A Dynamic Attribute Based Group Signature Scheme with Attribute Anonymity

Abstract. Attribute Based Group Signature (ABGS) scheme is a group signature scheme where the group members possessing certain privileges(attributes) are only eligible for signing the document. There is no ABGS scheme which provide efficient way to preserve *Attribute Anonymity* with constant signature size. We have come up with an ABGS scheme which achieves the same and proven that it is secure under random oracle model with DL, q-SDH and XDH assumptions. We have also given the revocation mechanism to revoke multiple group members at anytime, which makes the scheme dynamic and practical.

Keywords: dynamic, attribute based, group signature, attribute anonymity, user revocation

1 Introduction

The group signature (GS) scheme, first introduced by Chaum and van Heyst in [10], is a signature scheme where any member belonging to the group can sign anonymously on any document on behalf of the group and later in case of any dispute the signer's identity can be revealed from the signature with the help of group's secret key. The applications for group signature scheme includes company authenticating pricelist, press releases, digital contracts [9], anonymous credit cards, access control [20], e-cash [17], e-voting, e-auction [27]. Since the publication of [10], many other schemes were proposed, including [11, 9, 1, 2, 4, 7], and more security requirements were added, including unlinkability, unforgeability, collision resistance [3], exculpability [3] and framing resistance [11]. In [4] Bellare et al. have given the formal definitions of the security properties of the group signature scheme by combining all the above security requirements into two, namely *Full-Anonymity* and *Full Traceablity.* In [5] Bellare et al. have extended the definitions for dynamic group settings, where the number of group members are not fixed or known in the setup phase, i.e. user can join or leave or revoke from the group at any time. The basic security requirements of group signature scheme in dynamic group settings are *Anonymity*, *Traceability* and *Non-frameability*, and are formally defined in [5]. *Anonymity* means that the signature should not reveal the identity of the signer. *Traceability* means that the valid signature should always trace back to the valid identity with the help of the group secret key. *Non-frameability* means that even if two or more members collude(including group manager), they should not be able to generate a signature which trace back to a non-colluded member.

Attribute Based Group Signature (ABGS) scheme is a group signature scheme where the group members possessing certain privileges, characterized by attributes, are only eligible for signing the document [19]. In ABGS, each member is assigned some attributes. The attribute relationships are represented by an *access tree*. The group members whose attributes satisfy the access tree are only eligible to sign the document. In otherwords, ABGS Scheme is a kind of group signature scheme where an user with a set of attributes can prove anonymously whether he possess these attributes or not [14]. The first ABGS was proposed by Dalia Khader [19], which uses the Goyal's Attribute Based Encryption(ABE) [16] and Boneh's GS [7]. Dalia Khader listed *Attribute Anonymity*, the verifier should be able to verify whether the signer has required attributes without learning which set of attributes he used for signing, as a desirable feature to achieve. Later Dalia Khader proposed the ABGS scheme [18] with member revocation feature without achieving *Attribute Anonymity*. Moreover, Dalia Khader schemes [19, 18] are not suitable when there is a frequent change in relationships among attributes since for each change the scheme has to reissue all the keys. Emura et al. proposed the dynamic ABGS scheme [14], which is efficient when there is a frequent change in attribute's relationships but it does not provide *Attribute Anonymity*. Moreover the signature size in both the schemes depend on the number of attributes. Yi Qian et al. [25] have given the ABGS scheme with *Attribute Anonymity*, but the scheme is not efficient to handle frequent changes in attribute relationships. If *Attribute Anonymity* is not preserved then it will also violate *Anonymity* property in some cases, thus the basic *Anonymity* property itself will not be met. Consider the case where there is a unique attribute which belongs to only one group member and whenever the verifier finds that attribute in the signature then he can conclude that the signature is signed by that particular group member who alone owns that attribute; thus *Anonymity* itself is not preserved which is the basic security requirement in any group signature scheme. Many similar cases exist. Thus *Attribute Anonymity* is as important as *Anonymity* property.

Recently, Maji et al. in [22] have proposed an Attribute-Based Signatures (ABS), where a signer can sign a message with a predicate that is satisfied by his attributes. Here, the signature reveals no information about the signer's identity but guarantees that the signer posses the required attributes. An ABGS scheme can behave exactly similar to an ABS scheme with the following modifications: adding attribute anonymity, removing the option of revealing signer's identity, removing the option of revealing attribute sets and making access tree publicly verifiable for its correctness. Moreover in ABS, singer trusts the attribute-issuing authority that it will not sign on behalf of him. But in ABGS, the group manager cannot do so. Thus, all the applications of ABS can also be handled by ABGS with attribute anonymity scheme.

Contribution: In this paper, we propose an ABGS scheme with *Attribute Anonymity* by using the Emura et al.'s ABGS scheme [14] as a base scheme. And we have proved that the proposed ABGS scheme is secure under random oracle model with DL, q-SDH and XDH assumptions. The proposed ABGS scheme achieves the better security bounds for *Anonymity* than in Emura et al.'s ABGS scheme and also achieves the constant signature size. We have also provided independent opening of the signer's identity and attribute sets identity from the signature, thus these tasks can be assigned to two independent authorities and it is also useful when anyone wants to know the privileges of the signer rather than its identity. We have also given the membership revocation mechanism for the proposed scheme, which allows group manager to revoke multiple members from the group at anytime, which makes scheme suitable for dynamic groups and more practical. We have also given an idea how to incorporate Verifier-Local Revocation (VLR) process into the proposed scheme. In this scheme, the group manager sends a revocation message only to the signature verifiers so that there is no need for the unrevoked signers to update their keys [8]. We have also shown how to publicly verify whether the public values of the access tree are valid or not, which allows the scheme to be suitable to all the applications of ABS [22]. For XDH assumption to hold, we assume that the instantiation of the bilinear groups are done using the Weil or Tate pairing over MNT curves, since in the supersingular curves the DDH problem is known to be easy on all cyclic subgroups [7, 15].

In Section 2, Definitions are listed and the proposed ABGS scheme with *Attribute Anonymity* is described in Section 3. Its security analysis is given in Section 4 followed by conclusion in Section 5.

2 Definitions

2.1 Bilinear Maps and Complexity Assumptions

Definition 1 (Bilinear Maps). *Bilinear map is defined as follows:*
Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_3$ be cyclic groups of prime order p. Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let ψ be an efficiently computable isomorphism, $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ with $\psi(g_2) = g_1$. The bilinear map e is an efficiently computable function, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ with the following properties.

– *Bilinearity : $\forall u, u' \in \mathbb{G}_1$ and $\forall v, v' \in \mathbb{G}_2$, $e(uu', v) = e(u, v)e(u', v)$ and $e(u, vv') = e(u, v)e(u, v')$.*
– *Non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_3}(1_{\mathbb{G}_3}$ is the $\mathbb{G}_3's$ identity element).*

The scheme is based on the *discrete logarithm (DL), q - Strong Diffie-Hellman (q-SDH)* [6] and *eXternal Diffie-Hellman (XDH)* [7] assumptions. $x \in_R S$ denotes the operation of picking an element x of S uniformly at random. ϕ denotes the null value.

Definition 2 (XDH assumption). *Bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ and an efficiently computable one-way isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ are given. Then the XDH assumption states that the DDH problem is hard in $\mathbb{G}_1$ even if DDH problem is easy in $\mathbb{G}_2$ [7].*

2.2 Access Tree

An access tree T consists of threshold gates as non-leaf nodes and attributes as leaves. Let l_x be the number of children of node x, and k_x $(0 < k_x \leq l_x)$ be the threshold value on the threshold gate of node x. A threshold gate represents that the number k_x of l_x children branching from the current node x need to be satisfied in order to imply that the parent node is satisfied. Satisfaction of a leaf is achieved by owning an attribute. The notation $Leaves \models T$ expresses the fact that a set of attributes $Leaves$ satisfies the access tree T. The verifier chooses an access tree that applies to his requirements and group manager will generate the public values of the access tree. The signer uses the needed attribute certificates, which he owns, to satisfy the access tree.

2.3 ABGS

Group signature scheme is a signature scheme where any member belonging to the group can sign anonymously on any document on behalf of the group and later in case of any dispute the signer's identity can be revealed from the signature with the help of group's secret key.

ABGS scheme is a group signature scheme where the group members possessing certain privileges, characterized by attributes, are only eligible for signing the document. In ABGS, each member is assigned some attributes. The attribute relationships are represented by an access tree. The group members whose attributes satisfy the access tree are only eligible to sign the document.

2.4 ABGS with Attribute Anonymity Model

Let GM be the group manager, k the security parameter, $params$ the system parameter, T be an access tree with a set of attributes $\{att_i\}_{i=1}^{r}$ for $r \geq 1$, where $att_i \in Att$ is assigned on each leaf, $\mathcal{T}$ the public values associated with T, gpk the group public key, ik the issuing key used for issuing a membership and attribute certificates, ok_{User} the user opening key used for opening the signer's identity from the group signature, ok_{Att} the attribute opening key used for opening the attribute set from the group signature, (upk_i, usk_i) the verification/signing key of a signature scheme $DSig$ for user U_i $(i = 1, 2, ..., n)$, A_i the membership certificate for U_i, $\Upsilon_i \subseteq Att$ the attributes of U_i, $\{T_{i,j}\}_{\mathrm{att}_j \in \Upsilon_i}$ the attribute certificates of U_i, sk_i the member secret key for U_i which includes both A_i and $\{T_{i,j}\}_{\mathrm{att}_j \in \Upsilon_i}$, and reg be the registration table with the group manager where the current group member's membership certificates are stored. In the Join algorithm the notation $\mathsf{Join}(\langle\text{input of GM}\rangle, \langle\text{input of user}\rangle)$ is used.

