On the Security of Identity Based Threshold Unsigncryption Schemes

S. Sharmila Deva Selvi, S. Sree Vivek, C. Pandu Rangan
TCS Lab, Department of CSE,
Indian Institute of Technology Madras (IITM)
Chennai, India
Email: {sharmila,svivek,prangan}@cse.iitm.ac.in

S. Priti
Department of CSE,
National Institute of Technology Bhopal (NITB)
Bhopal, India
Email: preetriti@gmail.com

***Abstract*—Signcryption is a cryptographic primitive that provides confidentiality and authenticity simultaneously at a cost significantly lower than that of the naive combination of encrypting and signing the message. Threshold signcryption is used when a message to be sent needs the authentication of a certain number of members in an organisation, and until and unless a given number of members (known as the threshold) join the signcyption process, a particular message cannot be signcrypted. Threshold unsigncryption is used when this constraint is applicable during the unsigncryption process. In this work, we cryptanalyze two threshold unsigncryption schemes. We show that both these schemes do not meet the stringent requirements of insider security and propose attacks on both confidentiality and unforgeability. We also propose an improved identity based threshold unsigncryption scheme and give the formal proof of security in a new stronger security model.**

***Keywords*-Identity Based Cryptography, Threshold Unsigncryption, Cryptanalysis, Random Oracle Model.**

I. Introduction

Signcryption, proposed by Zheng in 1997 [16] is a cryptographic primitive which combines the functionality of digital signature and encryption. It not only provides authenticity and confidentiality in a single step, but also gives more efficient computations than the naive *Sign-then-Encrypt* and *Sign-and-Encrypt* approaches. Followed by the first construction in [16], many new schemes and improvements have been proposed [3][11][5][7]. Threshold Signcryption is a primitive which is the integration of threshold cryptography and signcryption. Identity based cryptosystem was proposed by Shamir in [15]. It provides a more convenient alternative to conventional Public Key Infrastructure (PKI) because it solves the problem of *Key-Management* and public key *Certification*, which were considered to be tedious in PKI.

In identity based threshold unsigncryption scheme, a signcrypted message can be unsigncrypted only when more than t members out of n members in a receiver group join during the unsigncryption protocol execution. Similarly, in identity based threshold signcryption scheme atleast t members out of a total of n members in the sender group jointly signcrypt the message. In both the cases $t \leq n$. To the best of our knowledge, there were three PKI based threshold signcryption schemes in the literature [1],[6] and [17]. Almost all of them have security weaknesses and these weaknessess were reported in [13]. The weakness in identity based threshold signcryption scheme by Fagen Li et al. was shown in [14]. Similarly, two identity based threshold unsigncryption schemes appear in the literature [9] and [10]. Both of them do not meet the stringent requirements of insider security. Even the security proofs are not consistent with the security model specified and no proof for existential unforgeability were given.

Our Contribution: In this paper, we show the weaknesses in the schemes reported in [9] and [10]. Specifically, we show that, the identity based threshold unsigncryption scheme from pairings [9] by Fagen Li et al. is not CPA (Chosen Plaintext Attack) secure and is existentially forgeable. The identity based signcryption scheme with (t, n) shared unsigncryption [10] by Fagen Li et al. is not aCCA (Adaptive Chosen Ciphertext Attack) secure and is existentially forgeable. We also propose an improvement for the threshold unsigncryption scheme in [9] and formally prove the security in the random oracle model. There is no formal security model for threshold unsigncryption in the literature, we have given an appropriate security model considering a stronger attack model for the system. We have used this model to prove the security, whereas the original scheme [9] did not give a formal treatment for the security of the scheme. We have also changed some notations in the original schemes to maintain notational consistency.

II. Preliminaries

In this section, we briefly describe the basic tools used for the construction of the scheme.

A. Computational Assumptions

We use two computational hard problems *Computational Bilinear Diffie Hellman Problem* for proving the confidentiality of the system and *Compuational Diffie Hellman Problem* for proving the unforgeability of the system. The complete description of these standard and well-known problems can be found in [8].

B. General Framework of Identity Based Threshold Unsigncryption

In identity based threshold unsigncryption scheme, a trusted central authority namely the private key generator (PKG) generates the system parameters. The private key corresponding to the identites in the system are generated by the PKG. In threshold unsigncryption scheme t out of n members in a receiver group should be able to unsigncrypt the ciphertext. On behalf of every receiver group, the PKG generates a group public and group private key using the group identity. It also generates n shares of the group private key and distributes one share to each member of the receiver group. Thus, each user in the system has his own private, public key pair as well as a private key-share if he belongs to some receiver group. For unsigncrypting the ciphertext, each group member generates the unsigncryption share using the group private key share given by PKG and the ciphertext. In a (t, n) threshold unsigncryption scheme, any t unsigncryption shares may be combined by a single legitimate user called $clerk$ to unsigncrypt the ciphertext. A typical identity based threshold unsigncryption scheme consists of the following eight algorithms:

Setup(κ***)***: Given a security parameter κ, the PKG generates the public parameters $params$, and a corresponding master secret key s.

Extract(ID_A***)***: Given the identity ID_A of the user A, the PKG computes the public key Q_A and private key D_A, and sends it to the user through a secure channel. Here we denote the groups with the subscript notation $\mathbb{A}$ or $\mathbb{B}$, depending whether it is a sender or receiver. The public key of the group $G_{\mathbb{A}}$ is $Q_{\mathbb{A}}$ and private key is $D_{\mathbb{A}}$.

Key-Share Distribution($D_{\mathbb{B}}, n, t$***)***: Given the private key $D_{\mathbb{B}}$ of user group $G_{\mathbb{B}}$, the number of members n in the unsigncryption group and t and the number of threshold members the PKG runs this algorithm to compute the private key shares Δ_i and the corresponding verification keys τ_i of these n members, by using $Shamir's$ (t, n) threshold scheme. Then each pair of private/verification key share (Δ_i, τ_i) is sent to the appropriate receiver group member. Each member of the group can have independent private and public keys.

Signcryption($m, D_A, Q_{\mathbb{B}}$***)***: To signcrypt a message m to the receiver group $G_{\mathbb{B}}$, the sender A runs this algorithm and obtains the signcryption σ.

SignVer(σ, Q_A***)***: This signature verification algorithm can be run by anyone to check the validity of the signature of the sender on σ.

Unsigncryption Share-Generation($\sigma, G_{\mathbb{B}}$***)***: This algorithm is run by each member in the receiver group $G_{\mathbb{B}}$ (let $\{B_1, B_2, ..., B_n\}$ be the set of all members in the group), to generate the unsigncryption shares after the signature is verified.

ShareVer($\sigma_1, \sigma_2, ..., \sigma_t$***)***: Given the unsigncryption share σ_i for $i = 1$ to t, the clerk runs this algorithm to verify the validity of the shares obtained from the t members of the group $G_{\mathbb{B}}$. Without loss of generality and to reduce the messy notations we assume that the first t members of the group $G_{\mathbb{B}}$ form the group $T_{\mathbb{B}}$ (let $\{B_1, B_2, ..., B_t\}$ be the set of all members in this group), who contribute their unsigncryption shares.

ShareCombine($\sigma_1, \sigma_2, ..., \sigma_t$***)***: The clerk runs this algorithm after verifying the validity of the shares from all the members of the group $T_{\mathbb{B}}$, to obtain the unsigncryption of the signcryption σ i.e. the plaintext m.

C. Security Model for ID- Based Threshold Unsigncryption

The formal security of signcryption scheme was first proposed by $Baek$ and $Zheng$ in [3]. The semantic security of identity based signcryption was first proposed by Malone-Lee in [11], this was later improved by $Boyen$ et al. in [5] providing notions for insider security and this was further modified by $Sherman$ et. al in [7] which incorporates security against *adaptive chosen ciphertext attack*, *identity attacks* and *existential unforgeability against chosen message attack*. The schemes LGH-IDBTUSC [9] and LXH-IDBSSSU [10] extended these notions and claim their schemes to be secure, but they fail to capture the security model proposed. In [2], the authors have proposed a gCCA2 secure model, but have restricted the adversary from querying a class of ciphertexts that are used to mount the CCA attack. We have described a security model by adding the insider security notion for both confidentiality and unforgeability, granting the adversary the freedom of querying any ciphertext of his choice, except the challenge ciphertext. Our model for confidentiality is similar to the model by Baek and Zheng in [4]. Here, in the confidentiality game we provide the adversary with the $t - 1$ private key shares of the target recipient group, and an unsigncryption share oracle for querying the unsigncryption shares of the uncorrupted members. Thus we give maximum advantage to the adversary and also capture the exact threshold concept in real scenario.

1) ***CONFIDENTIALITY:****: An identity based threshold unsigncryption scheme (ID-TUSC) is said to be indistinguishable against adaptive chosen ciphertext attacks (IND-ID-TUSC-aCCA2) if no polynomially bounded adversary has a non-negligible advantage in the following game between the Challenger $\mathcal{C}$ and the Adversary $\mathcal{A}$:*

Initial: The challenger $\mathcal{C}$ runs the **Setup** algorithm to generate the public parameters $params$ and master private key s. $\mathcal{C}$ gives $params$ to $\mathcal{A}$ and keeps the master private key s secret from $\mathcal{A}$.

