Breaking and Building of Group Inside Signature

S. Sree Vivek⋆, S. Sharmila Deva Selvi, S.Gopinath, C. Pandu Rangan⋆

Theoretical Computer Science Lab,
Department of Computer Science and Engineering,
Indian Institute of Technology Madras,
Chennai, India-600036

Abstract. Group Inside Signature (GIS) is a signature scheme that allows the signer to designate his signature to be verified by a group of people, so that members other than the designated group cannot verify the signature generated by him. In Broadcast Group Oriented Signature (BGOS), an user from one group can designate his signature to be verified by members of other group. The GIS and BGOS schemes [5], [6] and [7] which we consider are certificateless schemes. An Adaptable Designated Group Signature (ADGS), is one in which an user can designate his signature to be verified by a selected set of members who are from different groups. The ADGS scheme [8] which we consider here is an identity based scheme. In this paper, we present the cryptanalysis of four schemes that appeared in [5], [6], [7] and [8]. We show that, both GIS schemes [5], [6] and BGOS scheme [7] suffers from Type-I and Type-II vulnerabilities and ADGS [8] is universally forgeable. We also present a new scheme for ADGS (N-ADGS) and proved its security in the random oracle model. The existing model for ADGS did not consider unlinkability which is one of the key properties required for ADGS. We provide security model for unlinkability and also prove our scheme is unlinkable.

Keywords: Group Inside Signature, Broadcast Group Oriented Signature, Adaptable Designated Group Signature, Identity Based, Certificateless, Cryptanalysis.

1 Introduction

In 1984, Shamir [10] introduced the concept of identity based cryptography and proposed the first identity based signature scheme. The first practical identity based encryption scheme [2] was realized by Boneh-Franklin in 2001 using Weil pairing. Since then, pairing was used in designing various cryptosystems. The idea of identity based cryptography allows any arbitrary string that uniquely identifies the user to be used as his public key. Identity based cryptography serves as an efficient alternative to Public Key Infrastructure (PKI) based systems.

Certificateless encryption schemes and signature schemes were first defined and proposed by Al-Riyami and Paterson [1] in 2003. Definitions for certificateless encryption and signature schemes in [1] consists of seven algorithms. The simplified definition for certificateless signature schemes was proposed by Hu, Wong, Zhang and Deng in [3]. Certificateless cryptography is intended to solve the key escrow problem which is inherent in identity based cryptography, while at the same time, eliminates the use of certificates which were used in the conventional PKI. In a certificateless cryptosystem, the Key Generation Center (KGC) issues partial private key to the users whose identity is assumed to be unique in the system. The user also independently generates additional public/private key pair. Cryptographic operations such as signing can then be performed successfully only when both the user partial private key and the user generated private key are used. Knowing only one of them is not sufficient to carry out any cryptographic operations such as signing a message.

In general, digital signatures are publicly verifiable. Jackbson et.al (1996) [4] proposed a concept called Designated Verifier Signatures (DVS). In DVS, only a designated person can verify the signature signed by the signer. DVS achieves this property by providing ability to the designated verifier to simulate the signers signature. In the same year Jackbson et.al proposed a stronger notion called strong DVS (SDVS), in which the third party cannot even verify the validity of the signature since it involves the private key of the designated verifier. Even if the designated verifier gives his private key to a third party, the third party can only check the validity but cannot authenticate the signature.

Extending a single party verification scheme to a designated group verification scheme is a challenging problem. In practice, there may be different group models as discussed below.

⋆ Work supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation sponsored by Department of Information Technology, Government of India

First, in networks like Local Area Networks, all group members reside in a single network and no member of the group may hang outside network. Certificateless GIS schemes [5] and [6] provide solutions for designating a signature to be verified inside such a group.

Secondly, in distributed networks, the users of different companies or institutions naturally come under different work groups. If a member of one group wants to send a signed document to members of another group, BGOS [7] can be used. Moreover the signer wants to prevent the members outside the designated group from verifying the signature. The scheme in [7] focuses on this problem.

Finally, in distributed networks, a signer may want several members to verify his signature, no matter whether those members are in same or different groups. The signer wants to prevent the members outside the defined group from verifying the signature. This model can be visualized as a more generalized version of the previous two models. ADGS scheme in [8] focuses on this problem. In fact even if a designated verifier v_i belongs to a group say G, while v_i can verify the signature of the sender, other members of G cannot verify the signature.

1.1 Our Contribution

In this paper, we show that GIS in [5],[6] and BGOS in [7] are not secure against both Type-I and Type-II adversaries. We also show that the basic ADGS scheme [8] is universally forgeable. We present an extended security model for ADGS by adding the notion of *unlinkability*. This is done for the first time and also we propose a new scheme(New-ADGS) and prove its security formally in random oracle model.

2 Preliminaries

2.1 Bilinear Pairing

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

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

2.2 Computational Assumptions

In this section, we review the computational assumptions related to bilinear maps that are relevant to the protocols we discuss.

Bilinear Diffie-Hellman Problem (BDHP) Given $(P, aP, bP, cP) \in \mathbb{G}_1^4$ for unknown $a, b, c \in_R \mathbb{Z}_q^*$, the BDH problem in $\mathbb{G}_1$ is to compute $\hat{e}(P, P)^{abc}$.

Definition. The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the BDH problem in $\mathbb{G}_1$ is defined as

$$Adv_{\mathcal{A}}^{BDH} = Pr\left[\mathcal{A}(P, aP, bP, cP) = \hat{e}(P, P)^{abc} \mid a, b, c \in \mathbb{Z}_q^*\right]$$

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

Decisional Bilinear Diffie-Hellman Problem (DBDHP) Given $(P, aP, bP, cP, \alpha) \in \mathbb{G}_1^4 \times \mathbb{G}_2$ for unknown $a, b, c \in \mathbb{Z}_q^*$, the DBDH problem in $\mathbb{G}_1$ is to decide if $\alpha = \hat{e}(P, P)^{abc}$.

Definition. The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the DBDH problem in $\mathbb{G}_1$ is defined as $Adv_{\mathcal{A}}^{DBDH} = \left|Pr\left[\mathcal{A}(P, aP, bP, cP, \hat{e}(P, P)^{abc}) = 1\right] - Pr\left[\mathcal{A}(P, aP, bP, cP, \alpha) = 1\right]\right|$ The *DBDH Assumption* is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{DBDH}$ is negligibly small.

Computation Diffie-Hellman Problem (CDHP) Given $(P, aP, bP) \in \mathbb{G}_1^3$ for unknown $a, b \in \mathbb{Z}_q^*$, the CDH problem in $\mathbb{G}_1$ is to compute abP.

Definition. The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the CDH problem in $\mathbb{G}_1$ is defined as

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

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

2.3 Forgeries in Digital Signature schemes.

- A **total break results** in the recovery of the signers secret key.
- An **universal forgery attack** results in the ability to forge signatures for any message on behalf of user without knowing the private key of the user.
- A **selective forgery** attack results in a signature on a message of the adversary choice.
- An **existential forgery** merely results in some valid message/signature pair not already known to the adversary.

2.4 Adversarial models for GIS[5] and BGOS[7] Schemes.

Both GIS [5] and BGOS [7] are in the certificateless setting, therefore we consider both Type-I and Type-II adversaries. We explain the distinguishing abilities of these adversaries below.
Type I Adversary: This type of adversary named as $\mathcal{A}_I$ does not have access to the master private key, but $\mathcal{A}_I$ has the ability to replace the public key of any entity with a value of its choice. The objective behind the definition of this adversary is to capture the attacks by entities other than the KGC.
Type II Adversary: This type of adversary named as $\mathcal{A}_{II}$ has access to the master private key. However, $\mathcal{A}_{II}$ cannot perform public key replacement. This adversary is defined to capture the attacks by dishonest KGC.

Now, we present the security model for existential unforgeability of a certificateless signature (CLS) scheme under chosen message attack for both Type-I and Type-II attacks.

Type-I Adversary($\mathcal{A}_I$).