Each document M is associated with some attribute relationships whose access tree is denoted by T. The user U_i can make a group signature on the document M if there exists a set of attributes $\zeta \subseteq \Upsilon_i$ with the user such that $\zeta \models T$.

Definition 3 (ABGS with Attribute Anonymity). *Unless otherwise indicated, algorithms are randomized.*

- $\mathsf{Setup}(1^k)$*: This algorithm takes the security parameter k as an input and returns the system parameter params.*
- $\mathsf{KeyGen}(params)$*: This algorithm takes the system parameter, params and returns the group public key gpk, the issuing key ik, the user opening key ok_{User}, the attribute opening key ok_{Att} and the registration table $reg = \phi$.*
- $\mathsf{BuildTree}(params, gpk, ik, T)$*: This algorithm takes as input $params, gpk, ik$, and the access tree T whose leaves are associated with a subset of Att, and returns $\mathcal{T}$, the public values of the tree T.*
- $\mathsf{Join}(\langle params, gpk, ik, upk_i, \Upsilon_i\rangle, \langle params, gpk, upk_i, usk_i\rangle)$*: This is the interactive group joining protocol takes input $params, gpk$, the issuing key ik, upk_i and $\mathrm{U}_i{}'s$ attributes Υ_i from GM, and $params, gpk, upk_i$, and usk_i from U_i. In the protocol U_i ends with a member secret key sk_i and GM ends with an updated reg.*
- $\mathsf{Sign}(params, gpk, sk_i, \zeta, M, \mathcal{T})$*: This algorithm takes $params, gpk, sk_i$, an attributes set ζ, message $M, \mathcal{T}$ and returns a group signature σ on M, where $\zeta \subseteq \Upsilon_i$ and $\zeta \models T$.*
- $\mathsf{Verify}(params, gpk, M, \sigma, \mathcal{T})$*: This is a deterministic algorithm takes params, $gpk, M, \sigma, \mathcal{T}$ as an input and returns $1/0$. If 1 then the algorithm claims that the σ is a valid signature. Otherwise, σ is invalid.*
- $\mathsf{OpenUser}(params, gpk, ok_{User}, \sigma, M, \mathcal{T}, reg)$*: This is a deterministic algorithm takes as input $params, gpk, ok_{User}, \sigma, \mathcal{T}, M, reg$, and returns $i \geq 0$, an integer. If $i \geq 1$, the algorithm is claiming that the group member with identity i produced σ, and if $i = 0$, it is claiming that no group member produced σ.*
- $\mathsf{OpenAtt}(params, gpk, ok_{Att}, \sigma, M, \mathcal{T})$*: This is a deterministic algorithm takes as input $params, gpk, ok_{Att}, \sigma, \mathcal{T}, M, reg$, and returns the attribute set $\zeta \subseteq$ Att. Here it claims that ζ is the valid attribute set that is used in producing σ. If $\zeta = \phi$, then the algorithm claim that the valid attribute set has not been used in producing σ.*
- $\mathsf{Revoke}(params, gpk, ik, i)$ *: This algorithm takes $params, gpk, ik$ and i as an input and revokes the group member with identity i.*

If the access tree T is changed to T', then GM runs $\mathsf{BuildTree}(params, gpk, ik, T')$, and opens $\mathcal{T}'$, a public information associated with T, and store it in a public repository. When any new attribute att_{m+1} is added to an access tree, an attribute certificate corresponding to that specific attribute att_{m+1} needs to be issued for the eligible user(s) only, by the GM.

Definition 4 (`Attribute Anonymity`). *We say that the ABGS scheme preserve* Attribute Anonymity *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- `Setup` : *The challenger runs* `KeyGen`$(params)$*, and obtains gpk, ik, ok_{User}* and ok_{Att} *then he runs* `BuildTree`$(params, gpk, ik, T)$ *for some initial access tree T and obtains $\mathcal{T}$. Challenger gives $params, gpk, \mathcal{T}, ok_{User}$* and ik to $\mathcal{A}$.
- `Phase1` : *$\mathcal{A}$ can send following queries to the challenger,*

  - `Join` : *$\mathcal{A}$ can request the challenger, the Join procedure for any honest member of her choice. $\mathcal{A}$ plays the role of corrupted GM on these queries.*
  - `Signing` : *$\mathcal{A}$ can request a group signature σ on any, message M, access tree T, U_i and set of attributes $\zeta_i \subseteq \Upsilon_i$ such that $\zeta_i \models T$, of her choice.*
  - `Corruption` : *$\mathcal{A}$ can request the secret key sk_i of the members U_i of her choice.*
  - `OpenUser` : *$\mathcal{A}$ can request the signer's identity of some valid group signatures σ.*
  - `OpenAtt` : *$\mathcal{A}$ can request the attribute set of some valid group signatures σ.*
  - `Re − BuildTree` : *$\mathcal{A}$ can request the public values for any access tree T. The challenger returns $\mathcal{T}$.*
- `Challenge` : *$\mathcal{A}$ outputs M^*, T^*, a non-corrupted user U_i (which is not queried in* `Corruption` *queries) such that there exists two attribute sets $\zeta_{i_0}, \zeta_{i_1} \subseteq \Upsilon_i$ and $\zeta_{i_0} \models T^*, \zeta_{i_1} \models T^*$ holds. Then the challenger uniformly selects $b \in_R \{0,1\}$, uses ζ_{i_b} to make a group signature σ^* on M^* and returns σ^* to $\mathcal{A}$.*
- `Phase2` : *$\mathcal{A}$ can make the* `Signing`, `Corruption`, `Join`, `OpenUser`, `OpenAtt` *and* `Re − BuildTree` *queries. Note that* `Corruption` *query includes U_i and* `OpenAtt` *query does not include σ^*.*
- `Output` : *$\mathcal{A}$ outputs a bit b', and wins if $b' = b$.*

The advantage of $\mathcal{A}$ is defined as $Adv^{att-anon}(\mathcal{A}) = |Pr(b = b') - \frac{1}{2}|$.

In `Join` queries, $\mathcal{A}$ can play the role of corrupted GM (the same as in `SndToU` [5] oracle).

Definition 5 (`Anonymity`)**.** *We say that the ABGS scheme preserve* Anonymity *if for all PPT $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible.*

- `Setup` : *The challenger runs* KeyGen(*params*)*, and obtains gpk, ik, ok_{User}* and *ok_{Att} then he runs* `BuildTree`(*$params, gpk, ik, T$*) *for some initial access tree T and obtains $\mathcal{T}$. Challenger gives $params, gpk, \mathcal{T}, ok_{Att}$* and *$ik$* to *$\mathcal{A}$*.
- `Phase1` : *$\mathcal{A}$ can issue the* `Signing`, `Corruption`, `Join`, `OpenUser`, `OpenAtt` *and* `Re − BuildTree` *queries. All queries are the same as in* Attribute Anonymity *game.*
- `Challenge` : *$\mathcal{A}$ outputs M^*, T^*, and a non-corrupted users $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ and ζ. Note that $\zeta \subseteq \Upsilon_{i_0}, \zeta \subseteq \Upsilon_{i_1}$ and $\zeta \models T^*$. The challenger randomly selects $b \in_R \{0,1\}$ and responds with a group signature σ^* on M^* of group member U_{i_b}.*
- `Phase2` : *$\mathcal{A}$ can make the* `Signing`, `Corruption`, `OpenUser`, `OpenAtt`, `Join` *and* `Re − BuildTree` *queries. Note that* `Corruption` *query includes both $\mathrm{U}_{i_0}, \mathrm{U}_{i_1}$ and* `OpenUser` *query does not include σ^*.*
- `Output` : *$\mathcal{A}$ outputs a bit b', and wins if $b' = b$.*

The advantage of $\mathcal{A}$ is defined as $Adv^{anon}(\mathcal{A}) = |Pr(b = b') - \frac{1}{2}|$.

The definition of `Traceability`, `Non-frameability` and `Collusion resistance of Attribute Certificates` are similar to the one given in [14].

3 Proposed Scheme

An ABGS scheme with attribute anonymity together with their procedure of assignment of secret values to the access trees are presented in this section.