Phase 1: In this phase, $\mathcal{A}$ performs a series of queries in an adaptive fashion, i.e. each query may depend on the responses to the previous queries. The following queries are allowed:

Key Extraction queries: $\mathcal{A}$ chooses an identity ID_i and gives it to $\mathcal{C}$ which computes the corresponding private key D_i and sends it to $\mathcal{A}$.

Signcryption queries: $\mathcal{A}$ produces the sender identity ID_i, receiver group $G_{\mathbb{J}}$'s identity $ID_{\mathbb{J}}$ and a plaintext message m to $\mathcal{C}$. $\mathcal{C}$ computes the signcryption σ of the message m, and sends σ to $\mathcal{A}$.

Unsigncryption queries: $\mathcal{A}$ produces the sender identity ID_i, receiver group $G_{\mathbb{J}}$'s identity $ID_{\mathbb{J}}$ and a signcryption σ. $\mathcal{C}$ obtains the message m from σ and returns it to $\mathcal{A}$.

Challenge: At the end of *Phase 1*, $\mathcal{A}$ sends to $\mathcal{C}$ two plaintexts m_0 and m_1 of equal length, and two identities ID_A and $ID_{\mathbb{B}}$, on which he wishes to be challenged. The adversary shouldn't have asked for the Key-Extraction query on $ID_{\mathbb{B}}$. The challenger $\mathcal{C}$ flips a fair coin $b \in_R \{0,1\}$ and computes the challenge signcryption σ^* on the message m_b and returns σ^* to $\mathcal{A}$ along with the $t-1$ private key shares of the members in the target recipient group $G_{\mathbb{B}}$.

Phase 2: In this phase $\mathcal{A}$ can adaptively perform polynomially bounded number of queries again as in *Phase 1* with the restriction that $\mathcal{A}$ cannot make a key extraction query on $ID_{\mathbb{B}}$ and cannot query the unsigncryption oracle on σ^* from sender A to the receiver group $G_{\mathbb{B}}$. In this phase, $\mathcal{A}$ is allowed one more query stared below:

Unsigncryption share queries: $\mathcal{A}$ produces the sender identity ID_i and the receiver group's identity $ID_{\mathbb{J}}$, the t^{th} member of the receiver group and a signcryption σ. $\mathcal{C}$ obtains the unsigncryption share of the t^{th} member by first retrieves the $t-1$ private key shares Δ_i, for $i=1$ to $t-1$ given to the adversary and then generates the share by running the **Unsigncryption Share-Generation**(σ, G_j) algorithm. $\mathcal{C}$ then returns σ_t to $\mathcal{A}$, iff σ is a valid signcryption from ID_i to $ID_{\mathbb{J}}$ by running the $SignVer$ algorithm, otherwise returns $\perp$.

The advantage of $\mathcal{A}$ is defined as $Adv(\mathcal{A}) = \mid 2P[b' = b] - 1 \mid$ where $P[b' = b]$ denotes the probability that $b' = b$. The adversary is allowed to make key extraction query on the signcrypting identity ID_A. This is to meet the stringent requirements of insider-security. It also ensures the forward security of the scheme, i.e. confidentiality is preserved even if the sender's private key is compromised.

*2) **EXISTENTIAL UNFORGEABILITY**: An identity based threshold unsigncryption scheme (IDTUSC) is said to be secure against an existential forgery for adaptive chosen messages attacks (EF-IDTUSC-aCMA) if no polynomially bounded adversary has a non-negligible advantage in the following game*

Initial: The challenger $\mathcal{C}$ runs the $Setup$ algorithm to generate the master public key $params$ and master private key s. $\mathcal{C}$ gives $params$ to $\mathcal{A}$ and keeps the master private key s secret from $\mathcal{A}$.

Training Phase: $\mathcal{A}$ makes polynomially bounded number of queries adaptively to the various oracles provided by $\mathcal{C}$, as described in *Phase 1* of the confidentiality game.

Forgery: At the end of the *Training Phase*, $\mathcal{A}$ chooses a message m and produces a signcryption σ^* on m with the sender and receiver identities ID_A and $ID_{\mathbb{B}}$ respectively, such that the triplet $(\sigma^*, ID_A, ID_{\mathbb{B}})$ was not the output of any previous queries to the *Signcryption Oracle* with m as the message and the private key of ID_A was not queried during the *Training Phase*. $\mathcal{A}$ wins the game if the result of **SignVer** is not $\perp$ symbol.

The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins. $\mathcal{A}$ is allowed to make a key extraction query on the forged ciphertext's receiver $ID_{\mathbb{B}}$. Again, this is to capture the notion of insider security.

Notations: From now on we represent Fagen Li et al.'s identity based threshold unsigncryption scheme from pairing [9] as LGH-IDBTUSC and Fagen Li et al.'s identity based signcryption scheme with (t,n) shared unsigncryption [10] as LXH-IDBSSSU.

III. Review and Attack of LGH-IDBTUSC

In this section, we review the identity based threshold unsigncryption scheme by Fagen Li et al.'s (LGH-IDBTUSC) proposed in [9]. We also show that it is not insider secure from CPA atack against confidentiality and is existentially forgeable.

A. Review of LGH-IDBTUSC

The LGH-IDBTUSC scheme involves four roles: The PKG, the sender A, the receiver group $G_{\mathbb{B}} = \{B_1, B_2, ..., B_n\}$ and the clerk - a member of the group who combines the unsigncryption shares from the other members, to unsigncrypt the ciphertext.

Setup: Given κ as input, the PKG does the following:

- Chooses $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and a generator P of $\mathbb{G}_1$,
- Chooses a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^{\delta}$, $H_3 : \{0,1\}^* \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q^*$ and $H_4 : \mathbb{G}_2 \times \mathbb{G}_2 \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q^*$.
- Chooses $s \in \mathbb{Z}_q^*$ and computes $P_{pub} = sP$.
- Chooses a secure symmetric cipher $(\mathcal{E}, \mathcal{D})$.

$params = \langle \mathbb{G}_1, \mathbb{G}_2, \delta, \hat{e}, P, P_{pub}, H_1, H_2, H_3, H_4, \mathcal{E}, \mathcal{D} \rangle$ and s is the master secret key.

Extract: The input to this algorithm may be the identity of an individual user or a group. The PKG computes the $Q_A = H_1(ID_A)$ and the private key $D_A = sQ_A$. The extract procedure is same for both user and the group.

Key-Share Distribution: Let t (a threshold) and n satisfies the condition $1 \leq t \leq n < q$. The PKG performs the following

- Chooses $R_j \in_R \mathbb{G}_1^*$, for $1 \leq j \leq t-1$ and constructs a function $F(u) = D_{\mathbb{B}} + \sum_{j=1}^{t-1} u^j R_j$.

- Computes the private key share of each $B_i \in G_{\mathbb{B}}$ as $\Delta_i = F(i)$ and the verification key $\tau_i = \hat{e}(\Delta_i, P)$.

- Sends the private key share Δ_i and the verification key τ_i to B_i. B_i then keeps Δ_i as secret while making τ_i public.

Signcryption: To signcrypt a message m to the recipient group $G_{\mathbb{B}}$, the sender A chooses $x \in_R \mathbb{Z}_q^*$ and computes the signcryption $\sigma = (c, r, V)$ as follows:

1) k_1 = $\hat{e}(P, P_{pub})^x$ and k_2 = $H_2(\hat{e}(P_{pub}, Q_{\mathbb{B}})^x)$.
2) c = $\mathcal{E}_{k_2}(m)$, r = $H_3(c, k_1)$ and V = xP_{pub} - rD_A.

SignVer: This algorithm can be run by anyone who wants to verify the signature on the signcryption σ. Compute k_1' = $\hat{e}(P, V)\hat{e}(P_{pub}, Q_A)^r$ and accept iff $r \stackrel{?}{=} H_3(c, k_1')$. Otherwise, return *Invalid Signcryption*.

Unsigncryption Share-Generation: Each $B_i (1 \leq i \leq t)$ checks the validity of σ by running **SignVer**. If σ is valid, each $B_i (1 \leq i \leq t)$ computes $\eta_i = \hat{e}(\Delta_i, Q_A)$; $\mu_i = \hat{e}(T_i, Q_A)$; $\omega_i = \hat{e}(T_i, P)$; $v_i = H_4(\eta_i, \mu_i, \omega_i)$ and $W_i = T_i + v_i\Delta_i$, for $T_i \in_R \mathbb{G}_1$ and sends $\sigma_i = (i, \eta_i, \mu_i, \omega_i, v_i, W_i)$ to the clerk. Otherwise, B_i returns *Invalid Signcryption*.

ShareVer: The clerk computes v_i' = $H_4(\eta_i, \mu_i, \omega_i)$ and then checks if $v_i' \stackrel{?}{=} v_i, \hat{e}(W_i, Q_A)\eta_i^{v_i'} \stackrel{?}{=} \mu_i$, and $\hat{e}(W_i, P)\tau_i^{v_i'} \stackrel{?}{=} \omega_i$. If these tests hold, then σ_i from B_i is a valid unsigncryption share. Otherwise, the clerk returns *Invalid Share*.