- **Initialization:** The Challenger $\mathcal{C}$ runs the **Setup** and generates a master private key *masterkey* and the public parameters *params*. $\mathcal{C}$ keeps the *masterkey* secret and gives *params* to the adversary. Now the adversary chooses a target identity ID^* and gives it to the challenger $\mathcal{C}$. $\mathcal{A}_I$ is supposed to generate a valid forgery for the target identity ID^* on some message and it is not allowed to query partial private key for target identity ID^*. Note that $\mathcal{A}_I$ does not know the *masterkey*.
- **Training -Phase:** $\mathcal{A}_I$ interacts with $\mathcal{C}$ and is allowed to perform **Partial-Private-Key queries, Public-Key-Replacement queries** and **Sign queries** queries. These queries are answered by the corresponding oracles as explained below.
  - **Partial-Private-Key Oracle:** $\mathcal{A}_I$ can request the $PartialPrivateKey$ of the user whose identity is ID. $\mathcal{C}$ returns the $PartialPrivateKey$ S_{ID} to $\mathcal{A}_I$ as response.
  - **Public-Key-Replacement Oracle:** For any user whose identity is ID (including user with identity ID^*), $\mathcal{A}_I$ can choose a new *privatevalue* and compute the new public key $Q^{'}$. $\mathcal{A}_I$ can also set $Q^{'}$ as the new public key of the user by submitting $(Q^{'}, ID)$ to $\mathcal{C}$. $\mathcal{C}$ will replace the existing public key Q with $Q^{'}$ and after this replacement , only $Q^{'}$ is used as public key of ID.
  - **Sign Oracle:** $\mathcal{A}_I$ can ask the signature on a message m with ID as signer. $\mathcal{C}$ send to $\mathcal{A}_I$ a signature σ for the message m which is a valid signature under the current public key of ID.
- **Forgery:** Finally, $\mathcal{A}_I$ outputs a message/signature pair (m^*, σ^*) of the user whose identity is ID^*. This message/signature pair must satisfy the following requirements:
  - This signature should be valid under the current public key $Q^{'}$ of ID^*.
  - $\mathcal{A}_I$ should not have queried the **Partial-Private-Key** oracle for this user whose identity is ID^*.
  - σ^* should not be the output of any previous queries to the **Sign** oracle.

Type-II Adversary($\mathcal{A}_{II}$).

- **Initialization:** The challenger $\mathcal{C}$ runs the **Setup** and generates a master private key *masterkey* and the public parameters *params*. $\mathcal{C}$ gives the *masterkey*, *params* to $\mathcal{A}_{II}$. $\mathcal{A}_{II}$ chooses an user identity ID^* and gives it to $\mathcal{C}$. $\mathcal{A}_{II}$ is supposed to generate a valid forgery for the identity ID^* on some message. Note that $\mathcal{A}_{II}$ has access to *masterkey* but cannot replace the public key of the users.

- **Phase-I:** $\mathcal{A}_{II}$ interacts with $\mathcal{C}$ and can access only the **Sign** Oracle alone. Here $\mathcal{A}_{II}$ is prohibited to replace public key and $\mathcal{A}_{II}$ cannot request the $PartialPrivateKey$.
  - **Sign Oracle:** $\mathcal{A}_{II}$ can request a signature on a message m with ID as signer. $\mathcal{C}$ sends to $\mathcal{A}_{II}$ the signature σ for the message m.
- **Forgery:** Finally, $\mathcal{A}_{II}$ outputs a target message/signature pair (m^*, σ^*) of the user whose identity is ID^*. This message/signature pair must satisfy the following requirements:
  - This signature σ^* should pass the verification algorithm.
  - σ^* should not be the output of any previous queries to **Sign** oracle.

3 Review and Cryptanalysis of Certificateless Group Inside Signature [5].

In this section we review GIS scheme and show that the scheme does not resist both Type-I and Type-II attacks.

3.1 Review of GIS scheme [5].

Let $\mathbb{A}$= $\{a_1, a_2, ..., a_n\}$ be a group of n users. The set $\{ID_1, ID_2 ... ID_n\}$ is the corresponding identities of users $\{a_1, a_2, ..., a_n\}$. Let $a_i \in \mathbb{A}$ is the signer who wants to sign a message m, verifiable only by members of group $\mathbb{A}$.

- **Initialize:**
  Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of same prime order q. Let P be a generator of additive group $\mathbb{G}_1$ and $\mathbb{G}_2$ is a multiplicative group. Let H_0 and H_1 be cryptographic hash functions defined as H_0 :$\{0,1\}^* \rightarrow \mathbb{G}_1$ and H_1:$\{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$.
- **Key Generation:**
  To generate partial private keys for its members, the KGC performs the following:
  - Chooses k $\in_R \mathbb{Z}_q^*$ for the group $\mathbb{A}$ and
  - Computes $S_j = kH_0(ID_j)$, 1$\leq j \leq n$.
  - Sends S_j to a_j in a secure way.

  To generate public key $P_j = \langle P_{j1}, P_{j2} \rangle$ and private key D_j, user a_j, where 1$\leq j \leq n$, does the following:
  - Chooses $x_j \in_R \mathbb{Z}_q^*$ and computes $D_j = x_j S_j$.
  - Computes $P_{j1} = x_j H_0(ID_j)$, $P_{j2} = x_j P$.

  The user a_j takes D_j as its private key and $P_j = \{P_{j1}, P_{j2}\}$ as its public key, 1$\leq j \leq n$.
- **GIS Generation.**
  In order to sign a message m, that is verifiable by members of $\mathbb{A}$, a_i performs the following:
  - Chooses $r \in_R \mathbb{Z}_q^*$ and computes $U = rP_{i1}$.
  - Computes $h = H_1(m, U)$ and computes $V = (r + h)D_i$.
  - Broadcasts the signature σ as $\langle m, U, V \rangle$ to the group $\mathbb{A}$.
- **GIS Verification.**
  To verify the signature σ is sent by user a_i, whose identity is ID_i and public key is $P_i = \langle P_{i1}, P_{i2} \rangle$, the user a_j $\in \mathbb{A}$ performs the following checks:
  - Checks whether $e(P_{i1}, P) \stackrel{?}{=} e(P_{i2}, H_0(ID_i))$. If this does not hold, return error.
  - Checks the validity of the signature as $e(V, P_{j1}) \stackrel{?}{=} e(U, D_j)e(hP_{i1}, D_j)$, where $h = H_1(m, U)$. If both equations hold good then the user a_j accepts the signature σ on message m as valid.

3.2 Cryptanalysis of Certificateless Group Inside Signature (GIS) [5].

GIS scheme given in [5] allows the signer to designate his signature to be verified by a group of people who belong to the signer's group. Members other than the designated group should not be able to verify the signature generated by him. The scheme in [5] is not secure against Type-I and Type-II attacks.

Type-I Attack. On seeing a valid signature by an user on some message, anyone can commit a forgery on any message. During the unforgeability game between the challenger $\mathcal{C}$ and adversary $\mathcal{A}_I$, $\mathcal{C}$ gives $\mathcal{A}_I$ the public parameters *params* and $\mathcal{A}_I$ gives to $\mathcal{C}$ a target identity ID^*. $\mathcal{A}_I$ is supposed to generate a valid forgery for the target identity ID^* on some message and $\mathcal{A}_I$ is not allowed to query partial private key for the target identity ID^*. $\mathcal{A}_I$ interacts with $\mathcal{C}$ and access all the oracles with the restrictions given in the model. $\mathcal{A}_I$ can query signature on any message and user identity pair $\langle m, ID\rangle$. $\mathcal{A}_I$ can replace the public keys of suppose any user including user with identity ID^*. During the training-phase $\mathcal{A}_I$ receives a valid signature $\sigma = \langle m, U, V\rangle$ on a message m with target identity ID^* using the **Sign** oracle. Now we show how $\mathcal{A}_I$ can generate a valid signature σ^* on an arbitrary message m^* for the target identity ID^*, such that σ^* is not the output of previous queries to **Sign** oracle. This can be shown by the following computation done by $\mathcal{A}_I$

- Computes $U^* = U + hP_{i1}$ - $H_0(ID^*)$, where h= $H_1(m, U)$ computed from σ.
- Computes $h^* = H_1(m^*, U^*)$
- Replaces public keys of ID^* as $P^*_{i1} = \frac{1}{h^*}\ H_0(ID^*)$ and $P^*_{i2} = \frac{1}{h^*}\ P$.
- $V^* = V$.

Now we claim that $\sigma^* = \langle m^*, U^*, V^*\rangle$ is a valid signature on the message m^* by the user with identity ID^* (with respect to its newly replaced public key). $\mathcal{C}$ can check the validity of the forged signature σ^* as follows.

Correctness of public keys. It is clear that $\langle P^*_{i1},\ P^*_{i2}\rangle$ satisfies the verification
$e(P^*_{i1}, P) \stackrel{?}{=} e(P^*_{i2}, H_0(ID_i))$.