3.1 Assignment of Secret Values to Access Trees

To make the scheme suitable when there is a frequent change in attributes relationships we used the `Bottom-up Approach` for the construction of access tree, which is given by Emura et al. in [14] which includes three functions,

1. $\texttt{AddDummyNode}(T)$: This algorithm takes as input an access tree T, adds dummy nodes to it and returns the *extended access tree* T^{ext}. Let D_T be a set of dummy nodes added. Let $s_j \in \mathbb{Z}_p^*$ be a secret value for an attribute $\mathrm{att}_j \in Att$. Let $S = \{s_j\}_{\mathrm{att}_j \in Att}$.
2. $\texttt{AssignedValue}(p, S, T^{ext})$: This algorithm takes as input p, S and T^{ext} and returns a secret value for each dummy node $\{s_{d_j}\}_{d_j \in D_T}$ and a root secret s_T.
3. $\texttt{MakeSimplifiedTree}(Leaves, T^{ext})$: This algorithm takes as input the attributes set $Leaves \subseteq Att$ satisfying $Leaves \models T$, and returns the product of Lagranges coefficients $\Delta_{leaf}(\forall leaf \in Leaves \cup D_T^{Leaves})$. Such that,

$$\sum_{\mathrm{att}_j \in Leaves} \Delta_{\mathrm{att}_j} s_j + \sum_{d_j \in D_T^{Leaves}} \Delta_{d_j} s_{d_j} = s_T \qquad (1)$$

   holds, where $D_T^{Leaves} \subseteq D_T$ is the set of dummy nodes related to *Leaves*.

For more details of these algorithms refer [14].

3.2 Attribute Based Group Signature Scheme with Attribute Anonymity

In this section, we presents the variation of Emura's et al. ABGS scheme which achieves *Attribute Anonymity*. Let `NIZK` be a Non-Interactive Zero-Knowledge proof, `SPK` be the Signature Proof of Knowledge, and `Ext-Commit` be an extractable commitment scheme which uses the Pailler's encryption scheme [24], which is required in the security proof of the *Traceability*.

- $\texttt{Setup}(1^k)$:
  1. Define the cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ of prime order p, an one-way isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, a bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, and a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p^*$.
  2. Define the attributes $Att = \{\mathrm{att}_1, \mathrm{att}_2, ..., \mathrm{att}_m\}$.
  3. Outputs the system parameters, $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, Att)$.

- $\texttt{KeyGen}(params)$:
  1. Select the generators $g_1, g_3, g_4 \in \mathbb{G}_1$, and $g_2 \in \mathbb{G}_2$.

2. GM selects $\gamma \in_R \mathbb{Z}_p^*$, and computes $w = g_2^{\gamma}$.
3. GM selects $\mu \in_R \mathbb{Z}_p^*$, and computes $\mathcal{U} = g_2^{\frac{1}{\mu}}$.
4. GM selects $x'_1, x'_2, y'_1, y'_2, z_1, z_2 \in_R \mathbb{Z}_p^*$ and computes $C = g_3^{x'_1} g_4^{x'_2}, D = g_3^{y'_1} g_4^{y'_2}, E_1 = g_3^{z_1}$ and $E_2 = g_3^{z_2}$.
5. For each $\mathrm{att}_j \in \mathit{Att}$, GM selects $s_j \in_R \mathbb{Z}_p^*$, sets $S = \{s_j\}_{\mathrm{att}_j \in \mathit{Att}}$, and computes $g_{\mathrm{att}_j} = g_2^{s_j} (\forall \mathrm{att}_j \in \mathit{Att})$.
6. Outputs an user opening key, $ok_{User} = z_1$, an attribute opening key, $ok_{Att} = z_2$, an issuing key, $ik = (\gamma, \mu, S)$, and a group public key, $gpk = (g_1, g_2, g_3, g_4, w, \mathcal{U}, C, D, E_1, E_2, \{g_{\mathrm{att}_j}\}_{\mathrm{att}_j \in \mathit{Att}})$.

– $\texttt{BuildTree}(params, gpk, ik, T)$:
  1. GM runs $T^{ext} \leftarrow \texttt{AddDummyNode}(T)$ and $\texttt{AssignedValue}(p, S, T^{ext})$ and gets $\{s_{d_j}\}_{d_j \in D_T}, s_T$.
  2. GM computes $v = g_2^{s_T}$.
  3. Outputs $\mathcal{T} = (\{s_{d_j}\}_{d_j \in D_T}, v, T^{ext})$.

  The validity of $\mathcal{T}$ is publicly verifiable, see appendix A. This help to reduce the trust level of GM and this scheme can be used as ABS by disposing ok_{User} and ok_{Att}.

– $\texttt{Join}(< params, gpk, ik, upk_i, \Upsilon_i >, < params, gpk, upk_i, usk_i >)$:
  U$_i$ gets sk_i[1] $= ((A_i, x_i, y_i), \{T_{i,j}\}_{\mathrm{att}_j \in \Upsilon_i})$, where (A_i, x_i, y_i) is a membership certificate, $\{T_{i,j}\}_{\mathrm{att}_j \in \Upsilon_i}$ is the set of attribute certificates and Υ_i is the set of U$_i$'s attributes.
  1. U$_i$ picks $y_i \in_R \mathbb{Z}_p^*$ and computes $c_i = \texttt{Ext-Commit}(y_i)$,
     $F_i = E^{y_i}$ and
     $\pi_1 = \texttt{NIZK}\{y_i : F_i = E^{y_i} \wedge c_i = \texttt{Ext-Commit}(y_i)\}$.
  2. U$_i$ sends F_i, c_i and π_1 to GM.
  3. GM checks π_1. If π_1 is not valid, then abort.
  4. GM selects $x_i \in_R \mathbb{Z}_p^*$ and computes $A_i = (g_1 F_i)^{1/(\gamma + x_i)}$,
     $B_i = e(g_1 F_i, g_2)/e(A_i, w)$,
     $D_i = e(A_i, g_2)$,
     $T_{i,j} = A_i^{s_j \mu} (\forall \mathrm{att}_j \in \Upsilon_i)$ and
     $\pi_2 = \texttt{NIZK}\{x_i, \{s_j \mu, s_j\}_{(\mathrm{att}_j \in \Upsilon_i)} : B_i = D_i^{x_i} \wedge T_{i,j} = A_i^{s_j \mu} (\forall \mathrm{att}_j \in \Upsilon_i) \wedge g_{\mathrm{att}_j} = g_2^{s_j} (\forall \mathrm{att}_j \in \Upsilon_i) \wedge e(T_{i,j}, \mathcal{U}) = e(A_i, g_{\mathrm{att}_j}) (\forall \mathrm{att}_j \in \Upsilon_i)\}$.
  5. GM sends $A_i, B_i, D_i, \{T_{i,j}\}_{\mathrm{att}_j \in \Upsilon_i}$ and π_2 to U$_i$.
  6. U$_i$ checks π_2. If π_2 is not valid, then abort.
  7. U$_i$ makes $S_{i,A_i} = DSig_{usk_i}(A_i)$ and sends to GM.
  8. GM verifies S_{i,A_i} with respect to upk_i and A_i. If S_{i,A_i} is valid, then GM sends x_i to U$_i$ and adds (U_i, A_i) to *reg*.
  9. U$_i$ checks the relation $e(A_i, g_2)^{x_i} e(A_i, w) e(E, g_2)^{-y_i} \overset{?}{=} e(g_1, g_2)$ to verify whether $A_i^{(x_i + \gamma)} = g_1 E^{y_i}$.

[1] Note that $x_i, T_{i,j}$ values can be store in a public repository, so the size of $sk'_i s$ can be reduce to two elements, i.e. $sk_i = (A_i, y_i)$.

GM chooses $s_{m+1} \in \mathbb{Z}_p^*$, and computes $g_{\mathrm{att}_{m+1}} = g_2^{s_{m+1}}$ when a new attribute att$_{m+1}$ is added. Let U$_i$ be issued $T_{i,m+1}$. Then GM computes $T_{i,m+1} = A_i^{s_{m+1}\mu}$ and $\pi_3 = \mathtt{NIZK}\{s_{m+1} : T_{i,m+1} = A_i^{s_{m+1}\mu} \wedge g_{\mathrm{att}_{m+1}} = g_2^{s_{m+1}} \wedge e(T_{i,m+1}, \mathcal{U}) = e(A_i, g_{\mathrm{att}_{m+1}})\}$, sends $T_{i,m+1}$ and π_3 to U$_i$, and opens $g_{\mathrm{att}_{m+1}}$.

