

Title	A Dynamic Attribute-Based Group Signature Scheme and its Application in an Anonymous Survey for the Collection of Attribute Statistics
Author(s)	Emura, Keita; Miyaji, Atsuko; Omote, Kazumasa
Citation	International Conference on Availability, Reliability and Security, 2009. ARES '09.: 487-492
Issue Date	2009-03
Type	Conference Paper
Text version	publisher
URL	<a href="http://hdl.handle.net/10119/8484">http://hdl.handle.net/10119/8484</a>
Rights	Copyright (C) 2009 IEEE. Reprinted from International Conference on Availability, Reliability and Security, 2009. ARES '09., 487-492. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of JAIST's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to <a href="mailto:pubs-permissions@ieee.org">pubs-permissions@ieee.org</a> . By choosing to view this document, you agree to all provisions of the copyright laws protecting it.
Description	



# A Dynamic Attribute-Based Group Signature Scheme and its Application in an Anonymous Survey for the Collection of Attribute Statistics

Keita Emura, Atsuko Miyaji, and Kazumasa Omote

School of Information Science,  
Japan Advanced Institute of Science and Technology,  
1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan

**Abstract**—Recently, cryptographic schemes based on the user’s attributes have been proposed. An Attribute-Based Group Signature (ABGS) scheme is a kind of group signature schemes, where a user with a set of attributes can prove anonymously whether she has these attributes or not. An access tree is applied to express the relationships among some attributes. However, previous schemes do not provide the changing an access tree. In this paper, we propose a Dynamic ABGS scheme that enables an access tree to be changed. Our ABGS is efficient in that re-issuing of the attribute certificate previously issued for each user is not necessary. Moreover, calculations depending on the number of attributes are calculated on the domain of a pairing. Therefore, the number of calculations in a pairing does not depend on the number of attributes associated with a signature. Finally, we discuss how our ABGS can be applied to an anonymous survey for collection of attribute statistics.

**Index Terms**—Attribute-based system, Group signature, Anonymous Authentication, Anonymous survey

## I. INTRODUCTION

Recently, cryptographic schemes based on the user’s attributes have been proposed. Attribute-Based Group Signature (ABGS) schemes [9], [10] are a kind of Group Signature (GS) schemes [7], [11], where a user with a set of attributes can prove anonymously whether she has these attributes or not. ABGS schemes have been proposed by Khader [9], [10] using Goyal’s attribute-based encryption scheme [8] and Boneh’s GS scheme [5]. To the best of our knowledge, these are the only proposals for an ABGS. Usually, users have many kinds of attributes, and there exist some relationships among these attributes. An access tree [8], [9], [10] is applied to express these relationships. However, [9] and [10] schemes do not provide the changing of the relationships among attributes. This means that attributes and relationships among these attributes can be determined only once. This is not practical. Moreover, in all previous ABGSs [9], [10], the number of calculations in a pairing depends on the number of attributes associated with a signature.

**Our Contribution** : In this paper, we propose a Dynamic ABGS scheme that enables an access tree to be changed. Our ABGS is efficient in that re-issuing of the attribute certificate previously issued for each user is not necessary.

Moreover, calculations depending on the number of attributes are calculated on the domain of a pairing. Therefore, the number of calculations in a pairing does not depend on the number of attributes associated with a signature. Finally, we discuss how our ABGS can be applied to an anonymous survey for collection of attribute statistics.

**Organization** : The paper is organized as follows. Definitions are given in Section II. Our scheme is described in Section III. Security analysis is performed in Section IV. Efficiency comparisons are presented in Section V. The application of our ABGS in an anonymous survey for the Collection of Attribute Statistics is demonstrated in Section VI.

