_S. $\mathscr{B}$ then searches the tuple $(r_1, c_b, h_{1_1}, \sigma_1, ID_R, ID_S, h_{2_1})$ in the list L_2. If such a tuple doesn't exist. $\mathscr{B}$ chooses a string uniformly at random, say h_{2_1} from $\{0,1\}^{l_1}$ and adds the tuple $(r_1, c_b, h_{1_1}, \sigma_1, ID_R, ID_S, h_{2_1})$ to the list L_2. $\mathscr{B}$ then computes $c'_1 = h_{2_1} \oplus m_1$. Let $C_1 \equiv (c_b, c'_1, h_{1_1}, \sigma_1)$. $\mathscr{B}$ then chooses a bit $v \in \{0,1\}$ uniformly at random and returns C_v to the adversary $\mathscr{A}$. For the sake of simplicity, we denote the challenged ciphertext C_v to be $C \equiv (c, c', h_1, \sigma)$.

$\mathscr{A}$ then performs a second series of queries which is treated in the same way for H_1, H_2, Secret Key (except secret key query on identity ID_R) and Signcryption queries. For Designcryption queries, given an input $C = (c_j, c'_j, h_{1j}, \sigma_j)$ where $c_j \in \mathscr{C}'$ and $h_{1j} \in \{0,1\}^{l_1}$, σ'_j, ID_{Rec_j}, $ID_{Sen_j} \in \{0,1\}^*$ (receiver's and sender's identities are ID_{Rec_j} and ID_{Sen_j} respectively), the following cases arise

1. If $(c_j, ID_{Rec_j}) \neq (c, ID_R)$, $\mathscr{B}$ responds in the same way as it does for designcryption queries in the first stage.

2. If $(c_j, h_{1j}, \sigma_j, ID_{Rec_j}, ID_{Sen_j}) = (c, h_1, \sigma, ID_R, ID_S)$, then $c'_j \neq c'$ (otherwise $(c_j, c'_j, h_{1j}, \sigma_j)$ will be same as the challenge ciphertext for the corresponding receiver's identity ID_{Rec_j} and sender's identity ID_{Sen_j}). Since $(c_j, ID_{Rec_j}) = (c, ID_R)$, decryption of c_j for identity ID_{Rec_j} will yield r_j which will be same as r_b. $\mathscr{B}$ then searches the tuple $(r_0, c_j, h_{1j}, \sigma_j, ID_{Rec_j}, ID_{Sen_j}, h_{1_0})$ in the list L_2 and computes $m_{j_0} = c'_j \oplus h_{1_0}$. Similarly, $\mathscr{B}$ searches the tuple $(r_1, c_j, h_{1j}, \sigma_j, ID_{Rec_j}, ID_{Sen_j}, h_{1_1})$ in the list L_2 and computes $m_{j_1} = c'_j \oplus h_{1_1}$. $\mathscr{B}$ then searches the tuple $(m_{j_0}, r_0, ID_{Rec_j}, ID_{Sen_j}, h_{1j})$ and $(m_{j_1}, r_1, ID_{Rec_j}, ID_{Sen_j}, h_{1j})$ in the list L_1. If either of the tuples exists, then $\mathscr{B}$ aborts the game and returns the corresponding bit as the final guess bit to the challenger. If no such tuples exist, then $\mathscr{B}$ chooses two strings uniformly at random, say h'_{1j} and $h^{\bar{\prime}}_{1j}$ from $\{0,1\}^{l_1}$ such that $h'_{1j} \neq h_{1j}$ and $h^{\bar{\prime}}_{1j} \neq h_{1j}$. $\mathscr{B}$ then adds the tuples $(m_{j_0}, r_0, ID_{Rec_j}, ID_{Sen_j}, h'_{1j})$ and $(m_{j_1}, r_1, ID_{Rec_j}, ID_{Sen_j}, h^{\bar{\prime}}_{1j})$ to the list L_1 and returns $\perp$ to the adversary $\mathscr{A}$.