ShareCombine: When the clerk collects valid unsigncryption shares from all the t members in group $T_{\mathbb{B}}$=$\{B_1, B_2, ..., B_t\}$, he computes $k_2' = H_2(\hat{e}(V, Q_{\mathbb{B}})(\prod_{j=1}^{t} \eta_j^{N_{0,j}})^r)$, where $N_{0,j} = \prod_{i=1, i\neq j}^{t} \frac{0-i}{j-i}$ $mod\ q$, i and j represent the i^{th} and j^{th} users in $T_{\mathbb{B}}$ and recovers $m = \mathcal{D}_{k_2'}(c)$.

B. Attacks on LGH-IDBTUSC

Fagen Li et al. in [9] claimed that their scheme is semantically secure i.e. indistinguishable against adaptive chosen ciphertext attack, but it is not. In the following section, we show the weakness in confidentiality and unforgeability of the scheme.

1) Attack on Confidentiality by the clerk:: The proposed scheme is insecure from the point of view of attack by the clerk, who gets the shares from the members in the group to unsigncrypt the signcrypted message. The attack follows:

The members in the group give their unsigncryption shares of the signcryption σ to the clerk and the share used for obtaining the unsigncryption key k_2 is $\eta_i = \hat{e}(\Delta_i, Q_A)$, which do not include any part of the current signcryption σ. So, if the clerk is malicious, once he gets these shares from the threshold members in $G_{\mathbb{B}}$, he can use the same shares to unsigncrypt the ciphertexts being sent to $G_{\mathbb{B}}$ from A, without requesting the unsigncryption shares from the group members and calculate the unsigncryption key k_2 as $k_2 = H_2(\hat{e}(V, Q_{\mathbb{B}})(\prod_{j=1}^{t} \eta_j^{N_{0,j}})^r)$, where $N_{0,j} = \prod_{i=1, i\neq j}^{t} \frac{0-i}{j-i}$ $mod\ q$, i and j represent the i^{th} and j^{th} users in $T_{\mathbb{B}}$ and η_j is from the previous unsigncryption.

Remark: This attack is possible because the components of a specific signcryption is not bound to the computations done during unsigncryption share generation, so it will not be specific for a particular signcryption.

2) Insider attack on Confidentiality by an adversary : The scheme proposed above lacks the notion of insider security. As in the above scheme once the adversary knows the private key of the sender he can calculate the encryption key k_2 and then unsigncrypt the signcryption. The attack follows:

The adversary $\mathcal{A}$ knows the private key D_A of the sender, so to unsigncrypt $\sigma = (c, r, V)$, where $V = xP_{pub} - rD_A$, he does the following:

Computes $V - rD_A$ = xP_{pub} and $k_2 = H_2(\hat{e}(xP_{pub}, Q_{\mathbb{B}}))$. Obtains $m = \mathcal{D}_{k_2}(c)$

Thus, it is not CPA secure. In the challenge phase of the confidentiality game, when the adversary $\mathcal{A}$ gets the ciphertext σ^*, he can unsigncrypt σ^* to find out whether it is the signcryption of the message m_0 or m_1, because here the adversary knows the private key of the sender.

Remark: This attack is possible because the designers have not strictly followed the notion of insider security, so the adversary can easily retrieve the component used in the ephimeral key generation, and can thus recover the message without the need of the secret key of the receivers.

3) Attack on Unforgeability by a forger : Any can forge a signcryption from a user A to receiver group $G_{\mathbb{B}'}$, if he has a valid signcryption from A to any receiver group $G_{\mathbb{B}}$. The attack follows:

The forger $\mathcal{F}$, has a valid signcryption $\sigma = (c, r, V)$ from user A to the receiver group $G_{\mathbb{B}}$. He claims $\sigma' = \sigma$ to be a valid signcryption from the same sender A to a different receiver group $G_{\mathbb{B}'}$. The signature is accepted if and only if it passes the verification in the **SignVer** algorithm. In this case, it passes the verification, as the equality $r \stackrel{?}{=} H_3(c, k_1')$ holds, where k_1' = $\hat{e}(P, V)\hat{e}(P_{pub}, Q_A)^r$. Thus σ' is accepted by $G_{\mathbb{B}'}$ as valid, but it will be unsigncrypted to some arbitrary message, thus making it existentially forgeable.

Remark: This attack is possible because the receiver's identity is not included in the sign verification procedure. Signcryption must include both the identities of the sender and the receiver and therefore omitting any of them in

the encryption or the signature will lead to attacks in confidentiality and unforgeability of the system.

IV. Review and Attack on LXH-IDBSSSU

In this section, we review the identity based signcryption scheme with (t,n) shared unsigncryption by Fagen Li et al.'s (LXH-IDBSSSU) presented in [10]. We also show that it is not insider secure against adaptive CCA atack on confidentiality and is existentially forgeable.

A. Review of LXH-IDBSSSU

The LXH-IDBSSSU scheme involves three roles: The PKG, the sender A and the receiver group $G_{\mathbb{B}}$=$\{B_1, B_2, ..., B_n\}$, who co-operatively participate in the unsigncryption process.

Setup: This is similar to the LGH-IDBTUSC scheme in section III-A.

Extract: Given an identity ID_A, the PKG computes the user public key $Q_A = H_1(ID_A)$, computes the user's private signcryption key $S_A = s^{-1}Q_A$ and private unsigncryption key $D_A = sQ_A$. The message recipient group $G_{\mathbb{B}}$ has a public key $Q_{\mathbb{B}}$ and a corresponding private unsigncryption key $D_{\mathbb{B}} = sQ_{\mathbb{B}}$.

Key-Share Distribution: This is also similar to the LGH-IDBTUSC scheme in section III-A.

Signcryption: This algorithm is run by the sender. To send a message m to the recipient group $G_{\mathbb{B}}$, sender A chooses $x \in_R \mathbb{Z}_q^*$ and computes the signcryption $\sigma = (c, r, V)$ as follows:

- Computes $k_1 = \hat{e}(P, Q_A)^x$ and $k_2 = H_2(\hat{e}(Q_A, Q_{\mathbb{B}})^x)$.
- Computes $c = \mathcal{E}_{k_2}(m)$, $r = H_3(c, k_1)$ and $V = (x - r)S_A$.

The signcryption $\sigma = (c, r, V)$ is then sent to the receiver group $G_{\mathbb{B}}$.

Unsigncryption: Let $T_{\mathbb{B}} = \{B_1, B_2, ..., B_t\}$, be the group of t members who want to cooperatively unsigncrypt the signcryption $\sigma = (c, r, V)$. Each user B_i does the following:

- Computes $k_1' = \hat{e}(V, P_{pub})\hat{e}(Q_A, P)^r$ and accepts σ iff $r \stackrel{?}{=} H_3(c, k_1')$, otherwise return $''Invalid''$.

- Computes $\eta_i = \hat{e}(\Delta_i, V)$; $\mu_i = \hat{e}(T_i, V)$; $\omega_i = \hat{e}(T_i, P)$; $v_i = H_4(\eta_i, \mu_i, \omega_i)$ and $W_i = T_i + v_i\Delta_i$ for $T_i \in_R \mathbb{G}_1$ and sends $\sigma_i = (i, \eta_i, \mu_i, \omega_i, v_i, W_i)$ to the other $t-1$ members in $T_{\mathbb{B}}$.

- Each $\sigma_j = (j, \eta_j, \mu_j, \omega_j, v_j, W_j)$ from $B_j (j \neq i)$ is verified as follows:

B_i first computes $v_j' = H_4(\eta_j, \mu_j, \omega_j)$ and then checks if $v_j' \stackrel{?}{=} v_j$, $\hat{e}(W_j, V)/\eta_j^{v_j'} \stackrel{?}{=} \mu_j$ and, $\hat{e}(W_j, P)/\tau_j^{v_j'} \stackrel{?}{=} \omega_j$.

If all the above tests hold, then σ_j from $B_j (j \neq i)$ is a valid unsigncryption share.

- Computes $k_2' = H_2(\prod_{j=1}^{t} \eta_j^{N_{0,j}} \hat{e}(Q_A, Q_{\mathbb{B}})^r)$, where $N_{0,j} = \prod_{i=1, i\neq j}^{t} \frac{0-i}{j-i} \ mod \ q$.

- Recovers $m = \mathcal{D}_{k_2'}(c)$.

B. Attack on LXH-IDBSSSU

Fagen Li et al. in [10] claimed that their scheme is semantically secure against adaptive chosen ciphertext attacks with insider security, but we show attacks confidentiality and unforgeability of the scheme.