Correctness of forged signature. Note that $\mathcal{C}$ will use the current public key of ID^* that was set by $\mathcal{A}_I$.

- $\mathcal{C}$ has to check whether $e(V^*, P_{j1}) \stackrel{?}{=} e(U^*, D_{j1})\ e(h^* P^*_{i1}, D_{j1})$. In fact

$$
\begin{aligned}
R.H.S &= e(U^*, D_{j1})\ e(h^* P^*_{i1}, D_{j1}) \\
&= e(U + hP_{i1} - H_0(ID^*), D_{j1})\ e(h^* P^*_{i1}, D_{j1}) \\
&= e(U + hP_{i1} - H_0(ID^*), D_{j1})\ e(H_0(ID^*), D_{j1}) \\
&= e(U, D_{j1})\ e(hP_{i1}, D_{j1}). \\
&= e(V, P_{j1}). \\
&= e(V^*, P_{j1}) \\
&= L.H.S
\end{aligned}
$$

Thus the forged signature σ^* passes the verification successfully.

Type-II Attack. Type-II attack is also possible on the same scheme. During the unforgeability game between the challenger $\mathcal{C}$ and adversary $\mathcal{A}_{II}$, $\mathcal{A}_{II}$ can interacts with $\mathcal{C}$ and access the **Sign** oracle with the restrictions given in the model. $\mathcal{A}_{II}$ can ask signature on any message and identity pair $\langle m, ID\rangle$. $\mathcal{A}_{II}$ has access to the master private key. So it can compute the private key of any user from its public keys $\langle P_{i1} P_{i2}\rangle$ as $D_i = kP_{i1}$. Since the public key P_{i1}=$x_i H_0(ID_i)$, so $\mathcal{A}_{II}$ can generate signature on behalf of any user and $\mathcal{A}_{II}$ can verify the signature of any user. Here, we can visualize $\mathcal{A}_{II}$ as the KGC because it knows the master private key in the scheme.

4 Review and Cryptanalysis of Broadcast Group Oriented Signatures [7].

In this section we review BGOS scheme and present the cryptanalysis of the same.

4.1 Review of Broadcast Group Oriented Signatures [7].

Let $\mathbb{A}$= $\{a_1, a_2, ..., a_n\}$ and $\mathbb{B} = \{b_1, b_2, ..., b_n\}$ be two groups. The set $\{ID_{a1}, ID_{a2}...ID_{an}\}$ is the corresponding identities of users $\{a_1, a_2, ..., a_n\}$ and the set $\{ID_{b1}, ID_{b2}...ID_{bn}\}$ is the corresponding identities of users $\{b_1, b_2, ..., b_n\}$. Let ID_A and ID_B be the identities of group $\mathbb{A}$ and group $\mathbb{B}$.

- **Initialize:**
  Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of same prime order q. Let P be a generator of additive group $\mathbb{G}_1$ and $\mathbb{G}_2$ is a multiplicative group. Let H_0 and H_1 be cryptographic hash functions defined as H_0 :$\{0,1\}^* \rightarrow \mathbb{G}_1$ and H_1:$\{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}^*_q$.

– **Key Generation:**
The KGC chooses $t, s \in_R \mathbb{Z}_q^*$ and computes public key pair (P_A, Q_A) for group $\mathbb{A}$, where $P_A = \{P_A^{(1)}, P_A^{(2)}\}$ and $Q_A = \{Q_A^{(1)}, Q_A^{(2)}\}$ are defined as follows.
  - $\{P_A^{(1)}, P_A^{(2)}\} = \{tsP, tP\}$.
  - $\{Q_A^{(1)}, Q_A^{(2)}\} = \{tsH_0(ID_A), tH_0(ID_A)\}$.
  - Computes partial private keys for each member of group $\mathbb{A}$ as $sH_0(ID_{ai})$ and $tH_0(ID_{ai})$ and then sends them to the corresponding member in a secure way.

  Each user $a_i \in \mathbb{A}$ chooses $x_i \in_R \mathbb{Z}_q^*$ and computes private key $D_{ai} = \{D_{ai}^{(1)}, D_{ai}^{(2)}\}$ as $\{x_i sH_0(ID_{ai}), x_i tH_0(ID_{ai})\}$ and sets the public key as $P_{ai} = x_i H_0(ID_{ai})$ and $Q_{ai} = x_i P$.

  Similarly, KGC chooses $s^{'} \in_R \mathbb{Z}_q^*$ uniformly at random and uses the same t used during the key generation for group $\mathbb{A}$, generates public key pair (P_B, Q_B) for group $\mathbb{B}$, where $P_B = \{P_B^{(1)}, P_B^{(2)}\}$ and $Q_B = \{Q_B^{(1)}, Q_B^{(2)}\}$ are defined as follows.
  - $\{P_B^{(1)}, P_B^{(2)}\} = \{ts^{'}P, tP\}$.
  - $\{Q_B^{(1)}, Q_B^{(2)}\} = \{ts^{'}H_0(ID_B), tH_0(ID_B)\}$.
  - Computes partial private keys as $s^{'}H_0(ID_{bi})$ and $tH_0(ID_{bi})$ and sends them to each member b_i of group $\mathbb{B}$ in secure way.

  Each user $b_i \in B$ chooses $y_i \in_R \mathbb{Z}_q^*$ and computes private key $D_{bi} = \{D_{bi}^{(1)}, D_{bi}^{(2)}\}$ as $\{y_i s^{'}H_0(ID_{bi}), y_i tH_0(ID_{bi})\}$ and sets the public key as $P_{bi} = y_i H_0(ID_{bi})$ and $Q_{bi} = y_i P$.

– **Public Key Verification:**
Each members of group $\mathbb{A}$ can verify the validity of group $\mathbb{A}$'s public keys as follows:
  - $e(P_A^{(1)}, H_0(ID_A)) = e(P, Q_A^{(1)})$ and
  - $e(P_A^{(2)}, H_0(ID_A)) = e(P, Q_A^{(2)})$.

  Similarly members of group $\mathbb{B}$ can verify the validity of group $\mathbb{B}$'s public keys as:
  - $e(P_B^{(1)}, H_0(ID_B)) = e(P, Q_B^{(1)})$ and
  - $e(P_B^{(2)}, H_0(ID_B)) = e(P, Q_B^{(2)})$.

  Public keys of member of group $\mathbb{A}$ can be verified by anyone as:
  $e(P_{ai}, P) = e(Q_{ai}, H_0(ID_{ai}))$.
  Public keys of member of group $\mathbb{B}$ can be verified by anyone as:
  $e(P_{bi}, P) = e(Q_{bi}, H_0(ID_{bi}))$. With these steps, the public keys can be verified even with out certificate.

– **Signature Generation:**
Assume that user b_i of group $\mathbb{B}$ wants to generates a signature of message m so that only members of group $\mathbb{A}$ can verify and no one outside the group $\mathbb{A}$ should not be able to verify the signature. User b_i generate the signature as follows:
  - Chooses $k \in_R \mathbb{Z}_q^*$ and computes $U_1 = kP_{bi}$ and $U_2 = kP_A^{(2)}$.
  - Computes $h = H_1(m, U_1)$ and generates $V = (h + k)(D_{bi}^{(2)} + P_A^{(1)})$.
  - b_i broadcasts the signature $\sigma = \langle m, U_1, U_2, V\rangle$ to group $\mathbb{A}$.

– **Signature Verification:**
User a_i of group $\mathbb{A}$ can verify the signature σ on m by performing the following check.
  - Computes $h^{'} = H_1(m, U_1)$.
  - Verifies $e(V, P_{ai}) \stackrel{?}{=} e(h^{'}P_{bi} + U_1, D_{aj}^{(2)})e(h^{'}P_A^{(2)} + U_2, D_{aj}^{(1)})$. If the check holds then accept the signature else reject.

4.2 Cryptanalysis of Broadcast Group Oriented Signature.

In BGOS, an user from one group can designate its signature to be verifiable by members of other group. In this section we present the cryptanalysis of BGOS scheme, which too has both Type-I and Type-II attacks.