## II. DEFINITIONS

### A. Bilinear Groups

**Definition 1: (Bilinear Groups)** We use bilinear groups and a bilinear map defined as follows:

- 1)  $\mathbb{G}_1, \mathbb{G}_2$  and  $\mathbb{G}_3$  are cyclic groups of prime order  $p$ .
- 2)  $g_1$  and  $g_2$  are generators of  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , respectively.
- 3)  $\psi$  is an efficiently computable isomorphism  $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$  with  $\psi(g_2) = g_1$ .
- 4)  $e$  is an efficiently computable bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$  with the following properties.
  - Bilinearity : for all  $u, u' \in \mathbb{G}_1$  and  $v, v' \in \mathbb{G}_2$ ,  $e(uu', v) = e(u, v)e(u', v)$  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 unit).

### B. Access Tree

Let  $Att = \{att_1, \dots, att_m\}$  be a set of attributes. For  $\Gamma \subseteq 2^{Att} \setminus \{\emptyset\}$ ,  $\Gamma$  satisfies the monotone property: if  $\forall B, C \subseteq Att$ ,  $B \in \Gamma$  and  $B \subseteq C$ , then  $C \in \Gamma$  holds. Let access structures for  $Att$  be a set of  $\Gamma$  which satisfies the monotone property. An access tree [8], [9], [10]  $T$  is used for expressing an access structure by using a tree structure. An access tree is a tree, where threshold gates are defined on each interior node of the tree, and the leaves are associated with attributes. These attributes are subsets of  $Att$ . Let  $\ell_x$  be the number of children of node  $x$ , and  $k_x$  ( $0 < k_x \leq \ell_x$ ) be the threshold value on the

threshold gate of node  $x$ . We call the threshold gate “OR gate” when  $k_x = 1$ , and “AND gate” when  $k_x = \ell_x$ . The notation  $Leaves \models T$  expresses the fact that a set of attributes  $Leaves$  satisfies the access tree  $T$ .

### C. Model and Security Definitions

In this subsection, we define the model of an ABGS. An ABGS is a kind of GS, where a user  $U_i$  with a set of attributes  $\Gamma_i \subseteq Att = \{att_1, \dots, att_m\}$  can prove anonymously whether she has these attributes or not.  $U_i$  has a membership certificate  $A_i$  and a set of attribute certificates  $\{T_{i,j}\}_{att_j \in \Gamma_i}$ .  $U_i$  makes a group signature associated with  $\zeta \subseteq \Gamma_i$ . Usually, for a set of attributes  $Att$ , we construct an access tree to consider all relationships among these attributes. However, the access tree is changed when some threshold values are changed, or some attributes are deleted. Therefore, we define the model of the ABGS accepting a change of an access tree. We do not provide for the fact that a new attribute  $att \notin Att = \{att_1, \dots, att_m\}$  is added in an access tree. In this case, we have to re-issue an attribute certificate for users with  $att$  to execute the Join algorithm again. Let  $GM$  be the group manager.  $k$  the security parameter,  $params$  the system parameter,  $Att = \{att_1, \dots, att_m\}$  the universe of attributes,  $T_r$  the  $r$ -th access tree with a set of attributes  $\{att\}$ , where  $att \in Att$  is assigned on each leaf,  $\mathcal{T}_r$  the public values associated with  $T_r$ ,  $gpk$  the group public key,  $ik$  the group secret key which is used for issuing a membership certificate and making  $\mathcal{T}_r$ ,  $ok$  the opening key which is used for the opening procedure to reveal the signers' identification from the group signature,  $(upk_i, usk_i)$  the verification/signing key of a signature scheme  $DSig$ ,  $sk_i$  the member secret key for  $U_i$  ( $i = 1, 2, \dots, n$ ),  $\Gamma_i \subseteq Att$  attributes of  $U_i$ , and  $reg$  be the registration table for Open algorithm. Note that  $sk_i$  includes both  $A_i$  and  $\{T_{i,j}\}_{att_j \in \Gamma_i}$ . In Join algorithm, we use the notation  $Join(\langle \text{input of } GM \rangle, \langle \text{input of user} \rangle)$ .