3. If $(c_j, ID_{Rec_j}) = (c, ID_R)$, but $(h_{1j}, \sigma_j, ID_{Sen_j}) \neq (h_1, \sigma, ID_S)$. As $(c_j, ID_{Rec_j}) = (c, ID_R)$, decryption of c_j for identity ID_{Rec_j} will yield r_j which will be the same as r_b. $\mathscr{B}$ then searches in the list L_2 for the tuple $(r_0, c_j, h_{1j}, \sigma_j, ID_{Rec_j}, ID_{Sen_j}, h_{2j_0})$. If such a tuple doesn't exist, then $\mathscr{B}$ chooses a string uniformly at random, say $h_{2j_0} \in \{0,1\}^{l_1}$ and adds the tuple $(r_0, c_j, h_{1j}, \sigma_j, ID_{Rec_j}, ID_{Sen_j}, h_{2j_0})$ in the list L_2. $\mathscr{B}$ then computes $m_{j_0} = c'_j \oplus h_{2j_0}$. Similarly, $\mathscr{B}$ searches the tuple $(r_1, c_j, h_{1j}, \sigma_j, ID_{Rec_j}, ID_{Sen_j}, h_{2j_1})$ in the list L_2. If such a tuple doesn't exist, then $\mathscr{B}$ chooses a string uniformly at random, say $h_{2j_1} \in \{0,1\}^{l_1}$ and adds the tuple $(r_1, c_j, h_{1j}, \sigma_j, ID_{Rec_j}, ID_{Sen_j}, h_{2j_1})$ in the list L_2. $\mathscr{B}$ then computes $m_{j_1} = c'_j \oplus h_{2j_1}$. $\mathscr{B}$ then searches the tuple $(m_{j_0}, r_0, ID_{Rec_j}, ID_{Sen_j}, h_{1j})$ and $(m_{j_1}, r_1, ID_{Rec_j}, ID_{Sen_j}, h_{1j})$ in the list L_1. If either of the tuples exists, then $\mathscr{B}$ aborts the game and returns the corresponding bit as the final guess bit to the challenger. If no such tuple exists, then $\mathscr{B}$ chooses two strings uniformly at random, say h'_{1j} and $h^{\bar{\prime}}_{1j}$ from $\{0,1\}^{l_1}$ such that $h'_{1j} \neq h_{1j}$ and $h^{\bar{\prime}}_{1j} \neq h_{1j}$. $\mathscr{B}$ then adds the tuples $(m_{j_0}, r_0, ID_{Rec_j}, ID_{Sen_j}, h'_{1j})$ and $(m_{j_1}, r_1, ID_{Rec_j}, ID_{Sen_j}, h^{\bar{\prime}}_{1j})$ in the list L_1 and returns $\perp$ to the adversary $\mathscr{A}$.

In the above designcryption queries, $\mathscr{B}$ aborts the game if the following two cases arise:

1. $\mathscr{A}$ queries for $(m_{j_0}, r_0, ID_{Rec_j}, ID_{Sen_j}, h_{1j})$ or $(m_{j1}, r_1, ID_{Rec_j}, ID_{Sen_j}, h_{1j})$ randomly to H_1 oracle. Since, the choice of r_0 and r_1 by $\mathscr{B}$ is completely random from the adversary $\mathscr{A}$'s point of view, this case occurs with negligible probability.

2. $\mathscr{A}$ correctly decrypts c corresponding to the receivers's identity ID_R. Hence, in this case, the probability of winning the game by $\mathscr{B}$ will be 1.

Note that in the above game, $\mathscr{B}$ interacts with $Challenger_1$ as in the real game. Secret key query for identity ID_R has not been asked by $\mathscr{A}$ to $\mathscr{B}$, hence by $\mathscr{B}$ to $Challenger_1$. Besides it, $\mathscr{B}$ has not queried on the challenge ciphertext to $Challenger_1$.