Type-I Attack on BGOS Scheme [7] On seeing a valid signature by an user on some message, anyone can commit a forgery on any message. During the unforgeability game between the challenger $\mathcal{C}$ and adversary $\mathcal{A}_I$, $\mathcal{C}$ gives $\mathcal{A}_I$ the public parameters *params* and $\mathcal{A}_I$ gives to $\mathcal{C}$ a target identity ID^*_{bi}. $\mathcal{A}_I$ is supposed to generate a valid forgery for the target identity ID^*_{bi} on some message and it is not allowed to query partial private key for target identity ID^*_{bi}. $\mathcal{A}_I$ interacts with $\mathcal{C}$ and access all the oracles with the restrictions given in the model. $\mathcal{A}_I$ can query signature on any message and user identity pair $\langle m, ID\rangle$. $\mathcal{A}_I$ can replace the public keys of suppose any user including user with identity ID^*_{bi}. During the training-phase $\mathcal{A}_I$ receive a valid signature $\sigma = \langle m, U_1, U_2, V\rangle$ on a message m with target identity ID^*_{bi} using the **Sign** oracle. Now we show how $\mathcal{A}_I$ can generate a valid signature σ^* on an arbitrary message m^* for the target identity ID^*_{bi}, such that σ^* is not the output of previous queries to **Sign** oracle. This can be shown by the following computation done by $\mathcal{A}_I$

- Computes $U^*_1 = U_1 + hP_{bi}$-$H_0(ID^*_{bi})$ and $U^*_2 = U_2 + hP^{(2)}_A$ - P.
- Computes $h^* = H_1(m^*, U^*_1)$.
- Replaces ID^*_{bi}'s public keys as $P^*_{bi} = \frac{1}{h^*}H_0(ID^*_{bi})$ and $Q^*_{bi} = \frac{1}{h^*}P$.
- Replaces group $\mathcal{A}$'s public keys as $P^{(2)*}_A = \frac{1}{h^*}P$ and $Q^{(2)*}_A = \frac{1}{h^*}H_0(ID_A)$.
- $V^* = V$.

Now we claim that $\sigma^* = \langle m^*, U^*_1, U^*_2, V^*\rangle$ is a valid signature on the message m^* by the user with identity ID^*. $\mathcal{C}$ can check the validity of the forged signature σ^* as follows.

Correctness of Public Keys: The replaced public keys of group $\mathbb{A}$ $\langle P^{(2)*}_A, Q^{(2)*}_A\rangle$ passes the verification

$$e(P^{(2)^*}_A, H_0(ID_A)) \stackrel{?}{=} e(P, Q^{(2)^*}_A)$$

The replaced public keys of user b_i $\langle P^*_{bi}, Q^*_{bi}\rangle$ also passes the following verification:

$$e(P^*_{bi}, P) \stackrel{?}{=} e(Q^*_{bi}, H_0(ID_{bi}))$$

Correctness of forged signature: Note that $\mathcal{C}$ will use the current public key of ID^* that was set by $\mathcal{A}_I$. $\mathbb{C}$ has to check $e(V^*, P_{ai}) \stackrel{?}{=} e(h^*P^*_{bi} + U^*_1, D^{(2)}_{aj})e(h^*P^{(2)*}_A + U^*_2, D^{(1)}_{aj})$. Now,

$$\begin{aligned}
R.H.S =& e(h^*P^*_{bi} + U^*_1, D^{(2)}_{aj})e(h^*P^{(2)*}_A + U^*_2, D^{(1)}_{aj})\\
=& e(h^*P^*_{bi} + U_1 + hP_{bi} - H_0(ID^*_{bi}), D^{(2)}_{aj})e(h^*P^{2*}_A + U_2 + hP^{(2)}_A - P, D^{(1)}_{aj})\\
=& e(hP_{bi} + U_1, D^{(2)}_{aj})e(hP^{(2)}_A + U_2, D^{(1)}_{aj})\\
=& e(V, P_{ai})\\
=& e(V^*, P_{ai})\\
=& L.H.S
\end{aligned}$$

Thus the forged signature σ^* passes the verification successfully.

Type-II Attack on BGOS Scheme [7]. Type-II attack is also possible on BGOS [7] scheme. During the Unforgeability game between the challenger $\mathcal{C}$ and adversary $\mathcal{A}_{II}$, $\mathcal{A}_{II}$ can interact with $\mathcal{C}$ and can access **Sign** oracle with the restrictions given in the model. $\mathcal{A}_{II}$ can ask signature on any message and identity pair $\langle m, ID\rangle$. The adversary $\mathcal{A}_{II}$ can access the master private key. So, $\mathcal{A}_{II}$ can compute the full private key of any user from group $\mathbb{A}$ using the public keys $\langle P_{ai}, Q_{ai}\rangle$ as $\langle\{D^{(1)}_{ai}, D^{(2)}_{ai}\}\rangle = \langle sP_{ai}, tP_{ai}\rangle$ and any user from group B with public keys $\langle P_{bi}, Q_{bi}\rangle$ as $\langle\{D^{(1)}_{bi}, D^{(2)}_{bi}\}\rangle = \langle sP_{bi}, tP_{bi}\rangle$. As a result the KGC can generate signature on behalf of any user and also verify the signature of any user in any group, which contradicts the statement of the authors.

5 Review of another Group Inside Signature (GIS) [6]

5.1 Review of GIS [6]

Let $\mathbb{A}$= $\{a_1, a_2, ..., a_n\}$ be a group of n users. The set $\{ID_1, ID_2...ID_n\}$ is the corresponding identities of users $\{a_1, a_2, ..., a_n\}$. Let $a_i \in \mathbb{A}$ is the signer who wants to sign a message m, verifiable only by members of group $\mathbb{A}$.

Initialization: This algorithm is run by KGC. It takes security parameter 1^k as input and outputs public parameters *params* as output. The public parameters include the following.
- The public key pair $P_{pub} = \langle P_{pub1}, P_{pub2}\rangle$ is defined as $\langle g^{k^2}, g^{k^3}\rangle$.
- Two cryptographic hash functions namely H_0 and H_1 defined as $H_0 :\{0,1\}^* \to \mathbb{G}_1$ and $H_1:\{0,1\}^* \to \mathbb{G}_1$.

Key Extract: This a two step process. First KGC generates partial private key *ppk* for each user and transmits it in a secure way. Second each user after getting partial private key, computes his private key and public keys. The above process can be defined by the following steps.
- The KGC computes *ppk* for any user A as $D_A = g^{(k+H_1(ID_A))^{-1}k^{-1}}$.
- Any user A computes his private key after receiving *ppk* by selecting $x_A \in_R \mathbb{Z}_q^*$ S_A as $S_A = D_A^{x_A}$.
- Any user A computes and publishes his public keys $\langle X_A, Y_A\rangle$ as $\langle P_{pub1}^{x_A^{-1}}, P_{pub2}^{x_A^{-1}}\rangle$

Sign: To sign a message m, user A performs following steps.
- Chooses $a \in_R \mathbb{Z}_q^*$, computes $r = e(S_A, X_A^{H_1(ID_A)}.Y_A)^a$ and then computes $V = H_0(m||r)$.

So $\sigma = (m, U, V, ID_A)$ is the signature generated by user A on message m.

Verification: Without loss of generality we assume that user B who is the part of the group can performs the following steps for verifying a signature $\sigma = (m, U, V, ID_A)$ signed by user A.
- Computes $r^{'} = e(U, X_A^{H_1(ID_A)}.Y_A)e(S_B, X_B^{H_1(ID_B)}.Y_B)^{-V}$
- Checks $V \stackrel{?}{=} H_0(m||r^{'})$ if it holds output valid else output invalid.