1) Attack on Confidentiality by an adversary:: The scheme is not insider security, that is if the private key of the sender is compromised during the adaptive CCA attack, the adversary will be able to distinguish between the messages m_0 and m_1 of the challenge signcryption. The attack follows:

During the confidentiality game, in the challenge phase the adversary $\mathcal{A}$ gives two messages m_0 and m_1, the sender identity ID_A and the receiver group $G_{\mathbb{B}}$ to the challenger $\mathcal{C}$ and obtains the signcryption $\sigma^* = (c, r, V)$, where $V = (x - r)S_A$. $\mathcal{A}$ computes $V' = V + rS_A - rS_{A'}$ and queries to the unsigncryption oracle for the unsigncryption of $\sigma' = (c, r, V')$ from the sender $ID_{A'}$ to the receiver group $G_{\mathbb{B}}$. Now since this is different from the challenge signcryption σ^*, the oracle will unsigncrypt as follows:

- Verify σ' by calculating $k_1' = \hat{e}(V', P_{pub})\hat{e}(Q_{A'}, P)^r$, and this will be equal to $\hat{e}(P, Q_A)^x$ which is the same one used for σ^*, so it will pass the verification procedure.

- Now $\mathcal{C}$ will calculate $k_2 = H_2(\prod_{j=1}^{t} \eta_j^{N_{0,j}} \hat{e}(Q_{A'}, Q_{\mathbb{B}})^r)$, where $N_{0,j} = \prod_{i=1, i\neq j}^{t} \frac{0-i}{j-i} \ mod \ q$, and this will be equal to $H_2(\hat{e}(Q_A, Q_{\mathbb{B}}))$. Thus sends $m = \mathcal{D}_{k_2}(c)$ to the adversary. This m, obtained is the same m_b used in σ^*. Thus we have proved that the given scheme is not CCA secure.

Remark: This attack is possible because there is no binding of the sender's identity in the key generation part, so by just changing one component in the signature, the adversary can recover the plaintext used in the challenge phase by querying the unsigncryption oracle in the confidentiality game.

2) Attack on Unforgeability by a forger:: If the forger $\mathcal{F}$, has a valid signcryption $\sigma = (c, r, V)$ from user ID_A to the receiver group $G_{\mathbb{B}}$, $\mathcal{A}$ can produce a forgery $\sigma' = \sigma$ from the same sender ID_A to a different receiver group $G_{\mathbb{B}'}$. This is because it passes the sign verification procedure as the equality $r \stackrel{?}{=} H_3(c, k_1')$ holds, where k_1' = $\hat{e}(P, V)\hat{e}(P_{pub}, Q_A)^r$. Thus the signcryption is valid from ID_A to $G_{\mathbb{B}'}$ but it will be unsigncrypted to some arbitrary message, making it existentially forgeable.

Remark: This attack is possible because the receiver's identity is not included in the sign verification procedure.

V. Improved ID-Based Threshold Unsigncryption Scheme (I-IDBTUSC)

In this section, we have provided a fix for the LGH-IDBTUSC scheme, to make the scheme semantically secure and secure against malicious clerks.

The improved scheme involves four roles: The PKG, the sender A, the receiver group $G_{\mathbb{B}} = \{B_1, B_2, ..., B_n\}$ and the clerk - a member of the group who combines the unsigncryption shares from the other members, to unsigncrypt the signcryption.

Setup: This algorithm run by the PKG is similar to the original scheme in [9], only modifications made are in the definitions of the hash functions and a new hash function H_5 is defined. These are: H_2 $\mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \{0,1\}^{\delta}$, H_3 $\{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$ and $H_5 : \{0,1\}^* \times \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$. Thus, the system's public parameters published by the PKG are $(\mathbb{G}_1, \mathbb{G}_2, \delta, \hat{e}, P, P_{pub}, H_1, H_2, H_3, H_4, H_5, \mathcal{E}, \mathcal{D})$ and master secret key s is kept secret.

Extract: This algorithm is run by the PKG and is similar to the original scheme.

Key-Share Distribution: This algorithm run by the PKG is also similar to the previous scheme. The PKG computes the private key share for the i^{th} member as $\Delta_i = F(i)$, where $F(.)$ is the Lagrange polynomial as mentioned in the previous scheme and the verification key $\tau_i = \hat{e}(\Delta_i, P)$, and sends Δ_i and τ_i to $B_i (1 \leq i \leq n)$. Each user can then perform a consistency check to check the validity of their secret shares by performing the following check $\hat{e}(Q_{\mathbb{B}}, P_{pub}) \stackrel{?}{=} \hat{e}(\Delta_i, P)^{N_{0,i}} \prod_{k=1, k\neq i}^{t} \tau_k^{N_{0,k}}$, where $N_{0,k} = \prod_{i=1, i\neq k}^{t} \frac{0-i}{k-i}$, i and k represent the i^{th} and k^{th} users in $G_{\mathbb{B}}$

Signcryption: This algorithm is run by the sender. To send a message m to the recipient group $G_{\mathbb{B}}$, the sender A chooses x and ρ randomly from $\mathbb{Z}_q^*$ and computes the signcryption $\sigma = (c, k_1, V, Y)$ as follows:

1) $h = H_5(m, \rho)$
2) $k_1 = xP$ and $k_2 = H_2(\hat{e}(P_{pub}, Q_{\mathbb{B}})^h, k_1, Q_A, Q_{\mathbb{B}})$.
3) $Y = hP$.
4) $c = \mathcal{E}_{k_2}(m \parallel \rho)$.
5) $r = H_3(c, k_1, Y, Q_A, Q_{\mathbb{B}})$.
6) $V = xP_{pub} - rD_A$.

The signcryption $\sigma = (c, k_1, V, Y)$ is then sent to the receiver group $G_{\mathbb{B}}$.

SignVer: This algorithm can be run by anyone who wants to verify the signature on the signcryption σ. The clerk who wants to unsigncrypt σ, computes $r' = H_3(c, k_1, Y, Q_A, Q_{\mathbb{B}})$ and accepts the signature iff $\hat{e}(V, P) \stackrel{?}{=} \hat{e}(k_1 - r'Q_A, P_{pub})$. Otherwise, he returns *Invalid Signcryption*.

Unsigncryption Share-Generation: This algorithm is run by the receiver group. The clerk requests unsigncryption shares from each member in group $G_{\mathbb{B}}$. Each $B_i (1 \leq i \leq n)$ verifies the signature of σ by running **SignVer**. If it is valid, Each $B_i (1 \leq i \leq n)$ computes $\eta_i = \hat{e}(\Delta_i, Y)$; $\mu_i = \hat{e}(T_i, Y)$; $\omega_i = \hat{e}(T_i, P)$; $v_i = H_4(\eta_i, \mu_i, \omega_i)$ and $W_i = T_i + v_i \Delta_i$ for $T_i \in_R \mathbb{G}_1$ and sends $\sigma_i = (i, \eta_i, \mu_i, \omega_i, v_i, W_i)$ to the clerk. Otherwise, B_i returns *Invalid Signcryption*.

Sharever: The clerk firstly computes $v_i' = H_4(\eta_i, \mu_i, \omega_i)$ and then checks if $v_i' \stackrel{?}{=} v_i$, $\hat{e}(W_i, Y)/\eta_i^{v_i'} \stackrel{?}{=} \mu_i$, and $\hat{e}(W_i, P)/\tau_i^{v_i'} \stackrel{?}{=} \omega_i$. If these tests hold, then σ_i from B_i is a valid unsigncryption share. Otherwise, the clerk returns *Invalid Share*.

ShareCombine: When the clerk collects valid unsigncryption shares from the t members in group $T_{\mathbb{B}}$, he computes $k_2' = H_2((\prod_{j=1}^{t} \eta_j^{N_{0,j}}), k_1, Q_A, Q_{\mathbb{B}})$, where $N_{0,j} = \prod_{i=1, i\neq j}^{t} \frac{0-i}{j-i} \; mod \; q$, i and j represent the i^{th} and j^{th} users in $T_{\mathbb{B}}$ and recovers $m \parallel \rho = \mathcal{D}_{k_2'}(c)$. He then calculates $h = H_5(m, \rho)$ and accepts the message authenticity iff $Y \stackrel{?}{=} hP$, otherwise return *Invalid Signcryption*.

Correctness: To prove the correctness of I-IDBTUSC scheme, we show how the sign verification is done in *SignVer* algorithm:

$$\hat{e}(P, V) = \hat{e}(P_{pub}, k_1 - r'Q_A) = \hat{e}(sP, xP - r'Q_A)$$
$$= \hat{e}(P, s(xP - r'Q_A)) = \hat{e}(P, xP_{pub} - r'D_A)$$
$$= \hat{e}(P, V) \text{ [Iff } r = r']$$

We also show how k_2' calculated in *ShareCombine* is the same k_2 used in *Signcryption*:

$$k_2' = H_2((\prod_{j=1}^{t} \eta_j^{N_{0,j}}), k_1, Q_A, Q_{\mathbb{B}}).$$
$$= H_2((\prod_{j=1}^{t} \hat{e}(\Delta_j, Y)^{N_{0,j}}), k_1, Q_A, Q_{\mathbb{B}}).$$
$$= H_2((\prod_{j=1}^{t} \hat{e}(N_{0,j}\Delta_j, Y)), k_1, Q_A, Q_{\mathbb{B}}).$$
$$= H_2(\hat{e}(\sum_{j=1}^{t} N_{0,j}\Delta_j, Y), k_1, Q_A, Q_{\mathbb{B}}).$$
$$= H_2(\hat{e}(D_{\mathbb{B}}, Y), k_1, Q_A, Q_{\mathbb{B}}).$$
$$= H_2(\hat{e}(D_{\mathbb{B}}, P)^h, k_1, Q_A, Q_{\mathbb{B}}).$$

Remark: The last check during unsigncryption ensures that σ is a valid signcryption from ID_A to $ID_{\mathbb{B}}$. This check is needed to ensure the consistency of key generation i.e. to confirm that the message retrieved is encrypted using the same key, otherwise it will help the adversary to play with the challenge signcryption in the confidentiality game. In many identity based signcryption schemes this check is eliminated to introduce public verifiability, but without this check we will not be able to verify the signcryption validity.