**Definition 2:** ABGS

- **Setup**( $1^k$ ): This algorithm takes as input  $k$ , and returns  $params$ .
- **KeyGen**( $params$ ): This algorithm takes as input  $params$ , and returns  $gpk, ik, ok$  and  $reg = \emptyset$ .
- **BuildTree**( $params, ik, T_r$ ): This algorithm takes as input  $params, ik$  and  $T_r$  whose leaves are associated with a subset of  $Att$ , and returns  $\mathcal{T}_r$ .
- **Join**( $\langle params, gpk, ik, upk_i, \Gamma_i \rangle, \langle params, gpk, upk_i, usk_i \rangle$ ): This algorithm takes as input  $params, gpk, ik, upk_i$  and  $\Gamma_i$  from  $GM$ , and  $params, gpk, upk_i$  and  $usk_i$  from  $U_i$ , and returns  $sk_i$  and  $reg$ .
- **Sign**( $param, gpk, sk_i, M, \zeta_i, \mathcal{T}_r$ ): Let  $\zeta_i \subseteq \Gamma_i$  be a set of attributes such that  $\zeta_i \models T_r$ . This algorithm takes as input  $params, gpk, sk_i$ , a message  $M$ ,  $\zeta_i$  and  $\mathcal{T}_r$ , and returns  $\sigma$  associated with  $\zeta_i$ .
- **Verify**( $param, gpk, M, \sigma, \zeta, \mathcal{T}_r$ ): This algorithm takes as input  $params, gpk, \sigma, M, \sigma, \zeta$  and  $\mathcal{T}_r$ , and returns 1 if and only if  $\sigma$  is a valid signature.
- **Open**( $param, gpk, ok, \sigma, \zeta, \mathcal{T}_r, M, reg$ ): This algorithm takes as input  $params, gpk, ok, \sigma, \zeta, \mathcal{T}_r, M$  and  $reg$ ,

and returns the signer's identity  $i$ . If the signer is not included in  $reg$ , then this algorithm returns 0.

If the access tree  $T_r$  is changed to  $T_{r+1}$ , then  $GM$  runs  $BuildTree(params, ik, T_{r+1})$ , and opens  $\mathcal{T}_{r+1}$ , which is the public information associated with  $T_{r+1}$ .

**Definition 3:** Anonymity : Anonymity requires that for all PPT  $\mathcal{A}$ , the advantage of  $\mathcal{A}$  on the following game, is negligible.

- **Setup:** Let  $T_0$  be the initial access tree. The challenger runs  $KeyGen(params)$ , and obtains  $gpk, ik$  and  $ok$ . Moreover, the challenger runs  $BuildTree(params, ik, T_0)$ , and obtains  $\mathcal{T}_0$ .  $\mathcal{A}$  is given  $params, gpk, \mathcal{T}_0$  and  $ik$ .
- **Phase1:**  $\mathcal{A}$  can send these queries as follows:
  - **Join :**  $\mathcal{A}$  requests the join procedure for honest member  $U_i$ .  $\mathcal{A}$  plays the role of corrupted  $GM$  on these queries.
  - **Signing :**  $\mathcal{A}$  requests a group signature  $\sigma$  for all messages  $M$ , and all members  $U_i$  with a set of attributes  $\zeta_i \subseteq \Gamma_i$ .
  - **Corruption :**  $\mathcal{A}$  requests the secret key  $sk_i$  for all members  $U_i$ .
  - **Open :**  $\mathcal{A}$  requests the signer's identity with a message  $M$  and a valid signature  $\sigma$ .
  - **Re-BuildTree :**  $\mathcal{A}$  sends an access tree  $T_r$ . The challenger returns public values  $\mathcal{T}_r$ .
- **Challenge:**  $\mathcal{A}$  outputs  $M^*$ , non-corrupted users  $U_{i_0}, U_{i_1}$  and  $\zeta$ . Note that  $\zeta \subseteq \Gamma_{i_0}, \zeta \subseteq \Gamma_{i_1}$  and  $\zeta \models T^*$ , where  $T^*$  is the access tree on the challenge phase. The challenger uniformly selects  $b \in_R \{0, 1\}$ , and responds with a group signature on  $M^*$  by group member  $U_{i_b}$ .
- **Phase2:**  $\mathcal{A}$  can make the Signing, Corruption, Open, Join and Re-BuildTree queries. Note that Corruption queries include both  $U_{i_0}$  and  $U_{i_1}$ .
- **Output:**  $\mathcal{A}$  outputs a bit  $b'$ , and wins if  $b' = b$ .