At the end of the simulation, $\mathscr{B}$ will use the bit guessed by $\mathscr{A}$ to guess the challenge bit with S_{IBE}. If $\mathscr{A}$ guesses $w \in \{0,1\}$, $\mathscr{B}$ will also guess the same bit *viz* w. We divide the analysis of the success probability of $\mathscr{B}$ into two cases:

- $\mathscr{B}$ does not abort the game.
  1. If $b = v$, the simulation is perfect and the ciphertext, C, produced by $\mathscr{B}$ will be a valid ciphertext of the message m_v corresponding to ID_R (receiver's identity) and ID_S (sender's identity). Let
     - E_1 denote the event that $\mathscr{B}$ wins.
     - E_2 denote the event that $b = v$.
     - E_3 denote the event that $\mathscr{A}$ wins.
     - E_4 denote the event that C is a valid ciphertext.

     Then
     $$\Pr(E_1|E_2) = \Pr(E_3|E_4)$$
     WLOG, we assume
     $$\Pr(E_3|E_4) \geq \frac{1}{2}$$
  2. Suppose $b \neq v$.
     (a) Suppose $\mathscr{A}$ recognizes C as an invalid ciphertext. In that case, $\mathscr{B}$ will guess the bit $b = \bar{v}$, i.e., the complement of v. In this case, $\mathscr{B}$ will guess correctly.
     Let
     - E_5 denote the event that $b \neq v$.
     - E_6 denote the event that $\mathscr{A}$ recognizes C as an invalid ciphertext.

     Then
     $$\Pr(E_6|E_2) = 0$$
     and
     $$\Pr(E_1|E_5 \cap E_6) = 1$$
     Let
     $$\Pr(E_6|E_5) = p$$
     (b) Now assume that $\mathscr{A}$ doesn't recognize C as an invalid ciphertext. In this case, from $\mathscr{A}'s$ point of view, C will appear as a random ciphertext unless $\mathscr{A}$ queries H_2 on input (r_v, c_b). Since, r_v was chosen uniformly at random from $\{0,1\}^{l_1}$ by $\mathscr{B}$, hence the probability that $\mathscr{A}$ will query H_2 on input (r_v, c_b) is negligible. So, from $\mathscr{A}'s$ point of view, C will appear as a random ciphertext. Let,
     - E_7 denote the event that $\mathscr{A}$ recognizes C as a random ciphertext.

Then

$$\Pr(E_7|E_5) = 1 - p$$

$$\Pr(E_1|E_5 \cap E_7) = \tfrac{1}{2}$$

Note that

- $E_5 = E_2^C$
- $\Pr(E_2) = \Pr(E_5) = 1/2$
- $(E_5 \cap E_6) \cup (E_5 \cap E_7) = E_5$
- $E_2 \cap (E_5 \cap E_6) = \phi$
- $E_2 \cap (E_5 \cap E_7) = \phi$
- $(E_5 \cap E_6) \cap (E_5 \cap E_7) = \phi$
- $\Pr(E_5 \cap E_6) = \Pr(E_6|E_5)\Pr(E_5) = \frac{1}{2}p$
- $\Pr(E_5 \cap E_7) = \Pr(E_7|E_5)\Pr(E_5) = \frac{1}{2}(1-p)$

Therefore,

$$\Pr(E_1) = \Pr(E_1|E_2)\Pr(E_2) + \Pr(E_1|E_5 \cap E_6)\Pr(E_5 \cap E_6) + \Pr(E_1|E_5 \cap E_7)\Pr(E_5 \cap E_7) \tag{1}$$

$\Rightarrow \Pr(E_1) = \frac{1}{2}\Pr(E_1|E_2) + \frac{1}{2}p + \frac{1}{2}.\frac{1}{2}(1-p) \geq \frac{1}{2}\Pr(E_1|E_2) + \frac{1}{4}$
Since,

$$\Pr(E_1|E_2) = \Pr(E_3|E_4) \geq \frac{1}{2} \tag{2}$$

$\Rightarrow \Pr(E_1) \geq \frac{1}{2}\Pr(E_3|E_4) + \frac{1}{4} \geq \frac{1}{2}$

- $\mathscr{B}$ aborts the game.
  Let

  - E_8 denotes the event that $\mathscr{A}$ correctly decrypts the ciphertext c.