5.2 Cryptanalysis of GIS [6].

Chunbo Ma et al. have proposed another GIS [6] scheme. In this section, we present *Type-I* forgery on the scheme [6]. Here adversary $\mathcal{A}_I$ who considered to be inside the group can sign on behalf of any user on any message. During the unforgeability game between the challenger $\mathcal{C}$ and adversary $\mathcal{A}_I$, $\mathcal{C}$ gives $\mathcal{A}_I$ the public parameters *params* and a target identity ID_A. $\mathcal{A}_I$ is supposed to generate a valid forgery for the target identity ID_A on some message and it is not allowed to query partial private key for the target identity ID_A. $\mathcal{A}_I$ interacts with $\mathcal{C}$ and access all the oracles with the restrictions given in the model. $\mathcal{A}_I$ can query signature on any message and user identity pair $\langle m, ID\rangle$. $\mathcal{A}_I$ can replace the public keys of any user including user with identity ID_A. During the training-phase $\mathcal{A}_I$ receives a valid signature $\sigma = \langle m, U, V\rangle$ on a message m with target identity ID_A as the signer from the **Sign** oracle and also obtains the private key of some other user say ID_B from the **Key Extract** oracle. Now $\mathcal{A}_I$ can generate a valid signature σ^* on a message m^* for the target identity ID_A by using the private key of ID_B, such that σ^* is not the output of previous queries to **Sign** oracle. This can be shown by the following computation done by $\mathcal{A}_I$.First $\mathcal{A}_I$ computes the value $e(g, g^k)$ even though $\mathcal{A}_I$ may not know the value $e(g, g^k)$ directly, it can compute $e(g, g^k)$ as follows.

$$\begin{aligned} e(D_B, P_{pub2})e(D_B, (P_{pub1})^{H_1(ID_B)}) &= e(g^{\frac{k^2}{k+H_1(ID_B)}}, g)e(g^{\frac{kH_1(ID_B)}{k+H_1(ID_B)}}, g) \\ &= e(g^{\frac{k^2}{k+H_1(ID_B)}} g^{\frac{kH_1(ID_B)}{k+H_1(ID_B)}}, g) \\ &= e(g, g^k) \end{aligned}$$

Hence, $e(g, g^k)$ can be computed by $\mathcal{A}_I$ and subsequently $\mathcal{A}_I$ generates the forgery by performing the following:
- Computes $r^* = e(g, g^k)^{a^*}$.
- Computes $V^* = H_0(m^*||r^*)$.
- Computes $U^* = SK_B^{(a^*+v^*)}$.
- Replaces ID_A's public keys $X_A^* = X_A$ and $Y_A^* = X_A^{(-H_1(ID_A))}\ X_B^{H_1(ID_B)}\ Y_B$.
- Broadcasts the signature σ^* (m^*, U^*, V^*, ID_A).

Now challenger $\mathcal{C}$ can verify the validity of the signature using the private key of any group member say C as follows:

Computes $r^{'}$ as

$$\begin{aligned} &e(U^*, (X_A^*)^{H_1(ID_A)}.Y_A^*)e(S_C, X_C^{H_1(ID_C)}Y_C)^{-V^*} = \\ &\quad = e(U^*, X_A^{H_1(ID_A)}.X_A^{-H_1(ID_A)}X_B^{H_1(ID_B)}Y_B)\ e(S_C, X_C^{H_1(ID_C)}.Y_C)^{-V^*} \\ &\quad = e(g,g)^{k(a^*+V^*)}e(g,g)^{-V^*k}. \\ &\quad = e(g,g)^{ka^*} \\ &\quad = r' \end{aligned}$$

Checks $V^* \stackrel{?}{=} H_0(m^*||r^{'})$ if it holds σ^* is a valid forgery other wise not.

Since σ^* is a valid forgery which we showed now, we can claim that the scheme given in [6] is having *Type-I* forgery.

6 Adaptable Designated Group Signature(ADGS)

This section explains generic model of ADGS and the security notion for GDS namely *unforgeability* and *unlinkability*. *Unforgeability* notion is same as that of any digital signature scheme but this paper is the first in literature to consider the notion of *unlinkability*, which is a very essential property of ADGS. The definition of *unlinkability* will be given shortly.

6.1 Generic model

An Adaptable designated group signature scheme consists of the following four algorithms: **Initialize**,**Key Generation**,**Sign** and **Verify**. The algorithms are described as follows.

- **Initialize:** This algorithm takes security parameter 1^k as input. The **PKG** runs this algorithm to produce public parameters *params* available globally and Msk, the master private key. The public parameters include master public key P_{pub} , cryptographic hash functions and the group definitions used in the scheme.
- **Key Generation/Extract:** This algorithm takes master private key s and identity of user ID_A as input and compute the private key D_A corresponding to ID_A. This is run by the PKG.
- **Sign:** The inputs to this algorithm are an identity ID_A and corresponding private key D_A, message m, and a list of designated group members as verifiers. **Sign** algorithm outputs σ, the signature on the message m which can be verified only by the group members.
- **Verify:**This oracle takes $(m, ID_S, ID_R, D_R, \sigma)$, where ID_S is signers identity and ID_R is the designated verifier, as input and output true if σ is a valid signature on message m with ID_R as designated verifier and ID_S as the signer. Members of the designated group alone can verify the signature.

6.2 Security notion for ADGS Scheme

Unforgeability. The most general notion of security for identity based signature scheme is security against existential forgery under adaptive chosen message and identity attack. We can say an identity based signature is secure against $existential\ forgery\ under\ adaptively\ chosen\ message\ and$ ID $attacks$ if no polynomial time adversary $\mathcal{A}$ has non-negligible advantage against a challenger $\mathcal{C}$ in the following game.

$\mathcal{C}$ runs Setup of the scheme and gives simulated *params* to $\mathcal{A}$. First, $\mathcal{A}$ will fix the target identity ID^* and gives it to $\mathcal{C}$.

Training Phase. $\mathcal{A}$ can access the following oracles with the restriction that $\mathcal{A}$ cannot ask the private key of ID^*.

- **Hash Oracles.** If the hash oracle is queried for the same input again it retrieves and returns the value from the list to $\mathcal{A}$ else $\mathcal{C}$ chooses a random value from the output range of the hash function for the given input, saves it in the list and returns the value to $\mathcal{A}$.
- **Extract Oracle.** Given an identity $ID \neq ID^*$, $\mathcal{C}$ returns the private key corresponding to that ID.
- **Sign Oracle.** Given an ID and a message m, $\mathcal{C}$ returns the signature σ corresponding to (ID, m).
- **Verify Oracle.**This oracle takes $(ID_S, ID_R, D_R, \sigma)$ where ID_S is signers identity and ID_R is the designated receiver as input and output true if σ is a valid signature with ID_R as designated verifier and ID_S as the signer. Members of the designated group alone can verify the signature.

Forgery. $\mathcal{A}$ outputs (ID^*, m^*, σ^*), where ID^* is the target identity, m^* is a message and σ^* is a signature such that σ^* was not the output of previous **Sign** query with $\langle m^*, ID^* \rangle$ as input. $\mathcal{A}$ wins the game if σ^* is a valid signature of m^* by ID^*.

Unlinkability. For showing a ADGS scheme is not verifiable outside the designated group, we first propose the model for *unlinkability*. This can be viewed as a game between challenger $\mathcal{C}$ and adversary $\mathcal{A}_l$.

Initialize The $\mathcal{C}$ runs the **Setup** algorithm and generates the system parameters *params* and the master private key Msk. It then delivers *params* to $\mathcal{A}_l$ and keeps msk secret. $\mathcal{A}_l$ outputs a list of identities $\mathcal{L}^*$ to $\mathcal{C}$, for which $\mathcal{A}$ is not allowed to ask **Extract** query.

Phase-1

- The adversary $\mathcal{A}_l$ can access the oracles as follows:
  - **Extract Oracle.** Given an $ID \notin \mathcal{L}^*$, $\mathcal{C}$ returns the private key corresponding to that ID. All the other oracles are the same as that of the unforgeability game.

Challenge: $\mathcal{A}_l$ produces two message of equal length and an arbitrary sender identity ID_A. The challenger $\mathcal{C}$ flips a coin, sampling a bit b $\in_R \{0,1\}$ and computes $\sigma^* =$ **Sign** $(m_b, ID_A, \{ID_1, ID_2, ..., ID_t\}, D_A)$. Here $\{ID_1, ID_2, ..., ID_t\} \in \mathcal{L}^*$. σ^* is returned to $\mathcal{A}_l$ as challenge signature.

Phase-2: $\mathcal{A}$ is allowed to make polynomially bounded number of new queries to the oracles with the restrictions that adversary should not query the *Extract* oracle for the private keys of users in $\mathcal{L}^*$ and $\mathcal{A}_l$ should ask verication of σ^* to $verify$ oracle.

Guess: At the end of this game, $\mathcal{A}$ outputs a bit $b^{'}$. $\mathcal{A}$ wins the game if $b^{'}=$ b.

7 Review and Cryptanalysis of ADGS Scheme [8]

7.1 Review of ADGS Scheme [8]

Consider a collection of $n+1$ users $\{a_0, ..., a_n\}$ with identities $\{ID_0, ID_1, ..., ID_n\}$ where the users may be from different groups. Let a_0 be an user who wants to sign a message to the remaining people $\mathbb{U} = \{a_1, ..., a_n\}$ such that only members $a_i \in \mathbb{U}$ for $1 \leq i \leq n$ should be able to verify its signature. The group $\mathbb{U}$ is defined by user a_0.