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

In Join queries,  $\mathcal{A}$  can play the role of corrupted  $GM$  (the same as in  $SndToU$  oracle, which is defined in [2]). Moreover, we consider the Anonymity for Key-Exposure, namely, corruption queries for  $U_{i_0}$  and  $U_{i_1}$  can be admitted in Phase 2. Even after a secret key is exposed, signatures produced by the member before Key-Exposure remain anonymous. A similar definition of our Key-Exposure has been given in [3] for the ring signature scheme. Moreover, our definition is the CCA-Anonymity model [5], [7], namely, open queries in the Anonymity game can be admitted.

**Definition 4:** Traceability requires that for all PPT  $\mathcal{A}$ , the probability that  $\mathcal{A}$  wins the following game is negligible.

- **Setup:** Let  $T_0$  be the initial access tree. The challenger runs  $KeyGen(params)$ , and obtains  $gpk, ik$  and  $ok$ . Moreover, the challenger runs  $BuildTree(params, ik, T_0)$ , and obtains  $\mathcal{T}_0$ .  $\mathcal{A}$  is given  $params, gpk, \mathcal{T}_0$  and  $ok$ .

- **Queries:**  $\mathcal{A}$  can issue the Signing, Corruption, Join and Re-BuildTree queries. All queries are the same as in the Anonymity game, except Join.
  - **Join :**  $\mathcal{A}$  requests the Join procedure for corrupted member  $U_i$ .

- **Output:**  $\mathcal{A}$  outputs a message  $M^*$ ,  $\sigma^*$  and  $\zeta^*$ . Moreover,  $T^*$  is the access tree in this phase, and  $\mathcal{T}^*$  is the public information associated with  $T^*$ .

$\mathcal{A}$  wins if (1)  $\text{Verify}(params, gpk, M^*, \sigma^*, \zeta^*, T^*) = 1$ , (2)  $\text{Open}(params, gpk, ok, \sigma^*, \zeta^*, T^*, M^*, reg) = 0$ , and (3)  $\mathcal{A}$  has not obtained  $\sigma^*$  in Signing queries on  $M^*$ ,  $\zeta^*$  and  $T^*$ . The advantage of  $\mathcal{A}$  is defined as the probability of  $\mathcal{A}$  wins.

In Join queries,  $\mathcal{A}$  can play the role of corrupted users (the same as in SndTol oracle, which is defined in [2]).

*Definition 5:* Collusion-Resistance requires that for all PPT  $\mathcal{A}$ , the probability that  $\mathcal{A}$  wins the following game is negligible.

- **Setup:** Let  $T_0$  be the initial access tree. The challenger runs  $\text{KeyGen}(params)$ , and obtains  $gpk$ ,  $ik$  and  $ok$ . Moreover, the challenger runs  $\text{BuildTree}(params, ik, T_0)$ , and obtains  $\mathcal{T}_0$ .  $\mathcal{A}$  is given  $params$ ,  $gpk$  and  $\mathcal{T}_0$ .
- **Queries:**  $\mathcal{A}$  can issue the Signing, Corruption, Join and Re-BuildTree queries. All queries are the same as in the Anonymity game.
- **Output:** Finally,  $\mathcal{A}$  outputs  $M^*$ ,  $\sigma^*$  and  $\zeta^*$ .  $T^*$  is the access tree in this phase, and  $\mathcal{T}^*$  is the public information associated with  $T^*$ .

$\mathcal{A}$  wins if (1)  $\text{Verify}(params, gpk, M^*, \sigma^*, \zeta^*, T^*) = 1$ , and (2)  $\mathcal{A}$  has not obtained attribute certificates associated with  $\zeta^*$  corresponding to a single user.