  Then

  $$\Pr(\mathscr{B} \text{ aborts the game}) = \Pr(E_8). \tag{3}$$

  In this case,

  $$\Pr(\mathscr{B} \text{ wins}) = 1. \tag{4}$$

Using equations (3) and (4), we get
$\Pr(\mathscr{B}$ wins$) = \Pr(\mathscr{B}$ wins $|$ $\mathscr{B}$ aborts the game$)\Pr(\mathscr{B}$ aborts the game$) + \Pr(\mathscr{B}$ wins $|$ $\mathscr{B}$ doesn't abort the game$)\Pr(\mathscr{B}$ doesn't abort the game$)$
$\Rightarrow \Pr(\mathscr{B}$ wins$) = 1.\Pr(E_8) + \Pr(E_1)(1 - \Pr(E_8))$
$\Rightarrow \Pr(\mathscr{B}$ wins$) = \Pr(E_1) + \Pr(E_8)(1 - \Pr(E_1))$
$\Rightarrow \Pr(\mathscr{B}$ wins$) \geq \Pr(E_1)$

Hence,
Advantage of $\mathscr{B} = \mathscr{A}dv(\mathscr{B}) = |\Pr(\mathscr{B} \text{ wins}) - \frac{1}{2}|$
$\geq |(\Pr(E_1) - \frac{1}{2})| \geq \frac{1}{2}(|\Pr(E_3|E_4) - \frac{1}{2}|) = \frac{1}{2}(\mathscr{A}dv(\mathscr{A})) = \frac{\varepsilon}{2}$

□

5.2 Ciphertext Unforgeability

We can similarly prove ciphertext unforgeability. We show that our scheme is ESUF-IBSC-CMA secure under the random oracle model provided the underlying IBS scheme is strongly unforgeable under adaptive chosen message attack.

Theorem 4. *Let $\mathscr{A}$ be a probabilistic polynomial time (PPT) adversary which can break our scheme in the ESUF-IBSC-CMA game with an advantage ε in the random oracle model. Then there exists a PPT adversary $\mathscr{B}$ which can break S_{IBS} (Identity Based Signature scheme used) in the SUF-ID-CMA game with an advantage ε.*

Proof. Suppose algorithm $\mathscr{B}$ receives public parameters $Params_{IBS}$ from $Challenger_1$ in the SUF-ID-CMA game. Then $\mathscr{B}$ will choose an Identity Based Encryption scheme S_{IBE} whose public parameters $Params_{IBE}$ are independently generated from the public parameters of S_{IBS}. We assume that $Params_{IBS} \cap Params_{IBE} = \phi$. $\mathscr{B}$ maintains two lists L_1 and L_2 for queries on the hash functions H_1 and H_2. Besides these, $\mathscr{B}$ maintains two other lists $\mathscr{S}_1$ and $\mathscr{S}_2$ for queries on the secret keys of different identities corresponding to S_{IBE} and S_{IBS} .

We now explain how requests from $\mathscr{A}$ are treated by $\mathscr{B}$. The response to H_1 and H_2 queries are exactly as in the proof of Theorem 4.1.

- Secret key queries : To a query for ID_i from $\mathscr{A}$, $\mathscr{B}$ responds to $\mathscr{A}$ in two steps:

  1. $\mathscr{B}$ sends ID_i to $Challenger_1$. $Challenger_1$ then runs the Key Generation algorithm of S_{IBS}. Let S_{IBS} return the corresponding secret key sk_{IBS_i}. $\mathscr{B}$ then adds (ID_i, sk_{IBS_i}) to the list $\mathscr{S}_2$ corresponding to the identity ID_i.
  2. As the constituent Identity Based Encryption scheme, S_{IBE}, is chosen by $\mathscr{B}$, so $\mathscr{B}$ generates the secret key sk_{IBE_i} corresponding to ID_i. $\mathscr{B}$ then adds (ID_i, sk_{IBE_i}) to the list $\mathscr{S}_1$ corresponding to the identity ID_i.

  $\mathscr{B}$ finally returns (sk_{IBE_i}, sk_{IBS_i}) to $\mathscr{A}$.