- **Initialize:**
  Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of same prime order q. Let P be a generator of additive group $\mathbb{G}_1$ and $\mathbb{G}_2$ be a multiplicative group. Let H_0 and H_1 be cryptographic hash functions defined as $H_0 :\{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_1:\{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$.
- **Key Generation:**
  Let ID_i denote the identity of user a_i. The PKG selects a random number s $\in_R \mathbb{Z}_q^*$, sets a_i's private key is $D_i = sH_0(ID_i)$ and a_i's public key is $Q_i = H_0(ID_i)$. It then communicates the private key of the users in a secure way and publishes the public parameters $params = \langle \mathbb{G}_1, \mathbb{G}_2, q, H_0, H_1, P, P_{pub} = sP \rangle$.
- **Designated Group Signature Generation(Sign):**
  To sign a message m the signer a_0 selects r,k,t $\in_R \mathbb{Z}_q^*$ and computes the value $T_i = kQ_i$, where $a_i \in_R \mathbb{U}$ $1 \leq i \leq n$. Then a_0 computes the following values
  - $V_0 = tsP$.
  - $V_1 = tkP$.
  - $V_2 = rkD_0$.
  - $T_0 = kQ_0$.
  - $h = H_1(m)$
  - $V = (r+h)D_0$.

  The signer a_0 produces the signature on message m as $\sigma = (m, V, V_0, V_1, V_2, T_0, ..., T_n)$.
- **Signature Verification:**
  - *Judge Verifier*. The aim of this step is to judge who can verify the signature. Using the value T_i, any member of the set $\mathbb{U}$ can verify whether he is eligible to perform verification by checking whether $e(T_i, V_0) \stackrel{?}{=} e(D_i, V_1)$. If the equation holds, then the corresponding member has the ability to verify the signature.
  - *Verify Signature*. The member $a_i \in \mathbb{U}$, who passes the above verification, can perform the signature verification as follows. Check whether $e(V, T_i) \stackrel{?}{=} e(V_2, Q_i)e(hT_0, D_i)$. If it holds then the signature is valid.

7.2 Cryptanalysis of ADGS [8] Scheme.

In this section, we present the cryptanalysis of ADGS [8]. We show that the ADGS scheme in [8], is universally forgeable by demonstrating two different ways to proceed with the attack.

Universal Forgery without having access to any previous signature. The scheme ADGS described above is universally forgeable. The adversary $\mathcal{A}$ can forge the signature of any user without seeing any valid signature previously signed by any user. $\mathcal{A}$ selects r^*, k^*, $t^* \in_R \mathbb{Z}_q^*$, computes $T_i^* = k^*Q_i \ for(i = 1 \ to$ n $a_i \in \mathbb{U})$. and then computes the following values.

- $V_0^* = t^*s^*P$.
- $V_1^* = t^*k^*P$.
- $V_2^* = r^*k^*P$.
- $h^* = H_1(m^*)$.
- $T_0^* = \frac{1}{h^*}k^*P$.

– $V^* = r^*P + P_{pub}$.
  $\mathcal{A}$ produces $\sigma^* = (m, V^*, V_0^*, V_1^*, V_2^*, T_0^*, ..., T_n^*)$ as a valid signature on message m^*.

Now the correctness of the forged signature σ^* can be shown as follows:
Correctness: The $L.H.S$ is

$$\begin{aligned} e(V^*, T_i^*) &= e(r^*P + P_{pub}, k^*Q_i) \\ &= e(r^*P, k^*Q_i)e(P_{pub}, k^*Q_i) \\ &= e(r^*k^*P, Q_i)e(k^*P, sQ_i) \\ &= e(V_2^*, Q_i)e(\tfrac{1}{h^*}T_0^*, D_i) \\ &= R.H.S \end{aligned}$$

Thus, we show that $\mathcal{A}$ is capable of generating a valid ADGS on behalf of user with out knowing users secret key.

Universal Forgery on seeing a signature of an user On seeing a valid signature by an user on some message, anyone can commit a forgery on any message. During the unforgeability game between the challenger $\mathcal{C}$ and adversary $\mathcal{A}$, $\mathcal{C}$ gives $\mathcal{A}$ the public parameters *params* and a target identity ID^*. $\mathcal{A}$ is supposed to generate a valid forgery for the target identity ID^* on some message and it is restricted to query private key for the target identity ID^*. $\mathcal{A}$ interacts with $\mathcal{C}$ and accesses all the oracles with the restrictions given in the model. $\mathcal{A}$ can query signature on any message and user identity pair $\langle m, ID\rangle$. $\mathcal{A}$ can replace the public keys of any user including user with identity ID^*. During the training-phase on receiving a valid signature $\sigma = \langle m, V, V_0, V_1, V_2, T_0, ..., T_n\rangle$ on a message m with target identity ID^* from the **Sign** oracle, $\mathcal{A}$ can generate a valid signature σ^* on a message m^* for the target identity ID^*, such that σ^* is not the output of previous queries to **Sign** oracle. This can be shown by the following computation done by $\mathcal{A}$

– Dividing V by h. $\frac{1}{h}V = (\frac{r}{h} + 1)D_0$ where $h = H_1(m)$.
– Computes $h^* = H_1(m^*)$.
– $V_0^* = V_0$ and $V_1^* = V_1$.
– $V_2^* = \frac{h^*}{h}V_2$.
– The remaining parameters $T_0, ..., T_n$, V_0 and V_1 are same as that of original signature.
– $V^* = h^*\frac{V}{h}$.

Now $\sigma^* = \sigma^*$ $(m^*, V^*, V_0, V_1, V_2^*, T_0, ..., T_n)$ is a valid signature on the message by the user with identity ID^*. $\mathcal{C}$ can check the validity of the forged signature σ^* as follows.

Correctness: The $L.H.S$ is

$$\begin{aligned} e(V^*, T_i) &= e((\tfrac{h^*}{h}r + h^*)D_0, k^*Q_i) \\ &= e(\tfrac{h^*}{h}rD_0, kQ_i)e(h^*D_0, kQ_i) \\ &= e(\tfrac{h^*}{h}rkD_0, Q_i)e(h^*kQ_0, D_i) \\ &= e(V_2^*, Q_i)e(\tfrac{1}{h^*}T_0^*, D_i) \\ &= R.H.S \end{aligned}$$

Now, it is clear that the forged signature σ^* passes the verification successfully.

8 New ADGS scheme(N-ADGS).

In this section we present a new identity based ADGS scheme. Assume that a signer a_0 has to designate his signature to be verified by n users namely $\{a_1, ..., a_n\}$. All the n users may be from different groups and are selected by a_0. The signer a_0 forms the set $\mathbb{U} = \{a_1, ..., a_n\}$ to generate the signature. In our scheme designated members of the group cannot simulate the signers signature.

– **N-ADGS Initialize:**
  The PKG initializes the system by executing this algorithm. This algorithm takes the security parameter 1^k as input and produces two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, where $|q| = k$, a generator P of $\mathbb{G}_1$, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and two cryptographic hash functions H_1 :$\{0,1\}^* \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$ and H_2 :$\{0,1\}^* \rightarrow \mathbb{G}_1$. The master private key is $s \in_R \mathbb{Z}_q^*$ and the master public key is set to be $P_{pub} = sP$. Sets $\theta = e(P_{pub}, R)$ where $R \in_R \mathbb{G}_1$. The public parameters are $\langle \mathbb{G}_1, \mathbb{G}_2, e, P, P_{pub}, P_{pub}, H_1, H_2, \theta, R\rangle$.

- **N-ADGS Key Generation/Extract:** This algorithm is executed by the PKG and on input of identity ID_i, PKG computes $Q_i = H_2(ID_i)$ and sets the private key as $D_i = sQ_i$. Now, D_i is sent to the user in a secure way.
- **N-ADGS Sign:** To sign a message m for a designated group of users $\mathbb{U} = (a_1, ..., a_n)$ with identities $(ID_1, ..., ID_n)$ the user with identity ID_0, private key D_0 and public key Q_0 performs the following steps:
  - Chooses r,k,t $\in_R Z_q^*$ and computes $T_i = \langle T_{i1}, T_{i2}\rangle$ as $\langle t(Q_i + R), kQ_i\rangle$ for$(i = 1$ to $n)$.
  - Computes $U_1 = rQ_0$, $U_2 = rkP$ and $U_3 = tP$.
  - Computes $\omega = e(D_0, U_3)$ and Computes $W = \theta^t\omega$.
  - Computes $h = H_1(m, \omega, U_1, U_2, U_3)$ and $V = rP_{pub} + hD_0$.