– $\mathtt{Sign}(params, gpk, sk_i, \zeta, M, \mathcal{T})$:
  A signer U$_i$ signs a message $M \in \{0,1\}^*$ as follows:
  1. U$_i$ chooses $\zeta \subseteq \Upsilon_i$ as an input such that $\zeta \models T$.
  2. U$_i$ runs $\mathtt{MakeSimplifiedTree}(\zeta, T^{ext})$ and gets the corresponding Δ_{att_j} ($\forall \mathrm{att}_j \in \zeta$), and Δ_{d_j} ($\forall d_j \in D_T^\zeta$).
  3. Note that $\Sigma_{\mathrm{att}_j \in \zeta} \Delta_{\mathrm{att}_j} s_j + \Sigma_{d_j \in D^\zeta} \Delta_{d_j} s_{d_j} = s_T$. Let $s_{T_1} = \Sigma_{\mathrm{att}_j \in \zeta} \Delta_{\mathrm{att}_j} s_j$ and $s_{T_2} = \Sigma_{d_j \in D_T^\zeta} \Delta_{d_j} s_{d_j}$. Thus, $s_{T_1} + s_{T_2} = s_T$.
  4. U$_i$ selects $\alpha_1, \alpha_2, \alpha_3, \delta \in_R \mathbb{Z}_p^*$, and computes $C_1 = A_i E_1^{\alpha_1}$,
     $C_2 = g_3^{\alpha_1}, C_3 = g_4^{\alpha_1}, C_4 = (CD^{\beta_1})^{\alpha_1}$,
     $C_5 = \Pi_{\mathrm{att}_j \in \zeta} T_{i,j}^{\delta \Delta_{\mathrm{att}_j}} E_2^{\alpha_2} = A_i^{\delta \mu s_{T_1}} E_2^{\alpha_2}$,
     $C_6 = g_3^{\alpha_2}, C_7 = g_4^{\alpha_2}, C_8 = (CD^{\beta_2})^{\alpha_2}$,
     $C_9 = \Pi_{d_j \in D_T^\zeta} A_i^{\delta \Delta_{d_j} s_{d_j}} E_2^{\alpha_3} = A_i^{\delta s_{T_2}} E_2^{\alpha_3}$,
     $C_{10} = g_3^{\alpha_3}, C_{11} = g_4^{\alpha_3}, C_{12} = (CD^{\beta_3})^{\alpha_3}$,
     $C_{13} = C_1^\delta, C_{14} = E_1^\delta, C_{15} = A_i^\delta$,
     where $\beta_1 = \mathcal{H}(C_1, C_2, C_3), \beta_2 = \mathcal{H}(C_5, C_6, C_7)$ and $\beta_3 = \mathcal{H}(C_9, C_{10}, C_{11})$.
  5. U$_i$ sets $\tau = \alpha_1 x_i + y_i$, and computes
     $V = \mathtt{SPK}\{(\alpha_1, \alpha_2, \alpha_3, x_i, \tau) : \frac{e(C_1,w)}{e(g_1,g_2)} = \frac{e(E_1,g_2)^\tau e(E_1,w)^{\alpha_1}}{e(C_1,g_2)^{x_i}} \bigwedge C_2 = g_3^{\alpha_1} \bigwedge C_3 = g_4^{\alpha_1} \bigwedge C_4 = (CD^{\beta_1})^{\alpha_1} \bigwedge C_6 = g_3^{\alpha_2} \bigwedge C_7 = g_4^{\alpha_2} \bigwedge C_8 = (CD^{\beta_2})^{\alpha_2} \bigwedge C_{10} = g_3^{\alpha_3} \bigwedge C_{11} = g_4^{\alpha_3} \bigwedge C_{12} = (CD^{\beta_3})^{\alpha_3} \bigwedge C_{13} = C_1^\delta \bigwedge C_{14} = E_1^\delta \bigwedge \frac{e(C_{13},g_2)}{e(C_{15},g_2)} = e(C_{14}, g_2)^{\alpha_1} \bigwedge \frac{e(C_5,\mathcal{U}) e(C_9,g_2)}{e(C_{13},v)} = \frac{e(E_2,\mathcal{U})^{\alpha_2} e(E_2,g_2)^{\alpha_3}}{e(C_{14},v)^{\alpha_1}}\}(M)$.
     (a) U$_i$ chooses blinding values $r_{\alpha_1}, r_{\alpha_2}, r_{\alpha_3}, r_\delta, r_{x_i}, r_\tau \in_R \mathbb{Z}_p^*$.
     (b) U$_i$ computes $R_1 = \frac{e(E_1,g_2)^{r_\tau} e(E_1,w)^{r_{\alpha_1}}}{e(C_1,g_2)^{r_{x_i}}}$,
         $R_2 = g_3^{r_{\alpha_1}}, R_3 = g_4^{r_{\alpha_1}}, R_4 = (CD^{\beta_1})^{r_{\alpha_1}}$,
         $R_5 = g_3^{r_{\alpha_2}}, R_6 = g_4^{r_{\alpha_2}}, R_7 = (CD^{\beta_2})^{r_{\alpha_2}}$,
         $R_8 = g_3^{r_{\alpha_3}}, R_9 = g_4^{r_{\alpha_3}}, R_{10} = (CD^{\beta_3})^{r_{\alpha_3}}$,
         $R_{11} = C_1^{r_\delta}, R_{12} = E_1^{r_\delta}, R_{13} = e(C_{14}, g_2)^{r_{\alpha_1}}$,
         $R_{Att} = \frac{e(E_2,\mathcal{U})^{r_{\alpha_2}} e(E_2,g_2)^{r_{\alpha_3}}}{e(E_1,v)^{r_{\alpha_1}}}$.
     (c) U$_i$ computes $c = \mathcal{H}(gpk, M, \{C_i\}_{i=1}^{15}, \{R_i\}_{i=1}^{13}, R_{Att})$.
     (d) U$_i$ computes $s_{\alpha_1} = r_{\alpha_1} + c\alpha_1, s_{\alpha_2} = r_{\alpha_2} + c\alpha_2, s_{\alpha_3} = r_{\alpha_3} + c\alpha_3, s_\delta = r_\delta + c\delta, s_{x_i} = r_{x_i} + cx_i, s_\tau = r_\tau + c\tau,$.
         Thus, $V = (c, s_{\alpha_1}, s_{\alpha_2}, s_{\alpha_3}, s_\delta, s_{x_i}, s_\tau)$.
  6. Outputs $\sigma = (\{C_i\}_{i=1}^{15}, V)$.

A signer U$_i$ proves the knowledge of $(\alpha_1, \alpha_2, \alpha_3, \delta, x_i, \tau)$ which satisfies the above 14 relations described in SPK. The first relation captures whether a signer has a valid membership certificate issued by the Join algorithm or not. The last relation captures whether a signer has valid attribute certificates or not. And the last but one relation links valid membership certificate with valid attribute certificates. Note that the signature includes the 3 modules of Cramer-Shoup encryption scheme [12], used to encrypt membership certificate, attribute certificates and dummy nodes.

– `Verify`$(params, gpk, M, \sigma, \mathcal{T})$:
  A verifier verifies the group signature σ as follows,
  1. The verifier compute $\beta_1 = \mathcal{H}(C_1, C_2, C_3), \beta_2 = \mathcal{H}(C_5, C_6, C_7)$ and $\beta_3 = \mathcal{H}(C_9, C_{10}, C_{11})$.
  2. The verifier computes $\widetilde{R_1} = \frac{e(E_1,g_2)^{s_\tau} e(E_1,w)^{s_{\alpha_1}}}{e(C_1,g_2)^{s_{x_i}}} \left(\frac{e(g_1,g_2)}{e(C_1,w)}\right)^c$,
     $\widetilde{R_2} = g_3^{s_{\alpha_1}} \left(\frac{1}{C_2}\right)^c$, $\widetilde{R_3} = g_4^{s_{\alpha_1}} \left(\frac{1}{C_3}\right)^c$, $\widetilde{R_4} = (CD^{\beta_1})^{s_{\alpha_1}} \left(\frac{1}{C_4}\right)^c$,
     $\widetilde{R_5} = g_3^{s_{\alpha_2}} \left(\frac{1}{C_6}\right)^c$, $\widetilde{R_6} = g_4^{s_{\alpha_2}} \left(\frac{1}{C_7}\right)^c$, $\widetilde{R_7} = (CD^{\beta_2})^{s_{\alpha_2}} \left(\frac{1}{C_8}\right)^c$,
     $\widetilde{R_8} = g_3^{s_{\alpha_3}} \left(\frac{1}{C_{10}}\right)^c$, $\widetilde{R_9} = g_4^{s_{\alpha_3}} \left(\frac{1}{C_{11}}\right)^c$, $\widetilde{R_{10}} = (CD^{\beta_3})^{s_{\alpha_3}} \left(\frac{1}{C_{12}}\right)^c$,
     $\widetilde{R_{11}} = C_1^{s_\delta} \left(\frac{1}{C_{13}}\right)^c, \widetilde{R_{12}} = E_1^{s_\delta} \left(\frac{1}{C_{14}}\right)^c, \widetilde{R_{13}} = e(C_{14}, g_2)^{s_{\alpha_1}} \left(\frac{e(C_{15},g_2)}{e(C_{13},g_2)}\right)^c$ and
     $\widetilde{R_{Att}} = \left(\frac{e(E_2,\mathcal{U})^{s_{\alpha_2}} e(E_2,g_2)^{s_{\alpha_3}}}{e(C_{14},v)^{s_{\alpha_1}}}\right) \left(\frac{e(C_{13},v)}{e(C_5,\mathcal{U})e(C_9,g_2)}\right)^c$.
  3. The verifier checks whether
     $c \overset{?}{=} \mathcal{H}(gpk, M, \{C_i\}_{i=1}^{15}, \{\widetilde{R_i}\}_{i=1}^{13}, \widetilde{R_{Att}})$.

– `OpenUser`$(params, gpk, ok_{User}, \sigma, M, \mathcal{T}, reg)$:
  1. `GM` verifies the validity of σ by using `Verify`$(param, gpk, M, \sigma, \mathcal{T})$. If σ is not a valid signature, then `GM` outputs $\perp$.
  2. `GM` computes $A_i = \frac{C_1}{C_2^{z_1}}$.
  3. `GM` searches A_i in reg, and outputs identity i. If there is no entry in reg, then `GM` outputs 0.