This property indicates that, for example, there are two users  $U_{i_0}$  and  $U_{i_1}$  with  $\{T_{i_0, j}\}_{att_j \in \Gamma_{i_0}}$  and  $\{T_{i_1, j}\}_{att_j \in \Gamma_{i_1}}$ , respectively. We assume that  $\Gamma_{i_0} \subset \zeta^* \wedge \Gamma_{i_0} \neq \zeta^*$ ,  $\Gamma_{i_1} \subset \zeta^* \wedge \Gamma_{i_1} \neq \zeta^*$ , and that  $\zeta^* \subseteq \Gamma_{i_0} \cup \Gamma_{i_1}$  hold. Then  $U_{i_0}$  and  $U_{i_1}$  cannot make a valid group signature with  $\zeta^*$  even if  $U_{i_0}$  and  $U_{i_1}$  collude with each other.

*Definition 6:* Non-Frameability requires that for all PPT  $\mathcal{A}$ , the probability that  $\mathcal{A}$  wins the following game is negligible.

- **Setup:** Let  $T_0$  be the initial access tree. The challenger runs  $\text{KeyGen}(params)$ , and obtains  $gpk$ ,  $ik$  and  $ok$ . Moreover, the challenger runs  $\text{BuildTree}(params, ik, T_0)$ , and obtains  $\mathcal{T}_0$ .  $\mathcal{A}$  is given  $params$ ,  $gpk$ ,  $\mathcal{T}_0$ ,  $ik$  and  $ok$ .
- **Queries:**  $\mathcal{A}$  can issue the Signing, Corruption, Join and Re-BuildTree queries. All queries are the same as in the Anonymity game.
- **Output:** Finally,  $\mathcal{A}$  outputs a message  $M^*$ , an honest member  $U_{i^*}$ ,  $\sigma^*$  and  $\zeta^*$ .  $T^*$  is the access tree in this phase, and  $\mathcal{T}^*$  is the public information associated with  $T^*$ .

$\mathcal{A}$  wins if (1)  $\text{Verify}(params, gpk, M^*, \sigma^*, \zeta^*, T^*) = 1$ , (2)  $\sigma^*$  opens to an honest member  $U_{i^*}$ , (3)  $\mathcal{A}$  has not obtained  $\sigma^*$  in Signing queries on  $M^*$ ,  $U_{i^*}$  and  $\zeta^*$ , and (4)  $\mathcal{A}$  has not

obtained  $sk_{i^*}$  in Corruption queries on  $U_{i^*}$ . The advantage of  $\mathcal{A}$  is defined as the probability of  $\mathcal{A}$  wins.

### III. PROPOSED SCHEMES

In this section, an ABGS together with an assignment of secret values to access trees is presented.

#### A. Assignment of Secret Values to Access Trees

The previous schemes [10], [9] use a ‘‘Top-Down Approach’’ construction for access trees (when threshold gates are defined on each interior node of the tree and the leaves are associated with attributes) as follows:

- A secret value of the root node is chosen.
- A polynomial  $q_{root}(x)$  of degree ‘‘threshold value  $-1$ ’’ is defined such that  $q_{root}(0)$  equals the secret value of the root node.
- A secret value of a child node is defined such as  $q_{root}(index(child))$ .
- Secret values of all nodes can be defined to execute this procedure recursively.

If the access tree is changed, then the above Top-Down Approach construction has to be executed again. This means that the secret values that are associated with attributes have to be re-issued to corresponding users, because these values have to be changed. In our proposal, a ‘‘Bottom-Up Approach’’ construction is introduced. The order of our construction is *different from that of* the Top-Down Approach construction, namely, first all secret values are chosen for each attribute associated with each leaf. These secret values of leaves will not be changed when the access tree is changed.