  Now $\sigma = (m, V, W, U_1, U_2, U_3, T_1, ..., T_n, \mathbb{U})$ is a valid signature on message m by ID_0, with the user group $\mathbb{U}$ as designated verifiers.
- **N-ADGS Verify:** Verification is a two step process. First step is to verify whether the verifier belongs to the group $\mathbb{U}$ and second step is to verify the validity of the signature.
  - *Judge Verifier:* Using the value $T_{i2} = kQ_i$, the verifier checks whether $e(T_{i2}, Q_0) \stackrel{?}{=} e(Q_i, U_1)$. If the verification holds then user with public key Q_i will do the next step in verification.
  - *Verify Signature:* Each designated verifier $a_i \in \mathbb{U}$ can verify the signature by performing the following steps.
    * Computes $\omega^{'} = We(D_i, U_3)e(P_{pub}, T_{i1})^{-1}$.
    * Computes $h^{'} = H_1(m, \omega^{'}, U_1, U_2, U_3)$.
    * Checks whether $e(V, T_{i2}) \stackrel{?}{=} e(h^{'}U_1, D_i)e(U_2, D_i)$.

  If the above check hold then the signature is valid. Otherwise the signature is invalid.

8.1 Security proof for N-ADGS

In this section we formally prove the security of N-ADGS. The two notions of security for N-ADGS are $unforgeability$ and *unlinkability*. $Unforgeability$ ensures that N-ADGS is not be forgeable. $Unlinkability$ ensures that members other than the designated group cannot be able to verify the signature. We prove the security of our scheme in the random oracle model.

Unforgeability Proof:

Theorem 1. *Our N-ADGS scheme is existentially unforgeable under chosen message and identity attack if **CDHP** is hard in $\mathbb{G}_1$.*

Proof. The challenger $\mathcal{C}$ receives an instance of CDHP $\langle P, aP, bP\rangle$ and it aims to compute abP. $\mathcal{C}$ uses $\mathcal{A}$ who is capable of breaking the existential unforgeability of N-ADGS to solve this instance. Now, $\mathcal{C}$ sets $P_{pub} = aP$, chooses $R \in_R \mathbb{G}_1$ and sets $\theta = e(P_{pub}, R)$ and sends *params* to $\mathcal{A}$. $\mathcal{A}$ on receiving system parameters, chooses the target identity ID^* and gives it to $\mathcal{C}$. $\mathcal{A}$ can access the following oracles which are controlled by $\mathcal{C}$.

Training Phase: Assume that for any identity ID, $\mathcal{A}$ queries H_2 and **Extract** oracle at most once and $\mathcal{A}$ queries H_2 oracle before it queries **Extract**, H_1 and **Sign** oracles.

- H_2 **Oracle:**
  $\mathcal{C}$ maintains a list L_2 to reply the H_2 oracle queries. It replies as follows
  - It searches list L_2 and if it finds a match in a tuple it will return the corresponding value.
  - Otherwise, it sets $H_2(ID_i)$ as
    * $H_2(ID_i) = bP$, if $ID_i = ID^*$ and adds tuple $\langle ID^*, bP, \perp, \perp\rangle$ to L_2.
    * $H_2(ID_i)$= x_iP and computes $D_i = x_iP_{pub}$, if $ID_i \neq ID^*$ and adds $\langle ID_i, x_iP, x_i, D_i\rangle$ to L_2.
- **Keygen/Extraction Oracle:** For any given identity $ID_i \neq ID^*$, $\mathcal{C}$ searches for private key D_i in list L_2, finds the corresponding x_i value and returns it to ID_i.
- H_1 **Oracle:**$(m, \omega, U_1, U_2, U_3)$. Let L_1 be the list associated with this oracle. If the tuple $\langle m, \omega, U_1, U_2, U_3\rangle$ is not queried already, $\mathcal{C}$ chooses randomly $h_i \in_R Z_q^*$ and adds $\langle m, \omega, U_1, U_2, U_3, h_i\rangle$ to the list L_1 and returns h_i to $\mathcal{A}$. Else if $\langle m, \omega, U_1, U_2, U_3\rangle$ is queried previously, returns h_i from $\langle m, \omega, U_1, U_2, U_3, h_i\rangle$ which is already stored in L_1.
- **Sign Oracle:** $\mathcal{A}$ queries the signature on message m for a signer identity $ID_0 \neq ID^*$, $\mathcal{C}$ responds according to the signing algorithm. If $ID = ID$, $\mathcal{C}$ responds as follows.
  - Chooses r, k, t and $z \in_R \mathbb{Z}_q$.
  - Computes $V = zP_{pub}$ and $U_1 = kbP$.
  - Chooses $h \in_R \mathbb{Z}_q^*$, computes $U_2 = zkP$ - hU_1, $U_3 = tP$, $T_{i1} = kQ_i$ and $T_{i2} = t(Q_i + R)$.

- Chooses $\omega \in_R \mathbb{G}_2$, computes $W = \theta^t \omega$ and adds $\langle m, \omega, U_1, U_2, U_3, h\rangle$ to the list L_2.
- Returns the signature σ= $(m, V, W, U_1, U_2, U_3, T_1, ..., T_n, \mathbb{U})$ to $\mathcal{A}$.

The correctness of σ can be shown as below. Let $a_i \in \mathbb{U}$ be a member of the designated group $\mathbb{U}$. a_i performs

- Compute $\omega^{'} = We(D_i, U_3)e(P_{pub}, T_{i1})^{-1}$ and $h^{'} = H_1(m^*, \omega^{'}, U_1^*, U_2^*, U_3^*)$.

$$\begin{aligned} e(V, T_{i2}) &= e(h^{'}U_1 + U_2, D_i) \\ &= e(h^{'}kbP + zP_{pub} - h^{'}kbP, kQ_i) \\ &= e(zP_{pub}, kQ_i) \\ &= e(V, T_{i2}) \end{aligned}$$

Thus we have shown that σ is a valid signature on m.

- **Verify Oracle:** This oracle takes (ID_S, ID_R, σ) where ID_S is signers identity and ID_R is the designated receiver as input. This oracle output true if σ is a valid signature on m with ID_R as designated verifier and ID_S as the signer. If the receivers identity $ID_R \neq ID^*$, the oracle proceeds as per the verifying algorithm. Else $ID_R = ID^*$, it performs computations as follows
  - Computes $\omega^{'} = e(D_S, U_3)$ and $h^{'} = H_1(m, \omega^{'}, U_1, U_2, U_3)$.
  - Computes $X = V$- $h^{'}D_S$.
  - Checks $e(X, Q_S) \stackrel{?}{=} e(U_1, P_{pub})$ if it holds it returns valid else it returns invalid.

Forgery : After the training phase is over, eventually, $\mathcal{A}$ outputs a forgery $\sigma^* = (ID^*, m^*)$. $\mathcal{C}$ checks the validity of the signature σ^* since it knows the private key of all designated verifiers in $\mathbb{U}$. If σ^* is a valid signature then by applying Forking Lemma [9], $\mathcal{C}$ replays the interaction with $\mathcal{A}$ with same random value but different hash function, and obtains two valid signatures as $\sigma^* = (ID^*, m^*, V, W, U_1, U_2\ , U_3, T_1, ..., T_n, \mathbb{U})$ and $\sigma^{*'}$= $(ID^*, m^*, V^{'}, W,\ U_1, U_2, U_3, T_1, ..., T_n, \mathbb{U})$. Let D^* be the private key of ID^*.
Since V=$rP_{pub} + hD^*$ and $V^{'}$=$rP_{pub} + h^{'}D^*$ are two valid signatures $\mathcal{C}$ gets V-$V^{'}$= $(h - h^{'})abP$ and $abP =$ $(h - h^{'})^{-1}$ $(V - V^{'})$. Which is the solution for the **CDHP** instance $\mathcal{C}$ has received.
Thus, if $\mathcal{A}$ can forge $N - ADGS$ scheme, with almost the same probability $\mathcal{C}$ can solve $CDHP$.

Unlinkability Proof.