- Signcryption queries : For input $m_i \in \mathscr{M}$, $ID_{Rec_i}, ID_{Sen_i} \in \{0,1\}^*$ (receiver's and sender's identities are ID_{Rec_i} and ID_{Sen_i} respectively), $\mathscr{B}$ first chooses a number, say $r_i \in \mathscr{R}$. $\mathscr{B}$ then checks in the list L_1 for the tuple $(m_i, r_i, ID_{Rec_i}, ID_{Sen_i}, h_{1i})$. If such a tuple doesn't exist, then $\mathscr{B}$ chooses a string, say h_{1i},uniformly at random from $\{0,1\}^{l_1}$ and adds the tuple $(m_i, r_i, ID_{Rec_i}, ID_{Sen_i}, h_{1i})$ to the list L_1. $\mathscr{B}$ then sends (h_{1i}, ID_{Sen_i}) to S_{IBS} to get a signature for h_{1i}. Let σ_i be the output from the signature algorithm of S_{IBS}. $\mathscr{B}$ then runs the Encryption algorithm of S_{IBE} corresponding to the input (r_i, ID_{Rec_i}). Let c_i be the output of the encryption algorithm of S_{IBE}. Then $\mathscr{B}$ searches the list L_2 for a tuple $(r_i, c_i, h_{1i}, \sigma_i, h_{2i})$. If such a tuple does not exist, $\mathscr{B}$ chooses a string uniformly at random, say h_{2i}, from $\{0,1\}^{l_1}$ and adds $(r_i, c_i, h_{1i}, \sigma_i, h_{2i})$ to the list L_2. $\mathscr{B}$ then computes $c_i' = h_{2i} \oplus m_i$ and returns $C_i = (c_i, c_i', h_{1i}, \sigma_i)$ to $\mathscr{A}$.

- Designcryption queries : Designcryption query for ciphertext $C_i = (c_i, c_i', h_i, \sigma_i)$, where $c_i \in \mathscr{C}'$, $h_i \in \{0,1\}^{l_1}$ and $\sigma_i \in \{0,1\}^{l_2}$ corresponding to the receiver's identity ID_{Rec_i} and the sender's identity $ID_{Sen_i} \in \{0,1\}^*$, is executed as follows :

  1. $\mathscr{B}$ searches the list $\mathscr{S}_1$ for the secret key corresponding to the identity ID_{Rec_i}. If it does not exist, $\mathscr{B}$ generates the secret key corresponding to ID_{Rec_i} using the key generation algorithm of S_{IBE}. Let sk_{IBE_i} be the corresponding secret key.

2. $\mathscr{B}$ then runs Designcryption($C_i, ID_{Rec_i}, ID_{Sen_i}, sk_{IBE_i}, Params$). Let x be the output of the Designcryption algorithm. $\mathscr{B}$ then returns x to $\mathscr{A}$.

Note that to designcrypt the message associated to the sender's identity ID_{Sen}, only the receiver's secret key is required. One does not require the secret key of ID_{Sen}.

Once this game is over, $\mathscr{A}$ submits a ciphertext $C = (c, c', h_1, \sigma)$ corresponding to the receiver's identity ID_{Rec} and the sender's identity ID_{Sen} such that the secret key corresponding to ID_{Sen} has not been queried earlier.

Regarding (h_1, σ, ID_{Sen}), the following cases arise:

1. $\mathscr{B}$ has not queried the $Challenger_1$ on h_1 corresponding to the sender's identity ID_{Sen}. $\mathscr{B}$ then submits (h_1, σ) corresponding to the sender's identity ID_{Sen} to $Challenger_1$. In this case, $\mathscr{B}$ wins the game.

2. (h_1, σ) corresponding to the sender's identity has been obtained by $\mathscr{B}$ from the the $Challenger_1$; hence, $\mathscr{A}$ has obtained some ciphertext $C_i \equiv (c_i, c'_i, h_{1i}, \sigma_i)$ from the Signcryption oracle for some receiver's idenity ID_{Rec_i} and ID_{Sen_i}, where $(h_{1i}, ID_{Sen_i}) = (h_1, ID_{Sen})$ and $\sigma_i \neq \sigma$. $\mathscr{B}$ then submits (h_1, σ) corresponding to the sender's identity ID_{Sen} to $Challenger_1$. In this case also, $\mathscr{B}$ wins the game.