– `OpenAtt`$(params, gpk, ok_{Att}, \sigma, M, \mathcal{T})$:
  1. `GM` verifies the validity of σ by using `Verify`$(param, gpk, M, \sigma, \mathcal{T})$. If σ is not a valid signature, then `GM` outputs $\perp$.
  2. `GM` computes $\widehat{A} = \frac{C_9}{{C_{10}}^{z_2}}$.
  3. For all $\zeta_k : \zeta_k \models T$, `GM` checks $\widehat{A} \overset{?}{=} C_{15}^{s_{T_2}^k}$, where $s_{T_2}^k = \Sigma_{d_j \in D_T^{\zeta_k}} \Delta_{d_j} s_{d_j}$. If any such ζ_k exists then `GM` outputs it else outputs ϕ.

– `Revoke`$(params, gpk, ik, \{k_j\}_{j=1}^r)$:
  This works almost exactly as in [13, 7]. Here `GM` revokes the users, $\{\mathrm{U}_{k_1}, ..., \mathrm{U}_{k_r}\}$. First we show how `GM` revoke the single user U_k:
  1. `GM` computes $\tilde{g}_2 = g_2^{\frac{1}{\gamma+x_k}}, \tilde{g}_1 = \psi(\tilde{g}_2), \tilde{g}_3 = g_3^{\frac{1}{\gamma+x_k}}, \tilde{g}_4 = g_4^{\frac{1}{\gamma+x_k}}, \tilde{E}_1 = E_1^{\frac{1}{\gamma+x_k}}, \tilde{w} = \tilde{g}_2^\gamma, \tilde{\mathcal{U}} = \tilde{g}_2^{\frac{1}{\mu}}, \tilde{C} = C^{\frac{1}{\gamma+x_k}}, \tilde{D} = D^{\frac{1}{\gamma+x_k}}$ and $\tilde{g}_{\mathrm{att}_j} = \tilde{g}_2^{s_j} (\forall \mathrm{att}_j \in Att)$.
  2. `GM` set new group public key, $\widetilde{gpk} = (\tilde{g}_1, \tilde{g}_2, \tilde{g}_3, \tilde{g}_4, \tilde{w}, \tilde{\mathcal{U}}, \tilde{C}, \tilde{D}, \tilde{E}_1, \{\tilde{g}_{\mathrm{att}_j}\}_{\mathrm{att}_j \in Att})$ and added it to the public repository.
  3. `GM` also computes $\tilde{\tilde{g}}_{\mathrm{att}_j} = \tilde{g}_2^{\mu s_j} (\forall \mathrm{att}_j \in Att)$ and $\tilde{\tilde{E}}_1 = \tilde{E}_1{}^\mu$.

4. GM set auxiliary values, $Aux = (A_k, x_k, \tilde{g}_1, \tilde{E}_1, \tilde{\tilde{E}}_1, \{\tilde{\tilde{g}}_{\text{att}_j}\}_{\text{att}_j \in Att})$ and send it to all group members.
5. GM outputs $\widetilde{gpk}, Aux$.

To revoke multiple users at a time, GM computes $\tilde{g}_2 = g_2^{\Pi_{j=1}^{r} \frac{1}{\gamma + x_{k_j}}}$, i.e. first compute the exponent value then perform exponentiation operation, and similarly other values, where $\{k_j\}_{j=1}^{r}$ are the user ids to be revoke.

– `Update`$(\widetilde{gpk}, Aux, sk_i)$:
Unrevoked user U_i updates their member certificates and attribute certificates, when user U_k is revoked, as follows:
  1.

$$\begin{aligned}
\tilde{A}_i &= \left(\frac{\tilde{g}_1 \tilde{E}_1^{\,y_i}}{A_i} \right)^{\frac{1}{x_i - x_k}} \\
&= \left(\frac{g_1^{\frac{1}{\gamma + x_k}} E_1^{\frac{1}{\gamma + x_k} y_i}}{(g_1 E_1^{y_i})^{\frac{1}{\gamma + x_i}}} \right)^{\frac{1}{x_i - x_k}} \\
&= \left(g_1^{\frac{1}{\gamma + x_k} - \frac{1}{\gamma + x_i}} E_1^{\left(\frac{1}{\gamma + x_k} - \frac{1}{\gamma + x_i}\right) y_i} \right)^{\frac{1}{x_i - x_k}} \\
&= \left(g_1^{\frac{x_i - x_k}{(\gamma + x_k)(\gamma + x_i)}} E_1^{\frac{x_i - x_k}{(\gamma + x_k)(\gamma + x_i)}} \right)^{\frac{1}{x_i - x_k}} \\
&= (\tilde{g}_1 \tilde{E}_1^{\,y_i})^{\frac{1}{\gamma + x_i}}.
\end{aligned}$$

Which is a valid member certificate.

  2. Similarly for each $att_j \in \Upsilon_i$ compute $\tilde{T}_{i,j} = \left(\frac{\tilde{\tilde{g}}_1 \tilde{\tilde{E}}_1^{\,y_i}}{T_{i,j}} \right)^{\frac{1}{x_i - x_k}}$
$= (\tilde{g}_1 \tilde{E}_1^{\,y_i})^{\frac{s_j \mu}{\gamma + x_i}}$, a valid attribute certificates.
  3. Outputs $\widetilde{sk}_i = (\tilde{A}_i, x_i, y_i, \{\tilde{T}_{i,j}\}_{att_j \in \Upsilon_i})$.

If multiple users have been revoked, say revoked user ids are $\{k_j\}_{j=1}^{r}$, then user U_i repeat this process r times each with x_{k_j}, for $j \in [1, r]$. Thus this scheme supports concurrent join and revocation, which makes it more suitable for dynamic groups. In this revocation mechanism the user is revoked by the publishing the value Aux and consequently all the unrevoked users need to get these updates to update their secrets and verifiers need to use the updated group public key to verify the signature from there on.

VLR Idea. The alternate revocation mechanism is also possible where revocation messages are only sent to signature verifiers, so that there is no need for unrevoked signers to updates their keys, this is refer as Verifier-Local Revocation (VLR) group signatures [8]. Using this revocation mechanism, only a fragment of the user's private key is placed on the revocation list RL, an extra data structure. To make it VLR ABGS scheme an additional values needs to be compute in the signature, i.e. $\hat{E_1}^{\beta} \in \mathbb{G}_2 : E_1 = \psi(\hat{E_1})$ and $E_1^{\beta\alpha_1}$, and revocation list contains $RL = \{A_{k_1}, ..., A_{k_r}\}$. Then the verifier after verifying the signature needs to perform revocation check whether the signer has been revoked or not, it can be done by checking $e(C_1/A, \hat{E_1}^{\beta}) \overset{?}{=} e(E_1^{\beta\alpha_1}, \hat{E_1})$, for each $A \in RL$. And it can be show that the scheme preserves $Selfless - Anonymity$ [8] under XDH assumption.

3.3 Properties

This signature contains 15 elements from $\mathbb{G}_1$(over 171 bits) and 7 elements from $\mathbb{Z}_p^*$, p is 170 bits. Therefore the signature can be encoded on 3755 bits(less than 470 bytes), a constant size signature. From the efficiency point of view most of the pairing values can be precomputed.

4 Security Analysis

In this section, we show that our scheme satisfies *Attribute Anonymity*, *Anonymity* and *Collision resistance of Attribute Certificates*. The *Traceability* and *Non-frameability* proofs are similar to the one given in [13, 14]. Let p, q_H and q_S be the order of bilinear groups, the number of hash queries and the number of signature queries, respectively.

Theorem 1. *The proposed ABGS scheme preserve Attribute Anonymity under XDH assumption(namely DDH assumption over $\mathbb{G}_1$) in random oracle model, i.e., $Adv^{att-anon}(A) \leq \frac{q_S+q_H}{p} + 2\epsilon_{ddh}$ holds, where ϵ_{ddh} is the DDH advantage of some algorithm.*

Proof. This proof uses the sequence of games [26] to prove the *Attribute Anonymity* of the proposed scheme under XDH assumption. Let $\mathcal{C}_j$ be the challenger of the j-th game and S_j be the event that the adversary $\mathcal{A}$ wins on j-th game. Let $\sigma^* = (\{C_i^*\}_{i=1}^{15}, V^*)$ be the challenge group signature.

Game 0: This is the original anonymity game defined in the Definition 7. $\mathcal{C}_0$ generates system parameters, $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, Att)$, chooses $x_1', x_2', y_1', y_2', z_{1_3}, z_{1_4}, z_{2_3}, z_{2_4} \in_R \mathbb{Z}_p^*$ and chooses generators $g_1, g_3, g_4 \in_R \mathbb{G}_1$ and $g_2 \in_R \mathbb{G}_2$. Moreover, $\mathcal{C}_0$ computes $C = g_3^{x_1'} g_4^{x_2'}, D = g_3^{y_1'} g_4^{y_2'}, E_1 = g_3^{z_{1_3}} g_4^{z_{1_4}}$ and $E_2 = g_3^{z_{2_3}} g_4^{z_{2_4}}$. Even though the computation of E_1 and E_2 components are slightly different from the actual key generation algorithm but it is perfectly valid as shown by W.Mao [23]. Other parameters are chosen according to the real scheme settings. $\mathcal{C}_0$ gives $params, gpk, ik$ and ok_{User} to $\mathcal{A}$. Note that $\mathcal{C}_0$ can answer all the queries generated by $\mathcal{A}$. After the phase 1 queries $\mathcal{A}$ outputs M^*, T^* and non-corrupted user U_i such that there exists two attribute sets $\zeta_0, \zeta_1 \subseteq \Upsilon_i$ which satisfies the access tree T^*, i.e. $\zeta_0 \models T^*$ and $\zeta_1 \models T^*$ holds, and $v^* = g_2^{s_{T^*}}$. The challenger $\mathcal{C}_0$ randomly selects $b \in_R \{0,1\}$, uses ζ_b to make a group signature σ^*, i.e. $\sigma^* \longleftarrow \texttt{Sign}(params, gpk, sk_i, \zeta_b, M^*, \mathcal{T}^*)$, and gives σ^* to $\mathcal{A}$. After phase 2 queries $\mathcal{A}$ outputs a bit b' and S_0 is the event that $b = b'$. Then the adversary's advantage $Adv^{att-anon}(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}| = |\Pr[S_0] - \frac{1}{2}|$.