A. Security of the Scheme (I-IDBTUSC)

In Fagen Li et al.'s LGH-IDBTUSC [9], the security proof given for confidentiality is not strictly according to the security model proposed, and they have also not given any proof for unforgeability. It has weakly implemented the notion of insider security. In this section we formally prove the security of I-IDBTUSC indistinguishable against adaptive chosen ciphertext attacks *IND-I-IDBTUSC-aCCA2* and existentially unforgeable against adaptive chosen message attack and identity attack *(EUF-I-IDBTUSC-aCMA)* in random oracle model, assuming that the adversary or the forger has the access to all the private keys except the private key of the target identity for the notion of insider security. We consider the security model given in section 2.5 to prove the security of the improved scheme I-IDBTUSC

1) Existentially Unforgeablity Proof of I-IDBTUSC:
Theorem 1: In the random oracle model, we assume that we have a forger $\mathcal{F}$ who is able to win the EUF-I-IDBTUSC-aCMA unforgeability game with an advantage $\varepsilon \geq 10(q_{H_3}+1)(q_S+q_{H_3})/2^{\kappa}$ and asking at most q_{H_1} identity hashing queries, q_E key extraction queries, q_{H_3} H_3 queries and q_S signcryption queries. Then there exists an algorithm $\mathcal{C}$ which can solve the CDH problem with advantage $\varepsilon' \geq 1/9$.

Proof: We use *Forking Lemma* to prove the unforgeability of the scheme.

The forking lemma essentially says the following: Consider a signature scheme producing signatures of the form m, σ_1, h, σ_2 where each of σ_1, h, σ_2 corresponds to one of the three phases of some honest-verifier zero-knowledge identification protocol i.e., σ_1 is a commitment by the prover/signer, $h = H[m, \sigma_1]$ serves to simulate a random challenge by the verifier, and σ_2 is the prover/signer's response to the challenge. Suppose that $\mathcal{F}$ is an adaptive CMA existential forger, who makes μ_S signature queries and μ_R random oracle queries, and forges a signature m, σ_1, h, σ_2 in time τ with probability $\varepsilon = 10(\mu_S+1)(\mu_S+\mu_R)/2^n$. If the triples σ_1, h, σ_2 can be perfectly simulated without knowing the private key (e.g., by manipulating the random oracles instead), then there exists an algorithm $\mathcal{A}'$ that, using $\mathcal{F}$ as a subroutine, produces two valid signatures m, σ_1, h, σ_2 and $m, \sigma_1, h', \sigma_2'$ such that $h \neq h'$, in expected time $\tau' \leq 120686\mu_R\tau/\varepsilon$.

First we show that our scheme I-IDBTUSC produces signature of the form σ_1, h, σ_2 which corresponds to the required three-phase honest-verifier zero-knowledge identification protocol. In the improved scheme I-IDBTUSC $\sigma_1 = k_1 = xP$ being the prover's commitment, $h = r = H_3(c, \sigma_1, Y, Q_A, Q_{\mathbb{B}})$ a hash value substituted for the verifier's challenge, and $\sigma_2 = V$ the prover's response.

The rest of the proof then consists of the following steps:

- A simulation step, in which we show how to simulate the signature without knowing the secret key of the sender. By Forking Lemma, this gives us a machine $\mathcal{A}'$ that produces two valid signatures (σ_1, h, σ_2) and $(\sigma_1, h', \sigma_2')$ with $h \neq h'$.

- A reduction step, in which we show how to solve the CDH problem by interacting with the machine $\mathcal{A}'$

The simulation of the real attack environment is shown below:

Let $\mathcal{C}$ be a challenger who is given the instance of the CDH problem, $P, \alpha P, \beta P$. His goal is to compute $\alpha\beta P$. Suppose there exists a forger $\mathcal{F}$, who can existentially forge the I-IDBTUSC scheme, now $\mathcal{C}$ runs $\mathcal{F}$ as a subroutine and act as $\mathcal{F}$'s challenger in the forgery game and using $\mathcal{F}$, solves the CDH problem instance with non-negligible advantage in polynomial time.

Assumptions: The following assumption is made:

- $\mathcal{F}$ queries $H_1(ID_i)$ before ID_i is used in any key extraction, signcryption and unsigncryption queries.

Here we do not need to provide the forger $\mathcal{F}$ with the unsigncryption share oracle because due to the notion of insider security, we have assumed that he can have access to the private keys of all the receiver group in the system, so providing him with the unsigncryption shares would not give him any advantage to forge the signature.

Let there be q_{H_1} identity hash queries, out of which q_G are the identity queries for groups in the system and q_I are the identity queries for individual users in the system. $\mathcal{C}$ chooses θ randomly from q_I i.e. $\theta \in_R \{1, 2, ..., q_I\}$. Now whenever $i = \theta$, ID_i is referred as ID^*. It sets $P_{pub} = \alpha P$ and $Q^* = H_1(ID^*) = \beta P$, thus $D^* = \alpha\beta P$ and $\mathcal{C}$ does not know D^*. These values will be used in the challenge phase.

Thus the goal of the challenger is to compute $D^* = \alpha\beta P$ which is the solution to the CDH problem.

Initial: $\mathcal{C}$ gives $\mathcal{F}$ the public parameters $(\mathbb{G}_1, \mathbb{G}_2, \delta, \hat{e}, P, P_{pub}, H_1, H_2, H_3, H_4, H_5, \mathcal{E}, \mathcal{D})$ where he sets $P_{pub} = \alpha P$, α is unknown to $\mathcal{C}$. This value simulates the master key s in the game.

Training Phase: The forger $\mathcal{F}$ queries $\mathcal{C}$ for the random oracles H_1, H_2, H_3, H_4 *and* H_5. As these answers are randomly generated $\mathcal{C}$ keeps the lists L_1, L_2, L_3, L_4 *and* L_5 to maintain consistency and to avoid collision. The queries to the random oracles $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$, $\mathcal{O}_{H_4}$, $\mathcal{O}_{H_5}$, $\mathcal{O}_{Key-Extract}$, $\mathcal{O}_{Signcryption}$, *and* $\mathcal{O}_{Unsigncryption}$ are answered as follows:

-Oracle $\mathcal{O}_{H_1}(ID)$: For a query $H_1(ID_i)$, if there exists a tuple (ID_i, d_e) then $\mathcal{C}$ returns d_eP as the answer else chooses $d_e \in_R \mathbb{Z}_q^*$, returns d_eP and updates the list L_1 with the tuple (ID_i, d_e). If $ID_i = ID^*$ it returns βP.

-Oracle $\mathcal{O}_{H_2}(g_e, k_1, ID_i, ID_j)$: For a query $H_2(g_e, k_1, Q_i, Q_j)$, if there exists a tuple $(g_e, k_1, Q_i, Q_j, k_2)$

then return k_2 as the answer else chooses $k_2 \in_R \{0,1\}^\delta$ such that no other tuple contains the same k_2 and returns k_2. It then updates the list L_2 with the tuple $(g_e, k_1, Q_i, Q_j, k_2)$.

-Oracle $\mathcal{O}_{H_3}(c, k_1, Y, ID_i, ID_j)$*:* For a query $H_3(c, k_1, Y, Q_i, Q_j)$, if there exists a tuple (c, k_1, Y, Q_i, Q_j, r) then return r as the answer else chooses $r \in_R \mathbb{Z}_q^*$ such that no other tuple contains the same r and then returns r. It then updates the list L_3 with the tuple (c, k_1, Y, Q_i, Q_j, r).

-Oracle $\mathcal{O}_{H_4}(\eta, \mu, \omega)$*:* For a query $H_4(\eta_e, \mu_e, \omega_e)$, if there exists a tuple $(\eta_e, \mu_e, \omega_e, v)$ then returns v as the answer else chooses $v \in_R \mathbb{Z}_q^*$, returns v and updates the list L_4 with the tuple $(\eta_e, \mu_e, \omega_e, v)$.

-Oracle $\mathcal{O}_{H_5}(m, \rho)$*:* For a query $H_5(m, \rho)$, if there exists a tuple (m, ρ, h) then returns h as the answer else chooses $h \in_R \mathbb{Z}_q^*$, such that no other tuple contains the same h, returns h and updates the list L_5 with the tuple (m, ρ, h).