3. (h_1, σ) corresponding to the sender's identity has been obtained by $\mathscr{B}$ from the $Challenger_1$; hence, $\mathscr{A}$ has obtained some ciphertext $C_i \equiv (c_i, c'_i, h_{1i}, \sigma_i)$ from the Signcryption oracle for some receiver's idenity ID_{Rec_i} and ID_{Sen_i}, where $(h_{1i}, \sigma_i, ID_{Sen_i}) = (h_1, \sigma, ID_{Sen})$. Since $h_{1i} = h_1$, with negligible probability, two different input value on H_1 will yield the same output (assuming H_1 as a random function). Hence, designcryption of $C = (c, c', h_1, \sigma)$, say m, corresponding to the receiver's identity ID_{Rec} and the sender's identity ID_{Sen}, and the designcryption of $C_i = (c_i, c'_i, h_{1i}, \sigma_i)$, say m_i, corresponding to the receiver's identity ID_{Rec_i} and sender's identity ID_{Sen_i} will be same, i.e. $m = m_i$. Again decryption of c, say r, corresponding to the receiver's identity ID_{Rec} and decryption of c_i, say r_i, corresponding to the receiver's identity ID_{Rec_i} will also be the same, i.e. $r = r_i$. Moreover, $ID_{Rec} = ID_{Rec_i}$. Hence in this case, $(m, r, h_i, \sigma) = (m_i, r_i, h_{1i}, \sigma_i)$ corresponding to the sender's identity ID_{Sen}. So, this case will not arise as per the definition of ESUF-IBSC-CMA's security game.

Case 1 and 2 implies that whenever $C = (c, c', h_1, \sigma)$ is a valid forged ciphertext submitted by $\mathscr{A}$ defined in the security game of ESUF-IBSC-CMA, the forged message-signature pair (h_1, σ) submitted by $\mathscr{B}$ to $Challenger_1$ will be a valid according to the definition of security game of SUF-ID-CMA.
Hence, whenever $\mathscr{A}$ is able to submit a valid ciphertext to $\mathscr{B}$, $\mathscr{B}$ will also be able to submit a valid forged message-signature pair to $Challenger_1$.
So, advantage of $\mathscr{B} = \mathscr{A}dv(\mathscr{B}) = \Pr(\mathscr{B}$ wins$) = \Pr(\mathscr{A}$ wins$) = \mathscr{A}dv(\mathscr{A}) = \varepsilon$.

□

6 Efficiency

1. Time Efficiency : In our proposed scheme, encryption and signature can be done in parallel in the signcryption algorithm and decryption and verification can be done in parallel in the designcryption algorithm. Let t_E, t_S, t_D, t_V and t_H be the time taken by the encryption, sign, decryption, verification and hash algorithms respectively. Then, assuming that the signature and encryption

are computed concurrently in signcryption, the time taken by our scheme in signcryption will be $T_{SC} = \max(t_E, t_S) + 2t_H$ and assuming that decryption and verification are computed concurrently in designcryption, the time taken by our scheme in designcryption will be $T_{DSC} = \max(t_D, t_V) + 2t_H$; whereas in the Sign-then-Encrypt approach, the total time taken in signcryption will be $(t_E + t_S)$ and in designcryption will be $(t_D + t_V)$. In general, $t_H << (t_E$ or t_S or t_D or $t_V)$.

2. Space Efficiency : In many cases, in practice, the ciphertext length bears (approx.) a constant ratio with the plaintext. This is also the case with many signature schemes. Let the output length of the encryption algorithm be (at most) αl_1, where l_1 is the bit length of message m. Let the output length of the signature corresponding to a l_1 bit message be (at most) $l_2 = \beta l_1$. Hence, the total length of ciphertext will be (at most) $(\alpha + \beta + 2)l_1$. But in the Sign-then-Encrypt approach, ciphertext length will be, roughly, $\alpha(\beta + 1)l_1$. Hence, our scheme is likely to produce a shorter ciphertext length compared to the Sign-then-Encrypt approach if $\alpha \geq \frac{2}{\beta} + 1$.