Game 1. This is same as Game 0 except SPK V^*, which includes the back-patch of the hash function $\mathcal{H}$, is simulated as follows for the challenge query:

1. $\mathcal{C}_1$ selects $c^*, s^*_{\alpha_1}, s^*_{\alpha_2}, s^*_{\alpha_3}, s^*_\delta, s^*_{x_i}, s^*_\tau \in_R \mathbb{Z}_p^*$.
2. $\mathcal{C}_1$ computes $R_1^* = \frac{e(E_1,g_2)^{s^*_\tau} e(E_1,w)^{s^*_{\alpha_1}}}{e(C_1^*,g_2)^{s^*_{x_i}}} \left(\frac{e(g_1,g_2)}{e(C_1^*,w)}\right)^{c^*}$,
$R_2^* = g_3^{s^*_{\alpha_1}} \left(\frac{1}{C_2^*}\right)^{c^*}, R_3^* = g_4^{s^*_{\alpha_1}} \left(\frac{1}{C_3^*}\right)^{c^*}, R_4^* = (CD^{\beta_1})^{s^*_{\alpha_1}} \left(\frac{1}{C_4^*}\right)^{c^*}$,
$R_5^* = g_3^{s^*_{\alpha_2}} \left(\frac{1}{C_6^*}\right)^{c^*}, R_6^* = g_4^{s^*_{\alpha_2}} \left(\frac{1}{C_7^*}\right)^{c^*}, R_7^* = (CD^{\beta_2})^{s^*_{\alpha_2}} \left(\frac{1}{C_8^*}\right)^{c^*}$,
$R_8^* = g_3^{s^*_{\alpha_3}} \left(\frac{1}{C_{10}^*}\right)^{c^*}, R_9^* = g_4^{s^*_{\alpha_3}} \left(\frac{1}{C_{11}^*}\right)^{c^*}, R_{10}^* = (CD^{\beta_3})^{s^*_{\alpha_3}} \left(\frac{1}{C_{12}^*}\right)^{c^*}$,
$R_{11}^* = C_1^{*s_\delta} \left(\frac{1}{C_{13}^*}\right)^{c^*}, R_{12}^* = E_1^{s^*_\delta} \left(\frac{1}{C_{14}^*}\right)^{c^*}, R_{13}^* = e(C_{14}^*, g_2)^{s^*_{\alpha_1}} \left(\frac{e(C_{15}^*,g_2)}{e(C_{13}^*,g_2)}\right)^{c^*}$ and
$R_{Att}^* = \left(\frac{e(E_2,\mathcal{U})^{s^*_{\alpha_2}} e(E_2,g_2)^{s^*_{\alpha_3}}}{e(C_{14}^*,v^*)^{s^*_{\alpha_1}}}\right) * \left(\frac{e(C_{13}^*,v^*)}{e(C_5^*,\mathcal{U}) e(C_9^*,g_2)}\right)^{c^*}$.
3. $\mathcal{C}_1$ defines $c^* = \mathcal{H}(gpk, M^*, \{C_i^*\}_{i=1}^{15}, \{R_i^*\}_{i=1}^{13}, R_{Att}^*)$.
4. $\mathcal{C}_1$ outputs $V^* = (c^*, s^*_{\alpha_1}, s^*_{\alpha_2}, s^*_{\alpha_3}, s^*_\delta, s^*_{x_i}, s^*_\tau)$.

If the simulation of the SPK fails then $\mathcal{C}_1$ aborts. The simulation fails if there is a collision while patching the hash oracle at $(gpk, M^*, \{C_i^*\}_{i=1}^{15}, \{R_i^*\}_{i=1}^{13}, R_{Att}^*)$ to c^*, in the challenge phase. This probability of collision, $\Pr[\texttt{abort}] \leq (q_S + q_H)/p$. If simulation succeed then after the phase 2 queries $\mathcal{A}$ output a bit b'. S_1 is the event that $b = b'$. Then $|\Pr[S_0] - \Pr[S_1]| = \Pr[\texttt{abort}]$ holds.

Game 2. This is same as Game 1 except C_5^*, C_6^*, C_7^* and C_8^* are constructed as follows: $\mathcal{C}_2$ chooses $\alpha_2, \alpha_2' \in_R \mathbb{Z}_p^*$ such that $\alpha_2 \neq \alpha_2'$, sets $U = g_3^{\alpha_2}, V = g_4^{\alpha_2'}$, and computes $C_5^* = A_i^{\mu\delta s_{T_1}} U^{z_{23}} V^{z_{24}}, C_6^* = U, C_7^* = V$ and $C_8^* = U^{x_1' + y_1'\beta_2} V^{x_2' + y_2'\beta_2}$, where $\beta_2 = \mathcal{H}(C_5^*, C_6^*, C_7^*)$. The challenger $\mathcal{C}_2$ gives $\sigma^* = (\{C_i^*\}_{i=1}^{15}, V^*)$ to $\mathcal{A}$. $\mathcal{A}$ output a bit b' after the phase 2 queries. S_2 is the event that $b = b'$. Note if $\alpha_2' = \alpha_2$, then $C_5^* = A_i^{\mu\delta s_{T_1}} g_3^{\alpha_2 z_{23}} g_4^{\alpha_2' z_{24}} = A_i^{\mu\delta s_{T_1}} E_2^{\alpha_2}, C_6^* = g_3^{\alpha_2}, C_7^* = g_4^{\alpha_2'} = g_4^{\alpha_2}$ and $C_8^* = (g_3^{x_1'} g_4^{x_2'})^{\alpha_2} (g_3^{y_1'} g_4^{y_2'})^{\alpha_2'\beta_2} = (CD^{\beta_2})^{\alpha_2}$ holds. This is the same as Game 1. Therefore, obviously $|\Pr[S_1] - \Pr[S_2]| = \epsilon_{ddh}$ holds.

Game 3. This is same as Game 2 except $C_9^*, C_{10}^*, C_{11}^*$ and C_{12}^* are constructed as follows: $\mathcal{C}_3$ chooses $\alpha_3, \alpha_3' \in_R \mathbb{Z}_p^*$ such that $\alpha_3 \neq \alpha_3'$, sets $U = g_3^{\alpha_3}, V = g_4^{\alpha_3'}$, and computes $C_9^* = A_i^{\delta s_{T_2}} U^{z_{23}} V^{z_{24}}, C_{10}^* = U, C_{11}^* = V$ and $C_{12}^* = U^{x_1' + y_1'\beta_3} V^{x_2' + y_2'\beta_3}$, where $\beta_3 = \mathcal{H}(C_9^*, C_{10}^*, C_{11}^*)$. The challenger $\mathcal{C}_2$ gives

$\sigma^* = (\{C_i^*\}_{i=1}^{15}, V^*)$ to $\mathcal{A}$. $\mathcal{A}$ output a bit b'. S_3 is the event that $b = b'$. Note if $\alpha_3' = \alpha_3$, then $C_9^* = A_i^{\delta s_{T_2}} g_3^{\alpha_3 z_{2_3}} g_4^{\alpha_3' z_{2_4}} = A_i^{\delta s_{T_2}} E_2^{\alpha_3}, C_{10}^* = g_3^{\alpha_3}, C_{11}^* = g_4^{\alpha_3'} = g_4^{\alpha_3}$ and $C_{12}^* = (g_3^{x_1'} g_4^{x_2'})^{\alpha_3} (g_3^{y_1'} g_4^{y_2'})^{\alpha_3' \beta_3} = (CD^{\beta_3})^{\alpha_3}$ holds. This is the same as Game 2. Therefore, $|\Pr[S_2] - \Pr[S_3]| = \epsilon_{ddh}$ holds.

Note that $\Pr[S_3] = \frac{1}{2}$ because all parts of the challenge group signature in the case of ζ_0 and all parts of the challenge group signature in the case of ζ_1 have the same distributions. Combining all the probabilistic relations from Game 0 to Game 3, $Adv^{att-anon}(\mathcal{A}) = \Pr[S_0] - \frac{1}{2} \leq \frac{q_S + q_H}{p} + 2\epsilon_{ddh}$ holds.