-Oracle $\mathcal{O}_{Key-Extract}(ID_i)$*:* For a query Extract($ID_i$):

1) If $ID_i = ID^*$, then $\mathcal{C}$ fails and aborts.
2) If $ID_i \neq ID^*$, then L_1 contains a pair (ID_i, d_e). So it returns $D_i = d_e P_{pub} = d_e \alpha P = \alpha Q_i$.

-Oracle $\mathcal{O}_{Signcryption}(m, ID_i, ID_j)$*:* For a signcryption query on message m, sender's identity ID_i and receiver's identity ID_j, the challenger $\mathcal{C}$ computes the signcryption σ as follows:

1) If $ID_i \neq ID^*$. In this case $\mathcal{C}$ knows the secret private key of the sender. He answers the query by running the algorithm **Signcryption**(m, D_i, ID_j). While answering the query he updates the lists
   $L_2 : \{\hat{e}(P_{pub}, Q_j)^h, k_1, Q_i, Q_j, k_2\}$
   $L_3 : \{c, k_1, Y, Q_i, Q_j, r\}$
   $L_5 : \{m, \rho, h\}$

2) If $ID_i = ID^*$, in this case $\mathcal{C}$ has to simulate Signcryption as follows:
   $\mathcal{C}$ chooses $r \in_R \mathbb{Z}_q^*, V \in_R \mathbb{G}_1, k_1 \in_R \mathbb{G}_1,$ *and* $\rho \in_R \mathbb{Z}_q^*$, and computes:
   a) $h = H_5(m, \rho)$ and updates the list L_5 with (m, ρ, h).
   b) $Y = hP$.
   c) $g_e = \hat{e}(Q_j, P_{pub})^h$.
   d) $k_2 = H_2(g_e, k_1, Q_i, Q_j)$ and updates the list L_2 with $(g_e, k_1, Q_i, Q_j, k_2)$.
   e) $c = \mathcal{E}_{k_2}(m \parallel \rho)$.

   He checks if L_3 contains a tuple $(c, k_1, Y, Q_i, Q_j, r')$ with $r' \neq r$, if it does then $\mathcal{C}$ repeats the above procedure with another random quadruple (r, V, k_1, ρ).

He then returns to the forger the signcryption $\sigma = (c, k_1, V, Y)$.

-Oracle $\mathcal{O}_{Unsigncryption}(\sigma, ID_i, ID_j)$*:* For the unsigncryption query of (c, k_1, V, Y) from the sender identity ID_i to the receiver ID_j, the challenger first verifies the signature by running the **SignVer** algorithm. $\mathcal{C}$ knows the private keys of all the groups. Therefore, if signature passes the verification then challenger unsigncrypts σ by running the Unsigncryption Share-Generation algorithm and then combines the shares using ShareCombine algorithm and returns m from $m \parallel \rho = \mathcal{D}_{k_2}(c)$, iff $Y \stackrel{?}{=} (H_5(m, \rho)).P$, otherwise returns *Invalid Signcryption*.

Forgery: Eventually $\mathcal{F}$ outputs a forged signcryption $\sigma^* = (c^*, k_1^*, V^*, Y^*)$ for some message m^* and the identities ID_A and $ID_{\mathbb{B}}$, where ID_A is the target identity chosen by $\mathcal{F}$ on which he wants to be challenged. Now, if the target identity ID_A chosen by the forger is not the same as chosen by the challenger ID^*, then $\mathcal{C}$ fails the simulation and aborts. In other case $\mathcal{C}$ checks for the validity of the forged message. The output signcryption is a valid forgery if the triple $(\sigma^*, ID_A, ID_{\mathbb{B}})$ was not the output of any previous queries to the *Signcryption Oracle* with m^* as the message and the private key of ID_A was not queried during the *Training Phase*. $\mathcal{F}$ wins the game if the result of **SignVer** is not $\perp$ symbol and the message unsigncrypted passes the validity check in the end.

Now we show the reduction to solve the CDH problem by constructing a Las Vegas Machine as follows:

If $\mathcal{F}$ is a sufficiently efficient forger in the above interaction, then following forking lemma in [12] we can construct a Las Vegas machine $\mathcal{A}'$ that outputs two signed messages $((ID_i, m,), k_1, V, Y)$ and $((ID_i, m), k_1, V', Y')$ with $r \neq r'$, where r and r' is computed using the publicly known values as $r = H_3(c, k_1, Y, Q_A, Q_{\mathbb{B}})$, and $r' = H_3(c, k_1, Y', Q_A, Q_{\mathbb{B}})$ and the same commitment x used in $k_1 = xP$.

Remark: We are implicitly coalescing the signing identity ID_i and the message m into a "generalised" forged message (ID_i, m) for the purpose of applying the forking lemma. This is in order to hide the identity-based aspect of the *EUF-I-IDBTUSC-aCMA* attack, and simulate the setting of an (identity-less) adaptive-CMA existential forgery for which the forking lemma is proven in [12].

Thus, given $\mathcal{F}$, we derive a machine $\mathcal{A}'$, and use it to construct a second machine $\mathcal{B}$ which is the reduction for the CDH problem. $\mathcal{B}$ proceeds as follows.

1) $\mathcal{B}$ runs $\mathcal{A}'$ to obtain two distinct forgeries $((ID_i, m), k_1, V, Y)$ and $((ID_i, m), k_1, V', Y')$ and computes $r = H_3(c, k_1, Y, Q_A, Q_{\mathbb{B}})$ and $r' = H_3(c, k_1, Y', Q_A, Q_{\mathbb{B}})$.
2) Now $\mathcal{B}$ unsigncrypts and obtains the signatures: $V = xP_{pub} - rD_A$ and $V' = xP_{pub} - r'D_A$, subtracts both the equations and derives the value of abP from $(r' - r)^{-1}(V - V') = D_A$ as $(r' - r)^{-1}(V - V') = \alpha\beta P$.

Note: In forking lemma we use the oracle replay technique, where the same random tape is used by the forger $\mathcal{F}$, but different oracles are used to answer the queries.

Thus, the challenger $\mathcal{C}$ obtains the solution $\alpha\beta P$ to the CDH problem instance $(P, \alpha P, \beta P)$, using a polynomial time forger *EUF-I-IDBTUSC-aCMA*.

Success Probability: The challenger $\mathcal{C}$ has the same advantage in solving the CDH problem as the forger $\mathcal{F}$ has in forging a valid signcryption. So, if there exists a forger who can forge a valid signcryption with non-negligible advantage, that means there exists an algorithm to solve the CDH problem with non-negligible advantage.

Based on the bound from the forking lemma in [12] if $\mathcal{F}$ succeeds in $time \leq t$ with probability $\varepsilon \geq 10(q_{H_3}+1)(q_S+q_{H_3})/2^{\kappa}$, then $\mathcal{B}$ can solve the CDH problem in expected time $\tau \leq 23.q_{H_3}t/\varepsilon$ with an advantage $\varepsilon' \geq 1/9$.

As per the advantage given above, with very negligible probability one can solve the CDH problem, therefore no forger can forge a valid signcryption with non-negligible advantage. Hence, our improved scheme is secure against any *EUF-I-IDBTUSC-aCMA* attack.

2) Confidentiality Proof of I-IDBTUSC : Theorem 2: In the random oracle model (where the hash functions are modelled as random oracles), we assume that we have an adversary $\mathcal{A}$ who is able to win the IND-I-IDBTUSC-aCCA2 confidentiality game (i.e. $\mathcal{A}$ is able to distinguish signcryptions given by the challenger), with an advantage ε and asking at most q_{H_1} identity hashing queries, at most q_E key extraction queries and at most q_{H_2} H_2 queries. Then, there exists an algorithm $\mathcal{C}$ that can solve the CBDH problem with an advantage

$$\text{Adv}(\mathcal{C}) = (\frac{\varepsilon+1}{2} - \frac{1}{2^{\delta}}).(\frac{1}{q_{H_2}}).(\frac{1}{q_{H_1}}).$$

where advantage of $\mathcal{C}$ is defined as

$$Adv_{\mathcal{A}}^{CBDH} = Pr\left[\mathcal{A}(P, \alpha P, \beta P, \gamma P) = \hat{e}(P,P)^{\alpha\beta\gamma} \mid \alpha, \beta, \gamma \in \mathbb{Z}_q^*\right]$$

Proof: Let $\mathcal{C}$ be a challenger who is given the instance of the CBDH problem, $(P, \alpha P, \beta P, \gamma P)$. His goal is to compute $\hat{e}(P,P)^{\alpha\beta\gamma}$. Suppose there exists an adversary $\mathcal{A}$, who can break the confidentiality of the I-IDBTUSC scheme, now $\mathcal{C}$ runs $\mathcal{A}$ as a subroutine and act as $\mathcal{A}$'s challenger in the confidentiality game and using $\mathcal{A}$ solves the CBDH problem instance with non-negligible advantage in polynomial time.

Assumptions: The following assumptions are made:

- $\mathcal{A}$ queries $H_1(ID_i)$ before ID_i is used in any key extraction, signcryption and unsigncryption queries.