7 Comparisons

Using our generic method, we composed two IBSC scheme - first one by composing Boneh-Franklin Identity Based Encryption (BF-IBE) [4] with Shamir's Identity Based Signature (SH-IBS) [1] scheme and the second one by composing Boneh-Franklin IBE [4] with Kurosawa-Heng Identity Based Signature (KH-IBS) ([7], page 113 of [2]) scheme. We compared these schemes with the Identity Based Sgncryption (IBSC) schemes proposed by Boyen [13], Chen-Malone-Lee [8] and Barreto et. al. [9]. BF-IBE + SH-IBS (Boneh-Franklin IBE and Shamir IBS) has more than double ciphertext overhead and BF-IBE + KH-IBS (Boneh-Franklin IBE and Kurosawa-Heng IBS) has almost double ciphertext overhead than IBSC schemes proposed by Boyen, Chen-Malone-Lee and Barreto. In case of time efficiency, both schemes (BF-IBE + SH-IBS and BF-IBE + KH-IBS) take less time in signcryption (note to remember that in our method, in signcryption, encryption and signature algorithm can be run in parallel) and designcryption compared to Boyen and Chen-Malone-Lee IBSC scheme. Barreto's scheme has lower cost of computation in signcryption than that of BF-IBE + SH-IBS and BF-IBE + KH-IBS but in designcryption, BF-IBE + SH-IBS has lower and BF-IBE + KH-IBS has almost equal cost of computation than that of Libert. Summary of the efficiency comparisons has been given in table 1.

References

[1] Shamir A. Identity based cryptosystems and signature schemes. In *Proc. of the 6th Annual International Cryptology Conference (CRYPTO'84), Santa Barbara, California, USA, LNCS*, volume 196, pages 47–53. Springer-Verlag, August 1984.

[2] Libert B. New Secure Applications of Bilinear Maps in Cryptography. Ph.D. Thesis, Catholic University, Louvain, January 2006.

[3] Cocks C. An Identity-based encryption based on quadratic residues. In *Proc.of the 8th IMA International Conference (Cryptography & Coding'01), Cirencester, UK, LNCS*, volume 2260, pages 360–363. Springer-Verlag, December 2001.

[4] Boneh D. and Franklin M. Identity-based encryption from the Weil pairing. In *Proc. of the 21st Annual International Cryptology Conference (CRYPTO'01), Santa Barbara, California, USA, LNCS*, volume 2218, pages 213–229. Springer-Verlag, August 2001.

[5] An J. H., Dodis Y., and Rabin T. On the security of joint signature and encryption. In *Proc. of the 21st International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02), Amsterdam, The Netherlands, LNCS*, volume 2332, pages 83–107. Springer-Verlag, April - May 2002.

Table 1: Efficiency Comparisons

Scheme	Signcryption				Designcryption			Ciphertext Length
	E	P	SM		E	P	SM	
Boyen IBSC [13]	1	1	3		0	4	2	$2\|G_1\|+\|ID\|+\|M\|$
Chen-Malone-Lee IBSC [8]	0	1	3		0	3	1	$2\|G_1\|+\|ID\|+\|M\|$
Barreto et. al. IBSC [9]	1	0	3		1	2	1	$2\|G_1\|+\|M\|$
Different IBSC constructed using [11] generic method								
BF-IBE [4] + SH-IBS [1]	1	1	1		2	1	1	$\|G_1\|+2\|\mathbb{Z}_N^*\|+3\|M\|$
BF-IBE [4] + KH-IBS ([7], [2])	1	1	2		1	2	1	$2\|G_1\|+\|\mathbb{Z}_q^*\|+3\|M\|$
Different IBSC constructed using our generic method								
BF-IBE [4] + SH-IBS [1]	1	1	1		0	1	1	$\|G_1\|+2\|\mathbb{Z}_N^*\|+4\|M\|$
BF-IBE [4] + KH-IBS ([7], [2])	1	1	2		0	1	1	$2\|G_1\|+\|\mathbb{Z}_q^*\|+4\|M\|$