Theorem 2. *The proposed scheme satisfies Anonymity under XDH assumption(namely DDH assumption over $\mathbb{G}_1$) in the random oracle model, i.e., $Adv^{anon}(\mathcal{A}) \leq \frac{q_S + q_H}{p} + 4\epsilon_{ddh}$ holds, where ϵ_{ddh} is the DDH-advantage of some algorithms.*

Proof. This proof also uses the sequence of games [26] to prove the *Anonymity* of the proposed scheme under XDH assumption. Let $\mathcal{C}_j$ be the challenger of the j-th game and $\mathcal{S}_j$ be the event that the adversary wins on j-th game. Let $\sigma^* = (\{C_i^*\}_{i=1}^{15}, V^*)$ be the challenge group signature.

Game 0: This is the original anonymity game defined in the Definition 8. $\mathcal{C}_0$ generates system parameters, $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, Att)$, chooses $x_1', x_2', y_1', y_2', z_{1_3}, z_{1_4}, z_{2_3}, z_{2_4} \in_R \mathbb{Z}_p^*$ and chooses generators $g_1, g_3, g_4 \in_R \mathbb{G}_1$ and $g_2 \in_R \mathbb{G}_2$. Moreover, $\mathcal{C}_0$ computes $C = g_3^{x_1'} g_4^{x_2'}, D = g_3^{y_1'} g_4^{y_2'}, E_1 = g_3^{z_{1_3}} g_4^{z_{1_4}}$ and $E_2 = g_3^{z_{2_3}} g_4^{z_{2_4}}$. Even though the computation of E_1 and E_2 components are slightly different from the actual key generation algorithm but it is perfectly valid as shown by W.Mao [23]. Other parameters are chosen according to the real scheme settings. $\mathcal{C}_0$ gives $params, gpk, ik$ and ok_{Att} to $\mathcal{A}$. Note that $\mathcal{C}_0$ can answer all the queries generated by $\mathcal{A}$. After the phase 1 queries $\mathcal{A}$ outputs M^*, T^* and a non-corrupted users U_{i_0} and U_{i_1} such that there exists $\zeta \subseteq \Upsilon_{i_0}$ and $\zeta \subseteq \Upsilon_{i_1}$ which satisfies the access tree T^*, i.e. $\zeta \models T^*$ holds. The challenger $\mathcal{C}_0$ randomly selects $b \in_R \{0,1\}$, uses sk_{i_b} to make a group signature σ^*, i.e. $\sigma^* \longleftarrow \mathtt{Sign}(params, gpk, sk_{i_b}, \zeta, M^*, \mathcal{T}^*)$, and gives σ^* to $\mathcal{A}$. After phase 2 queries $\mathcal{A}$ outputs a bit b' and S_0 is the event that $b = b'$. Then the adversary's advantage $Adv^{anon}(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}| = |\Pr[S_0] - \frac{1}{2}|$.

Game 1. This is same as Game 0 except SPK V^*, which includes the back-patch of the hash function $\mathcal{H}$, is simulated as follows for the challenge query: It is similar to the `Game 1` in the proof of `Theorem 1`. Therefore $|\Pr[S_0] - \Pr[S_1]| = \Pr[\mathtt{abort}]$ holds.

Game 2. This is same as Game 1 except C_1^*, C_2^*, C_3^* and C_4^* are constructed as follows: $\mathcal{C}_2$ chooses $\alpha_1, \alpha_1' \in_R \mathbb{Z}_p^*$ such that $\alpha_1 \neq \alpha_1'$, sets $U = g_3^{\alpha_1}, V = g_4^{\alpha_1'}$, and computes $C_1^* = A_{i_b} U^{z_{1_3}} V^{z_{1_4}}, C_2^* = U, C_3^* = V$ and $C_4^* = U^{x_1' + y_1' \beta_1} V^{x_2' + y_2' \beta_1}$, where $\beta_1 = \mathcal{H}(C_1^*, C_2^*, C_3^*)$. The challenger $\mathcal{C}_2$ gives $\sigma^* = (\{C_i^*\}_{i=1}^{15}, V^*)$ to $\mathcal{A}$. $\mathcal{A}$ output a bit b' after the phase 2 queries. S_2 is the event that $b = b'$. Note if $\alpha_1' = \alpha_1$, then $C_1^* = A_{i_b} g_3^{\alpha_1 z_{1_3}} g_4^{\alpha_1' z_{1_4}} = A_{i_b} E_1^{\alpha_1}, C_2^* = g_3^{\alpha_1}, C_3^* = g_4^{\alpha_1'} = g_4^{\alpha_1}$ and

$C_4^* = (g_3^{x_1'} g_4^{x_2'})^{\alpha_1} (g_3^{y_1'} g_4^{y_2'})^{\alpha_1'\beta_1} = (CD^{\beta_1})^{\alpha_1}$ holds. This is the same as Game 1. Therefore, obviously $|\Pr[S_1] - \Pr[S_2]| = \epsilon_{ddh}$ holds.

Game 3. This is same as Game 2 except C_5^*, C_6^*, C_7^* and C_8^* are constructed as follows: $\mathcal{C}_3$ chooses $\alpha_2, \alpha_2' \in_R \mathbb{Z}_p^*$ such that $\alpha_2 \neq \alpha_2'$, sets $U = g_3^{\alpha_2}, V = g_4^{\alpha_2'}$, and computes $C_5^* = A_{i_b}^{\mu s_{T_1}} U^{z_{23}} V^{z_{24}}, C_6^* = U, C_7^* = V$ and $C_8^* = U^{x_1'+y_1'\beta_2} V^{x_2'+y_2'\beta_2}$, where $\beta_2 = \mathcal{H}(C_5^*, C_6^*, C_7^*)$. The challenger $\mathcal{C}_3$ gives $\sigma^* = (\{C_i^*\}_{i=1}^{15}, V^*)$ to $\mathcal{A}$. $\mathcal{A}$ output a bit b' after the phase 2 queries. S_3 is the event that $b = b'$. Note if $\alpha_2' = \alpha_2$, then $C_5^* = A_{i_b}^{\mu s_{T_1}} g_3^{\alpha_2 z_{23}} g_4^{\alpha_2' z_{24}} = A_{i_b}^{\mu s_{T_1}} E_2^{\alpha_2}, C_6^* = g_3^{\alpha_2}, C_7^* = g_4^{\alpha_2'} = g_4^{\alpha_2}$ and $C_8^* = (g_3^{x_1'} g_4^{x_2'})^{\alpha_2} (g_3^{y_1'} g_4^{y_2'})^{\alpha_2'\beta_2} = (CD^{\beta_2})^{\alpha_2}$ holds. This is the same as Game 2. Therefore, $|\Pr[S_2] - \Pr[S_3]| = \epsilon_{ddh}$ holds.

Game 4. This is same as Game 3 except $C_9^*, C_{10}^*, C_{11}^*$ and C_{12}^* are constructed as follows: $\mathcal{C}_4$ chooses $\alpha_3, \alpha_3' \in_R \mathbb{Z}_p^*$ such that $\alpha_3 \neq \alpha_3'$, sets $U = g_3^{\alpha_3}, V = g_4^{\alpha_3'}$, and computes $C_9^* = A_{i_b}^{s_{T_2}} U^{z_{23}} V^{z_{24}}, C_{10}^* = U, C_{11}^* = V$ and $C_{12}^* = U^{x_1'+y_1'\beta_3} V^{x_2'+y_2'\beta_3}$, where $\beta_3 = \mathcal{H}(C_9^*, C_{10}^*, C_{11}^*)$. The challenger $\mathcal{C}_4$ gives $\sigma^* = (\{C_i^*\}_{i=1}^{15}, V^*)$ to $\mathcal{A}$. $\mathcal{A}$ output a bit b' after the phase 2 queries. S_4 is the event that $b = b'$. Note if $\alpha_3' = \alpha_3$, then $C_9^* = A_{i_b}^{s_{T_2}} g_3^{\alpha_3 z_{23}} g_4^{\alpha_3' z_{24}} = A_{i_b}^{s_{T_2}} E_2^{\alpha_3}, C_{10}^* = g_3^{\alpha_3}, C_{11}^* = g_4^{\alpha_3'} = g_4^{\alpha_3}$ and $C_{12}^* = (g_3^{x_1'} g_4^{x_2'})^{\alpha_3} (g_3^{y_1'} g_4^{y_2'})^{\alpha_3'\beta_3} = (CD^{\beta_3})^{\alpha_3}$ holds. This is the same as Game 3. Therefore, $|\Pr[S_4] - \Pr[S_3]| = \epsilon_{ddh}$ holds.