- $\mathcal{A}$ can corrupt at most $t-1$ unsigncryption members during the attack. That is he obtains $t-1$ private keys $\{\Delta_i\}_{(1\leq i\leq t)}$ of the corrupted unsigncryption members of the target identity after the challenge phase.

- $\mathcal{A}$ is given an unsigncryption share oracle $\mathcal{O}_{Unsigncryption-Share}$ in the second phase of his queries in the game, from which he may ask for the unsigncryption shares of the uncorrupted members of the target group.

Here we have given the unsigncryption share oracle only in the second phase, because in the challenge phase the adversary specifies the target identity of the recipient group, so he would be needing the unsigncryption shares of the uncorrupted members only after that. In the first phase his queries would only be for the complete unsigncryption of the signcryption not the shares.

Let there be q_{H_1} identity hash queries, out of which q_G are the identity queries for groups in the system and q_I are the identity queries for individual users in the system. $\mathcal{C}$ chooses θ randomly from q_G queries i.e. $\theta \in_R \{1, 2, ..., q_G\}$. Now, whenever $i = \theta$, ID_i is referred as ID^*. He sets $P_{pub} = \alpha P$; $Q^* = H_1(ID^*) = \beta P$ and $Y^* = \gamma P$, thus $D^* = \alpha\beta P$ but $\mathcal{C}$ does not know D^*. These values will be used in the challenge phase. Thus the solution to the CBDH problem is $\hat{e}(D^*, Y^*) = \hat{e}(P,P)^{\alpha\beta\gamma}$.

Initial: $\mathcal{C}$ gives $\mathcal{A}$ the public parameters $(\mathbb{G}_1, \mathbb{G}_2, \delta, \hat{e}, P, P_{pub}, H_1, H_2, H_3, H_4, H_5, \mathcal{E}, \mathcal{D})$ where he sets $P_{pub} = \alpha P$, α is unknown to $\mathcal{C}$. This value 'α' simulates the master key 's' in the game.

Phase 1: The adversary $\mathcal{A}$ queries $\mathcal{C}$ for the random oracles H_1, H_2, H_3, H_4 *and* H_5. As these answers are randomly generated $\mathcal{C}$ keeps the lists L_1, L_2, L_3, L_4 *and* L_5 to maintain consistency and to avoid collision. The queries to the random oracles $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$, $\mathcal{O}_{H_4}$, $\mathcal{O}_{H_5}$, *and* $\mathcal{O}_{Key-Extract}$ are answered in the similar manner as shown in the unforgeability game, so can be referred in the previous section and the queries to the random oracles $\mathcal{O}_{Signcryption}$, *and* $\mathcal{O}_{Unsigncryption}$ are answered as follows:

-**Oracle** $\mathcal{O}_{Signcryption}(m, ID_i, ID_j)$**:** For a signcryption query on message m and identities ID_i *and* ID_j, the challenger $\mathcal{C}$ can compute the signcryption σ by running the algorithm **Signcryption**(m, D_i, ID_j), because he knows the private keys of all the individual members in the system. While answering the query he updates the lists
$L_2 : \{\hat{e}(P_{pub}, Q_j)^h, k_1, Q_i, Q_j, k_2\}$
$L_3 : \{c, k_1, Y, Q_i, Q_j, r\}$
$L_5 : \{m, \rho, h\}$

He then returns to the adversary the signcryption $\sigma = (c, k_1, V, Y)$.

-**Oracle** $\mathcal{O}_{Unsigncryption}(\sigma, ID_i, ID_j)$**:** For the unsigncryption query of (c, k_1, V, Y) from the sender identity ID_i to the receiver ID_j, the challenger first verifies the signature by running the **SignVer** algorithm, if it passes the verification then challenger unsigncrypts the ciphertext c as follows:

- If $ID_j \neq ID^*$, in this case $\mathcal{C}$ knows the secret key of the receiver group, so he unsigncrypts σ by running the Unsigncryption Share-Generation algorithm and then

combines the shares using ShareCombine algorithm and returns m from $m \parallel \rho = \mathcal{D}_{k_2}(c)$, iff $Y \stackrel{?}{=} (H_5(m,\rho)).P$, otherwise returns *Invalid Signcryption.*

- If $ID_j = ID^*$, then $\mathcal{C}$ searches L_2 for a tuple $(g_e, k_1, Q_i, Q_j, k_2)$, and store all such tuples in list L_7. Now from all these tuples k_2 is retrieved to unsigncrypt c to obtain $m \parallel \rho = \mathcal{D}_{k_2}(c)$, and checks if $Y \stackrel{?}{=} (H_5(m,\rho))P$. For the tuples that passes the validity, checks whether $g_e \stackrel{?}{=} \hat{e}(Q_j, P_{pub})^{H_5(m,\rho)}$, if it does then returns m obtained from this g_e to the adversary, else if no such tuple passes the validity then returns *Invalid Signcryption.*

Challenge: Once $\mathcal{A}$ decides that phase one is over, he outputs two messages m_0 and m_1 of equal length and the identities of the sender as ID_A and of the receiver group as $ID_{\mathbb{B}}$, and this $ID_{\mathbb{B}}$ is the target identity on which he wants to be challenged.

Now, if the challenge identity $ID_{\mathbb{B}}$ chosen by the adversary is not the same identity ID^* set by the challenger, then $\mathcal{C}$ fails and aborts the game, because the simulation will not be helpful for him to solve the CBDH problem. Otherwise if $ID_{\mathbb{B}} = ID^*$, then $\mathcal{C}$ gives the signcryption σ^* by flipping a fair coin $b \in_R \{0,1\}$, and chooses $c^* \in_R \{0,1\}^*$; $r^* \in_R \mathbb{Z}_q^*$; $k_1^* \in_R \mathbb{G}_1$ and $V^* \in_R \mathbb{G}_1$, sets $Y^* = \gamma P$ and verifies if L_3 contains $(c^*, k_1^*, Y^*, Q_A, Q_{\mathbb{B}}, r')$, where $r' \neq r^*$, if so then choose another random quadruple (c^*, r^*, k_1^*, V^*) and then returns to $\mathcal{A}$ the challenge signcryption $\sigma^* = (c^*, k_1^*, V^*, Y^*)$.

Now, we assume that the adversary can corrupt atmost $t-1$ members in the target recipient group and without the loss of generality we assume that these are the first $t-1$ members. Now $\mathcal{C}$ picks $\Delta_i \in_R \mathbb{G}_1$ for $i = 1, 2, ..., t-1$ and computes $n-t+1$ verification keys of other members in $G_{\mathbb{B}}$ as:

$$\tau_j = \hat{e}(Q_{\mathbb{B}}, P_{pub})^{N_{0,j}} \prod_{k=1}^{t-1} \hat{e}(\Delta_k, P)^{N_{j,k}}$$

where $t \leq j \leq n$ and $N_{j,k}$ denotes the Lagrange coefficient $N_{j,k} = \prod_{l=0, l\neq k}^{t} \frac{j-l}{k-l}$. $\mathcal{C}$ then sends Δ_i of the corrupted members and τ_j of the uncorrupted members to the adversary $\mathcal{A}$. He then stores these values in the list L_6.

Phase 2: Now $\mathcal{A}$ performs second series of queries to the oracles treated in the same way as in the first phase. In this phase he is given one more oracle for querying the unsigncryption shares of the uncorrupted members. The queries to this oracle $\mathcal{O}_{Unsigncryption-Share}$ is answered as follows:

-**Oracle** $\mathcal{O}_{Unsigncryption-Share}(\sigma, ID_i, ID^*, t)$**:** For the unsigncryption share query of the t^{th} member for the signcryption (c, k_1, V, Y) from ID_i to ID^*, the challenger first verifies the signature by running the *SignVer* algorithm, if it passes the verification then challenger first obtains the private key-shares of the $t-1$ corrupted members of the group G^* from the list L_6 and then calculates the t^{th} unsigncryption share as:

First $\mathcal{C}$ searches L_2 for a tuple $(g_e, k_1, Q_i, Q_j, k_2)$, and store all such tuples in list L_7. Now from all these tuples k_2 is retrieved to unsigncrypt c to obtain $m \parallel \rho = \mathcal{D}_{k_2}(c)$, and checks if $Y \stackrel{?}{=} (H_5(m,\rho))P$. For the tuples that passes the validity, check whether $g_e \stackrel{?}{=} \hat{e}(Q_j, P_{pub})^{H_5(m,\rho)}$. It repeats this process until a valid tuple is found, else when all such tuples are exhausted then returns *Invalid Signcryption*. If any tuple passes the validity, then $\mathcal{C}$ computes:

1) $\eta_t = g_e^{N_{0,t}} \prod_{i=1}^{t-1} \hat{e}(\Delta_i, Y)^{N_{t,i}}$, where $\Delta_i' s$ are the private key-shares of the $t-1$ corrupted members, $N_{t,i} = \prod_{l=0}^{t-1} \frac{t-l}{i-l}$ denotes the Lagrange coefficient and g_e is the first entry in the found tuple.
2) Chooses random $v_t \in_R \mathbb{Z}_q^*$ and $W_t \in_R \mathbb{G}_1$ and computes: $\mu_t = \hat{e}(W_t, Y)/\eta_t^{v_t}$ and $\omega_t = \hat{e}(W_t, P)/\tau_t^{v_t}$.

then, checks the list L_4 with $(\eta_t, \mu_t, \omega_t, v_t')$, such that $v_t' \neq v_t$. If such a tuple exists then repeats the above procedure with another random tuple (v_t, W_t). If the condition satisfies then update the list L_4 and return to the adversary $\mathcal{A}$ the unsigncryption share $\sigma_t = (t, \eta_t, \mu_t, \omega_t, v_t, W_t)$.