- E denotes number of exponentiation.
- P denotes number of pairing. We assume $e : G_1 \times G_1 \to G_T$, if e is a symmetric bilinear map, else $e : G_1 \times G_2 \to G_T$, in case of asymmetric bilinear map.
- SM denotes number of scalar multiplication of a point on elliptic curve.
- $|G_1|$ denotes the bit length of an element in group G_1 used in pairing.
- $|G_2|$ denotes the bit length of an element in group G_2 used in pairing.
- $|G_T|$ denotes the bit length of an element in group G_T used in pairing.
- $|ID|$ denotes the bit length of an identity.
- $|M|$ denotes the message length.
- $|Z_N^*|$ denotes the length of an element in Z_N^* where $N = pq$ is the product of two prime numbers p and q.
- $|Z_q^*|$ denotes the length of an element of Z_q^*. Here q is a prime number.
- BF-IBE denotes Boneh-Franklin Identity Based Encryption scheme [4].
- SH-IBS denotes Shamir Identity Based Signature scheme [1].
- KH-IBS denotes Kurosawa-Heng Identity Based Signature scheme ([7],[2]).

[6] Baek J., Steinfeld R., and Zheng Y. Formal proofs for the security of signcryption. In *Proc. of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems(PKC'02), Paris, France, LNCS*, volume 2274, pages 80–98. Springer-Verlag, February 2002.

[7] Kurosawa K. and Heng S.H. Identity-based identification without random oracles. In *Proc. of the 5th International Conference on Computational Science and Its Applications (ICCSA'05), Singapore, LNCS*, volume 3481, pages 603–613. Springer-Verlag, May 2005.

[8] Chen L. and Malone-Lee J. Improved identity-based signcryption. In *Proc. of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Les Diablerets, Switzerland, LNCS*, volume

2437, pages 260–275. Springer-Verlag, January 2002.

[9] Barreto P.S.L.M., Libert B., McCullagh N., and Quisquater J. J. Efficient and provably-secure identity-based signatures and signcryption from Bilinear Maps. In *Proc. of the 11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'05), Chennai, India, LNCS*, volume 3788, pages 515–532. Springer-Verlag, December 2005.

[10] Sakai R., Ohgishi K., and Kasahara M. Cryptosystems Based on Pairing. In Symposium on Cryptography and Information Security-SCIS, January 2000.

[11] Pandey S.K. and Barua R. Construction of identity based signcryption schemes. In *Proc. of the 7th Web Information Systems and Applications Conference(WISA'10), Huhehot, China, LNCS*, volume 6513, pages 1–14. Springer-Verlag, August 2010.

[12] Matsuda T., Matsuura K., and Schuldt J.C.N. Efficient constructions of signcryption schemes and signcryption composability. In *Proc. of the 10th International Conference on Cryptology in India (INDOCRYPT'09), New Delhi, India, LNCS*, volume 5922, pages 321–342. Springer-Verlag, December 2009.

[13] Boyen X. Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography. In *Proc. of the 23rd Annual International Cryptology Conference (CRYPTO'03), Santa Barbara, California, USA, LNCS*, volume 2729, pages 383–399. Springer-Verlag, August 2003.

[14] Zheng Y. Digital signcryption or how to achieve Cost(Signature & Encryption) $<<$ Cost(Signature)+Cost(Encryption). In *Proc. of the 17th Annual International Cryptology Conference (CRYPTO'97), Santa Barbara, California, USA, LNCS*, volume 1294, pages 165–179. Springer-Verlag, August 1997.

Sumit Kumar Pandey did Master of Science in Mathematics from Indian Institute of Technology, Bombay, India and then did Master of Technology in Computer Science from Indian Statistical Institute, Kolkata, India. He is a senior research fellow in Applied Statistics Unit, Indian Statistical Institute, Kolkata. His interest is in Cryptography. His research interest includes designing Encryption, Signature and Signcryption schemes in traditional Public Key, Identity Based and Certificateless setting.

Rana Barua obtained his Ph. D. in Mathematics in 1987 from the Indian Statistical Institute for his work in Descriptive Set Theory. He is presently a professor of Mathematics at the Indian Statistical Institute, Kolkata. He has done work in Descriptive Set Theory, Recursion Theory, Automata Theory and his current research interests include Cryptography and Simple Voting Games.