Game 5. This is same as Game 4 except C_{13}^*, C_{14}^* and C_{15}^* are constructed as follows: $\mathcal{C}_5$ chooses $\delta, \delta', x \in_R \mathbb{Z}_p^*$ such that $\delta \neq \delta'$, sets $U = g_3^{\delta}, V = g_4^{\delta'}$, also set $g_1 = g_4^x$ and computes $C_{15}^* = (V^x U^{z_{13} y_{i_b}} V^{z_{14} y_{i_b}})^{\frac{1}{\gamma + x_{i_b}}}, C_{14}^* = U^{z_{13}} V^{z_{14}}$ and $C_{13}^* = C_{15}^* C_{14}^{*\alpha_1}$. The challenger $\mathcal{C}_5$ gives $\sigma^* = (\{C_i^*\}_{i=1}^{15}, V^*)$ to $\mathcal{A}$. $\mathcal{A}$ output a bit b' after the phase 2 queries. S_5 is the event that $b = b'$. Note if $\delta' = \delta$, then $C_{15}^* = (g_4^{x\delta'} g_3^{z_{13} y_{i_b} \delta} g_4^{z_{14} y_{i_b} \delta'})^{\frac{1}{\gamma + x_{i_b}}} = (g_1^{\delta} g_3^{z_{13} y_{i_b} \delta} g_4^{z_{14} y_{i_b} \delta})^{\frac{1}{\gamma + x_{i_b}}} = A_{i_b}^{\delta}, C_{14}^* = g_3^{z_{13}\delta} g_4^{z_{14}\delta'} = g_3^{z_{13}\delta} g_4^{z_{14}\delta} = E_1^{\delta}$ and $C_{13}^* = C_{15}^* C_{14}^{*\alpha_1} = C_1^{*\delta}$ holds. This is the same as Game 4. Therefore, $|\Pr[S_4] - \Pr[S_5]| = \epsilon_{ddh}$ holds.

Note that $\Pr[S_5] = \frac{1}{2}$ because all parts of the challenge group signature in the case of U_{i_0} and all parts of the challenge group signature in the case of U_{i_1} have the same distributions. Combining all the probabilistic relations from Game 0 to Game 5, $Adv^{anon}(\mathcal{A}) = \Pr[S_0] - \frac{1}{2} \leq \frac{q_S + q_H}{p} + 4\epsilon_{ddh}$ holds. And it is better than Emura et al.'s *Anonymity* adversary bounds.

Theorem 3. *We suppose an adversary $\mathcal{A}$ breaks the Traceability of the proposed scheme with the advantage ϵ. Then, in the random oracle model, we can construct an algorithm $\mathcal{B}$ that breaks the q-SDH assumption with the advantage $\frac{1}{6}\epsilon$.*

Proof. The proof is similar to the one given in [13, 14].

Theorem 4. *We suppose an adversary $\mathcal{A}$ breaks the Non-frameability of the proposed scheme with the advantage ϵ. Then, we can construct an algorithm $\mathcal{B}$ that breaks the DL assumption with the advantage $\frac{1}{12}(1+\frac{1}{n})\epsilon$, where n is the number of honest members.*

Proof. The proof is similar to the one given in [13, 14].

Theorem 5. *The proposed scheme preserved Collusion resistance of Attribute Certificates.*

Proof. We prove the theorem using two lemmas. In Lemma 1, we shown that it is negligible to produce a group signature using a forged attribute certificates. In Lemma 2, we shown that it is impossible to produce a group signature by colluding a valid attribute certificates of more than one group member.

Lemma 1. *The probability that a signature by forged attribute certificates passes the verification,* $\Pr(\mathsf{Verify}(params, gpk, M, \sigma, \mathcal{T}) = 1 \wedge \zeta \not\models T)$, *is atmost* $1/p$.

Proof. This proof is similar to the one given in proof 4 of [14].

Lemma 2. *Even if some malicious participants* $\mathrm{U}_{i_1}, ..., \mathrm{U}_{i_k} (k > 1)$ *with the set of attributes* $\zeta_{i_1}, ..., \zeta_{i_k}$ *collude, they cannot make a valid signature associated with an attribute tree* T, *where* $(\cup_{j=1}^{k} \zeta_{i_j} \models T)$ *and* $\zeta_{i_j} \not\models T (j = 1, ..., k)$ *with non-negligible probability.*

Proof. This proof is similar to the one given in proof 5 of [14].

5 Conclusion

We have proposed a solution to the attribute anonymity with the constant signature size for the dynamic ABGS scheme, and proven it is secure under random oracle model. We also gave the revocation mechanism which makes scheme attractive for dynamic groups. An efficiency of attribute set opening needs to be improve. A forward security and key insulated feature need to be added to overcome the group member's key exposure problem in the ABGS scheme. A threshold feature can be added to make it more suitable for some applications. And it is better to have ABGS scheme which is secure under standard model.

References

1. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: CRYPTO. pp. 255–270 (2000)
2. Ateniese, G., de Medeiros, B.: Efficient group signatures without trapdoors. In: Laih, C.S. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 2894, pp. 246–268. Springer (2003)
3. Ateniese, G., Tsudik, G.: Some open issues and new directions in group signatures. In: Franklin, M.K. (ed.) Financial Cryptography. Lecture Notes in Computer Science, vol. 1648, pp. 196–211. Springer (1999)

4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 2656, pp. 614–629. Springer (2003)
5. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 3376, pp. 136–153. Springer (2005)
6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 3027, pp. 56–73. Springer (2004)
7. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M.K. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 3152, pp. 41–55. Springer (2004)
8. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proceedings of the 11th ACM conference on Computer and communications security. pp. 168–177. CCS '04, ACM, New York, NY, USA (2004), `http://doi.acm.org/10.1145/1030083.1030106`
9. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: Jr., B.S.K. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1294, pp. 410–424. Springer (1997)
10. Chaum, D., van Heyst, E.: Group signatures. In: EUROCRYPT. pp. 257–265 (1991)
11. Chen, L., Pedersen, T.P.: New group signature schemes (extended abstract). In: EUROCRYPT. pp. 171–181 (1994)
12. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk [21], pp. 13–25
13. Delerablée, C., Pointcheval, D.: Dynamic fully anonymous short group signatures. In: Nguyen, P.Q. (ed.) VIETCRYPT. Lecture Notes in Computer Science, vol. 4341, pp. 193–210. Springer (2006)
14. Emura, K., Miyaji, A., Omote, K.: A dynamic attribute-based group signature scheme and its application in an anonymous survey for the collection of attribute statistics. In: ARES. pp. 487–492. IEEE Computer Society (2009)
15. Galbraith, S.D., Rotger, V.: Easy decision-diffie-hellman groups. Cryptology ePrint Archive, Report 2004/070 (2004), `http://eprint.iacr.org/`
16. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security. pp. 89–98. ACM (2006)
17. Hirschfeld, R. (ed.): Financial Cryptography, Second International Conference, FC'98, Anguilla, British West Indies, February 23-25, 1998, Proceedings, Lecture Notes in Computer Science, vol. 1465. Springer (1998)
18. Khader, D.: Attribute based group signature with revocation. Cryptology ePrint Archive, Report 2007/241 (2007), `http://eprint.iacr.org/`
19. Khader, D.: Attribute based group signatures. Cryptology ePrint Archive, Report 2007/159 (2007), `http://eprint.iacr.org/`
20. Kilian, J., Petrank, E.: Identity escrow. In: Krawczyk [21], pp. 169–185
21. Krawczyk, H. (ed.): Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings, Lecture Notes in Computer Science, vol. 1462. Springer (1998)

22. Maji, H., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Kiayias, A. (ed.) Topics in Cryptology CT-RSA 2011, Lecture Notes in Computer Science, vol. 6558, pp. 376–392. Springer Berlin / Heidelberg (2011), `http://dx.doi.org/10.1007/978-3-642-19074-2\_24`, 10.1007/978-3-642-19074-2_24
23. Mao, W.: Modern Cryptography: Theory and Practice. Prentice Hall Professional Technical Reference (2003)
24. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of the 17th international conference on Theory and application of cryptographic techniques. pp. 223–238. EUROCRYPT'99, Springer-Verlag, Berlin, Heidelberg (1999), `http://dl.acm.org/citation.cfm?id=1756123.1756146`
25. Qian, Y., Zhao, Y.: Strongly unforgeable attribute-based group signature in the standard model. pp. 843–852 (2010)
26. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), `http://eprint.iacr.org/`
27. Song, D.X.: Practical forward secure group signature schemes. In: ACM Conference on Computer and Communications Security. pp. 225–234 (2001)

A Verification of BuildTree

We know that, $g^{\Sigma_{i=\{1,\dots,n\}} s_i * L_i}$ is equivalent to $\prod_{i=\{1,\dots,n\}} g^{s_i * L_i}$, for $n \in \mathbb{Z}_p, \forall s_i, L_i \in \mathbb{Z}_p^*$, and g is from some cyclic group of prime order p. This is the idea which we applied here.

`Verify-BuildTree`$(params, gpk, \mathcal{T})$:

1. Compute $g_{d_j} = g_2^{s_{d_j}}, \forall d_j \in D_T$.
2. Randomly choose an attribute set,$Leaves \subseteq Att : Leaves \models T$.
3. Compute $g_{root} = \prod_{\text{att}_j \in Leaves} g_{\text{att}_j}^{\Delta_{\text{att}_j}} \times \prod_{d_j \in D_T^{Leaves}} g_{d_j}^{\Delta_{d_j}}$
   $= \prod_{\text{att}_j \in Leaves} g_2^{s_j \Delta_{\text{att}_j}} \times \prod_{d_j \in D_T^{Leaves}} g_2^{s_{d_j} \Delta_{d_j}}$
   $= g_2^{\sum_{\text{att}_j \in Leaves} \Delta_{\text{att}_j} s_j + \sum_{d_j \in D_T^{Leaves}} \Delta_{d_j} s_{d_j}}$
   $= g_2^{s_T}$ from (1).
4. Verify whether $g_{root} \overset{?}{=} v$. If not equals then $\mathcal{T}$ is the invalid public values of the access tree T.