Theorem 2. *Our N-ADGS scheme is unlinkable in the sense that members outside the group cannot verify the signature if **DBDHP** is hard in $\mathbb{G}_1$.*

Proof. The challenger $\mathcal{C}$ receives an instance (P, aP, bP, cP, α) of the DBDH problem. Its aim is to decide whether α $= e(P, P)^{abc}$ or not. Suppose there exist an adversary $\mathcal{A}_l$ who is capable of breaking the unlinkability of N-ADGS, we can show that $\mathcal{C}$ uses $\mathcal{A}_l$ to solve **DBDHP**. $\mathcal{C}$ sets $P_{pub} = cP,\ R = bP$, and $\omega = e(R, P_{pub})e(R, cP)$ and gives $\mathcal{A}_l$ the system parameters *params*. Now the adversary $\mathcal{A}_l$ outputs the list of identities $\mathcal{L}^* = \{ID_0^*, ID_1^*, ..., ID_t^*\}$ which is the set of target users, on which it is to be challenged.

Phase-1 In this phase $\mathcal{A}_l$ can access the various oracles which are maintained and manipulated by $\mathcal{C}$. The oracles respond as follows for the queries done by $\mathcal{A}$:

- H_2 **Oracle:**
  $\mathcal{C}$ maintains a list L_2 to reply the H_2 oracle queries. It replies as follows
  - It searches list L_2 and if it finds match it will return the corresponding value.
  - Otherwise, it sets $H_2(ID_i)$ as
    - $H_2(ID_i) = x_iP$-R if $ID_i \notin \mathcal{L}^*$ and computes $D_i = x_iP_{pub}$, adds tuple $\langle ID_i, x_iP - R, x_i, D_i\rangle$ to L_2.
    - $H_2(ID_i)$= x_iP if $ID_i \in \mathcal{L}^*$, adds tuple $\langle ID_i, x_iP, x_i, \perp\rangle$ to L_2.
- **Extraction Oracle:** For any given identity $ID_i \notin \mathcal{L}^*$, $\mathcal{C}$ searches the private key D_i in list L_2, finds the corresponding x_i value and returns it to $\mathcal{A}_l$.
- H_1 **Oracle:**$((m, \omega, U_1, U_2, U_3))$. Let L_1 be the list associated with this oracle. If the tuple $\langle m, \omega, U_1, U_2, U_3\rangle$ is not queried already, $\mathcal{C}$ chooses randomly $h_i \in_R Z_q^*$ and adds $\langle m, \omega, U_1, U_2, U_3, h_i\rangle$ to the list L_1 and returns h_i to $\mathcal{A}_l$. Else if $\langle m, \omega, U_1, U_2, U_3\rangle$ is queried previously, returns h_i from $\langle m, \omega, U_1, U_2, U_3, h_i\rangle$ which is already stored in L_1.
- **Sign Oracle:** $\mathcal{A}_l$ queries the signature on message m with signer identity $ID_0 \notin \mathcal{L}^*$ then it responds according to the signing algorithm. Otherwise, the oracle responds as follows.
  - Chooses randomly r,k,t and $z \in_R Z_q^*$.

- Chooses $h \in_R Z_q^*$.
- Computes $U_1 = kbP$ $U_2 = zkP$ - hU_1 and $U_3 = tP$.
- Computes $T_{i1} = kQ_i$ and $T_{i2} = t(Q_i + R)$.
- Chooses $\omega \in_R \mathbb{G}_2$, computes $W = \theta^t \omega$and computes $V = zP_{pub}$.
- Adds $\langle m, \omega, U_1, U_2, U_3, h_i \rangle$ to the list L_1.

It returns $\sigma =$ to $\mathcal{A}_l$ which is verifiable by $\mathcal{A}_l$.

– **Verify Oracle:** This oracle takes (ID_S, ID_R, σ) where ID_S is signers identity and ID_R is the designated receiver as input and output true if σ is a valid signature on m with ID_R as designated verifier and ID_S as the signer. If the receivers identity $ID_R \notin \mathcal{L}^*$, the oracle proceeds as per the verifying algorithm. Else $ID_R \in \mathcal{L}^*$, it performs computations as follows

- Computes $\omega' = e(D_S, U_3)$ and $h' = H_1(m, \omega', U_1, U_2, U_3)$.
- Computes $X = V$- $h' D_S$.
- Checks $e(X, Q_S) \stackrel{?}{=} e(U_1, P_{pub})$ if it holds it returns true else it returns false.

Challenge After the first query stage, $\mathcal{A}_l$ outputs two plain text messages m_0 and m_1 of equal length, together with a senders's identity ID_A on which he wishes to be challenged and a list of designated verifiers $\mathbb{U} \subseteq \mathcal{L}^*$. Now, C chooses a random bit $b \in_R 0, 1$ and signs message m_b as follows.

– Chooses randomly r,k,t and $z \in_R Z_q^*$.
– Chooses $h \in_R Z_q^*$.
– Computes $U_1 = kbP$ $U_2 = zkP$ - hU_1 and $U_3 = aP$.
– Computes $T_{i1} = kQ_i$ and $T_{i2} = x_i aP$, for $i = 1$ to t.
– Chooses $\omega \in_R \mathbb{G}_2$, compute $W = \alpha\omega$ and $V = zP_{pub}$.
– Adds $\langle m, \omega, U_1, U_2, U_3, h_i \rangle$ to the list L_1.
– Returns $\sigma = (m, V, W, U_1, U_2, U_3, T_1, ..., T_n)$ as a challenge to $\mathcal{A}_l$.

Phase-2 $\mathcal{A}_l$ can adaptively perform queries as in phase-1, with the restrictions that $\mathcal{A}_l$ should not query **Verify** oracle with(m, σ, ID_A, ID_i) where $ID_i \in \mathbb{U}$ and the **Extract** oracle for the private keys of identities $ID_1, ID_2, ..., ID_t$.

Guess:

At the end of this phase, $\mathcal{A}_l$ outputs a bit b'. $\mathcal{A}_l$ wins the game if b'= b. If $b = b'$ then σ^* is a valid signature on m_b from ID_A to the receivers in $\mathcal{L}^*$. It means $\mathcal{A}_l$ has successfully verified the signature and $\mathcal{A}_l$ has computed ω which is given as

$$\begin{aligned} We(D_i, U_3)e(P_{pub}, T_{i1})^{-1} &= \alpha\,\omega\,e(D_i, U_3)e(P_{pub}, T_{i1})^{-1}. \text{ (since } W = \alpha\omega) \\ &= \omega\ \alpha\ (e((x_i - b)cP, aP)e(cP, x_i aP)^{-1})^{-1}. \\ &= \omega\ \alpha\ (e(x_i aP, cP)e(-bcP, aP)e(cP, -ax_i P))^{-1}. \\ &= \omega\ \alpha\ (e(-abcP, P))^{-1}. \\ &= \omega \text{ iff } \alpha = \hat{e}(P, P)^{abc}. \end{aligned}$$

These calculations show that $\mathcal{A}_l$ get a correct ω if and only if $\alpha = e(P, P)^{abc}$.Thus if $\mathcal{A}_l$ can break our $N - ADGS$ scheme, with almost the same probability $\mathcal{C}$ can solve $DBDHP$.

Note: Our scheme can be extended to multi-receiver signcryption scheme.

9 Conclusion

In this paper we have presented attacks on certificateless schemes GIS [5], [6] BGOS [7] and an identity based ADGS [8]scheme. We have proposed a new identity-based ADGS scheme and we have deviced the model for linkability of identity based ADGS scheme. Also, we have formally proved the security of our scheme in random oracle model. We leave as an open problem to construct efficient identity based ADGS with constant size signature independent of the number of designated verifiers. Our scheme is secure against existential forgery on adaptively chosen message and ID attack under the hardness assumption of CDHP.

References

1. Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.
2. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

3. Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, and Xiaotie Deng. Key replacement attack against a generic construction of certificateless signature. In *ACISP*, volume 4058 of *Lecture Notes in Computer Science*, pages 235–246. Springer, 2006.
4. Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 1996.
5. Chunbo Ma, Faliang Ao, and Dake He. Certificateless group inside signature. pages 194–200, April 2005.
6. Chunbo Ma and Jun Ao. Certificateless group oriented signature secure against key replacement attack. Cryptology ePrint Archive, Report 2009/139, 2009. http://eprint.iacr.org/.
7. Chunbo Ma, Dake He, and Jun Ao. Broadcast group oriented signature. *Information, Communications and Signal Processing, 2005 Fifth International Conference on*, pages 454–458, 2005.
8. Chunbo Ma and Jianhua Li. Adaptable designated group signature. In *ICIC (1)*, volume 4113 of *Lecture Notes in Computer Science*, pages 1053–1061. Springer, 2006.
9. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
10. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 84*, Lecture Notes in Computer Science, pages 47–53. Springer-Verlag, 1985.