Since the challenge ciphertext given is not a valid one, therefore when the adversary calculates the correct value in the unsigncryption key, he will query H_2, but the challenger will not be able to give him the correct answer. The adversary now knows that the challenger is fooling him and *aborts* the game.

Success Probability: Now we analyse $\mathcal{C}'s$ success probability in the above confidentiality game. In the above simulation, all responses to the random hash oracles, signcryption and unsigncryption oracles are randomly distributed simulating the real experiment, as $\mathcal{A}$ is unaware of the secret parameters and identity guessed by the challenger. Whenever $\mathcal{A}$ chooses the same target identity guessed by $\mathcal{C}$ and with the event that $\mathcal{C}$ never aborts then the simulation provided by $\mathcal{C}$ is indistinguishable from a real attack scenario to the adversary $\mathcal{A}$. Also the challenge signcryption σ^* given to $\mathcal{A}$ passes the sign verification procedure so adversary is unable to know the invalidity of the signcryption, until he computes the unsigncryption key, in which case he will abort. By definition the advantage of such adversary $\mathcal{A}$ is given as $Adv(\mathcal{A}) = \mid 2P[b' = b] - 1 \mid$, ie $P[b' = b] = (\varepsilon+1)/2$.

Now we analyse the events when the above experiment fails to simulate the real attack environment. $\mathcal{C}$ aborts the experiment in the following scenario:

1) When $\mathcal{A}$ makes a key extraction query on ID^* in the first phase.

2) When $\mathcal{A}$ do not choose ID^* as the target recipient in the challenge phase.
3) When $\mathcal{C}$ rejects a valid signcryption during the unsigncryption query, even if the signcryption is valid.

To calculate that $\mathcal{C}$ aborts during the simulation, suppose there are q_{H_1} H_1 queries, q_E key extraction queries and q_{H_2} H_2 queries.

The probability that $\mathcal{C}$ does not abort in first phase is the probability that it does not make any key extraction queries on ID^* for the q_E queries, which is $(\frac{q_{H_1}-1}{q_{H_1}}).(\frac{q_{H_1}-2}{q_{H_1}-1})...(\frac{q_{H_1}-q_E}{q_{H_1}-q_E+1}) = (\frac{q_{H_1}-q_E}{q_{H_1}})$. Further, with a probability exactly $(\frac{q_{H_1}-q_E-1}{q_{H_1}-q_E}).(\frac{1}{q_{H_1}-q_E-1}) = (\frac{1}{q_{H_1}-q_E})$, $\mathcal{A}$ chooses to be challenged on the identity ID^*, where $(\frac{q_{H_1}-q_E-1}{q_{H_1}-q_E})$ being the probability of choosing an identity other than ID^* as the sender's and the later being the probability of choosing ID^* as the receiver's. Hence the probability that $\mathcal{A}'s$ response is helpful to $\mathcal{C}$ is $(\frac{1}{q_{H_1}})$.

Now the case when $\mathcal{A}$ makes random guess must also be considered because the random guess of the encryption key will not leave any entry in L_2 and will not help $\mathcal{C}$ in solving the problem. Thus, the probability of randomly guessing the encryption key is $(\frac{1}{2^\delta})$.

The third case where the simulation may fail will never happen because the adversary would never be able to find out whether the signcryption is valid or not, because he has only the *SignVer* algorithm to verify the signature of the sender not the validity of the signcryption, and the case when he knows the signcryption is valid, is when he has obtained it from the signcryption oracle, in that case the correct unsigncryption is provided with the help of the lists. Thus the probability with which a valid signcryption is rejected will never happen.

Now, we have assumed that the adversary $\mathcal{A}$ is capable of breaking the confidentiality of the system, thus he is able to calculate the correct unsigncryption key $k_2 = H_2(\hat{e}(D^*, Y^*), k_1^*, Q_A, Q^*)$. But since the ciphertext c^* formed is randomly chosen from $\{0,1\}^*$, when querying for the hash value H_2 he will be answered with some random k_2, using which for unsigncryption would not yield either m_0 or m_1. Thus now the challenger is caught and $\mathcal{A}$ aborts the game. Challenger is now sure that $\mathcal{A}$ must have queried H_2 with $(\hat{e}(D^*, Y^*), k_1^*, Q_A, Q^*)$. Thus, at the end of the game, $\mathcal{C}$ extracts the solution to the CBDH problem by searching the list L_2 with conditional probability $(\frac{1}{q_{H_2}})$.

Taking into account all the probabilities that $\mathcal{C}$ will not fail its simulation, the probability that $\mathcal{A}$ chooses to be challenged on the identity ID^*, and the probability that $\mathcal{A}$ wins the *IND-I-IDBTUSC-aCCA2* game, the value of Adv($\mathcal{C}$) for solving the CBDH problem instance is calculated as:

$$\text{Adv}(\mathcal{C}) = (\frac{\varepsilon+1}{2} - \frac{1}{2^\delta}).(\frac{1}{q_{H_2}}).(\frac{1}{q_{H_1}}).$$

VI. Conclusion

In this paper, we have cryptanalysed the identity based threshold unsingcryption schemes by Fagen Li et al. in [9] and Fagen Li et al. in [10]. We showed that both these schemes do not meet the stringent requirements of insider security and demonstrate attacks on both confidentiality and unforgeability. We have also proposed an improved identity based threshold unsigncryption scheme and gave the formal proof of security in a new stronger security model in the random oracle model. Thus our improved scheme remains the only provably secure identity based threshold unsingcryption scheme.

References

[1] Jun Li Aihong Ping, Minghui Zheng. A threshold subliminal channel for manet using publicly verifiable hybrid signcryption. In *ISW*, pages 218–232, 1994.

[2] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[3] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In *Public Key Cryptography, PKC-2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer, 2002.

[4] Joonsang Baek and Yuliang Zheng. Identity-based threshold decryption. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography, PKC-2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2004.

[5] Xavier Boyen. Multipurpose identity-based signcryption (a swiss army knife for identity-based cryptography). In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2003.

[6] Peng Changgen and Li Xiang. Threshold signcryption scheme based on elliptic curve cryptosystem and verifiable secret sharing. In *Wireless Communications, Networking and Mobile Computing, WCNM - 2005*, pages 1182–1185, 2005.

[7] Sherman S. M. Chow, Siu-Ming Yiu, Lucas Chi Kwong Hui, and K. P. Chow. Efficient forward and provably secure id-based signcryption scheme with public verifiability and public ciphertext authenticity. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 352–369. Springer, 2003.

[8] Ratna Dutta, Rana Barua, and Palash Sarkar. Pairing-based cryptographic protocols : A survey. Cryptology ePrint Archive, Report 2004/064, 2004. http://eprint.iacr.org.

[9] Fagen Li, Juntao Gao, and Yupu Hu. Id-based threshold unsigncryption scheme from pairings. In *Information Security and Cryptology, CISC 2005*, volume 3822 of *Lecture Notes in Computer Science*, pages 242–253. Springer, 2005.

[10] Fagen Li, Xiangjun Xin, and Yupu Hu. Id-based signcryption scheme with (t,n) shared unsigncryption. In *International Journal of Network Security*, volume 3, number 2, pages 155–159, 2006.

[11] John Malone-Lee. Identity-based signcryption. Cryptology ePrint Archive, Report 2002/098, 2002. http://eprint.iacr.org/.

[12] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, volume: 13(number: 3):361–396, 2000.

[13] S. Sharmila Deva Selvi, S. Sree Vivek, Shilpi Nayak, and C. Pandu Rangan. Breaking and building of threshold signcryption schemes. In *To appear in Inscrypt - 2009*, Lecture Notes in Computer Science. Springer, 2009.

[14] S. Sharmila Deva Selvi, S. Sree Vivek, C. Pandu Rangan, and N. Jain. Cryptanalysis of li et al.'s identity-based threshold signcryption scheme. In Cheng-Zhong Xu and Minyi Guo, editors, *EUC (2)*, pages 127–132. IEEE Computer Society, 2008.

[15] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[16] Yuliang Zheng. Digital signcryption or how to achieve cost (signature and encryption) $<<$ cost (signature) + cost(encryption). In *Advances in Cryptology - Crypto'97*, Lecture Notes in Computer Science, pages 165–179. Springer, 1997.

[17] Z.Zhang, C.Mian, and Q.Jin. Signcryption scheme with threshold shared unsigncryption preventing malicious receivers. In *TENCON'02*, volume 1. IEEE, 2002.