**Idea :** For a node  $x$  associated with the threshold value  $k_x$ ,  $\ell_x - k_x$  dummy nodes will be opened, where  $\ell_x$  is the number of children of  $x$ . Next, the threshold value is changed from  $k_x$  to  $\ell_x$ . Then, all threshold gates become AND gates. Children with  $k_x$  or more can compute the secret value of their parent node by using the number of  $\ell_x - k_x$  public dummy nodes. We define functions **AddDummyNode** which adds dummy nodes to the access tree, **AssignedValue** which assigns secret values for nodes on the access tree, and **MakeSimplifiedTree** which makes a simplified tree associated with a set of leaves. Let  $index$  be the function which returns the index of the node, and  $p$  be a prime number. We assume that  $T$  includes  $Att$ .

$\langle \text{AddDummyNode}(T) \rangle$  : This algorithm takes as input  $T$ , and returns the extended access tree  $T^{ext}$ .

- 1) For an interior node  $x$  of  $T$ , the number of dummy nodes  $\ell_x - k_x$  is added to  $x$ ’s children.
- 2) The threshold value defined in  $x$  is changed from  $k_x$  to  $\ell_x$ .
- 3) All nodes are assigned unique index numbers.
- 4) The resulting tree, called  $T^{ext}$ , is outputted.

Let  $D_T$  be a set of dummy nodes determined by **AddDummyNode**. We assume that  $T^{ext}$  includes  $D_T$ . Moreover, let  $s_j \in \mathbb{Z}_p$  be a secret value for an attribute  $att_j \in Att$ . Let  $S = \{s_j\}_{att_j \in Att}$ .

$\langle \text{AssignedValue}(p, S, T^{ext}) \rangle$  : This algorithm takes as input  $p$ ,  $S$  and  $T^{ext}$  and returns a secret value  $s_x \in \mathbb{Z}_p$  for each node

$x$  of  $T^{ext}$ . Let  $\{child\}_x$  be the set of node  $x$ 's children except the dummy nodes, and  $\{d\}_x$  be the set of node  $x$ 's dummy nodes.

- 1) For an interior node  $x$  of  $T^{ext}$ , a polynomial  $q_x$  of degree  $\ell_x - 1$  is assigned as follows:
  - a) For  $att_j \in \{child\}_x$ , let  $q_x$  be a polynomial of degree at most  $\ell_x - 1$  which passes through  $(index(att_j), s_j)$ , where  $s_j \in S$  ( $j = 1, 2, \dots, \ell_x$ ).
  - b) For a dummy node  $d_j \in \{d\}_x$ , the secret value  $s_{d_j} := q_x(index(d_j))$  ( $j = 1, 2, \dots, \ell_x - k_x$ ) is assigned.
  - c) For  $x$ ,  $s_x := q_x(0)$  is assigned.
- 2) Repeat the above procedure up to the root node,  $s_T := q_{root}(0)$  is the secret value of  $T$ .
- 3) Output  $\{s_{d_j}\}_{d_j \in D_T}$  and  $s_T$ .

$\langle \text{MakeSimplifiedTree}(Leaves, T^{ext}) \rangle$ : This algorithm takes as input the set of attributes  $Leaves \subseteq Att$  satisfying  $Leaves \models T$ , and returns the simplified access tree  $T^{Leaves}$  (which is the access tree associated with  $Leaves$ ) and a product of Lagrange coefficients  $\Delta_{leaf}$ .

- 1) The set of attributes  $\{att_j\}_{att_j \in Att \setminus Leaves}$  are deleted from  $T^{ext}$ .
- 2) An interior node  $x$  has children less than the threshold value (namely,  $\ell_x$ ), and is deleted from  $T^{ext}$  along with  $x$ 's descendants.
- 3) Let  $D^{Leaves}$  be the set of dummy nodes which have remained after (1) and (2), and  $T^{Leaves}$  be the access tree after (1) and (2).
- 4) For all nodes  $x$  of  $T^{Leaves}$  except root, we define  $L_x$  as follows:
  - a) For  $x$ , define the depth 2 subtree of  $T^{Leaves}$  with  $x$  as leaf node. Let  $c_x$  be the set of indices of leaves.
  - b) Compute  $L_x := \prod_{k \in c_x} \frac{-k}{index(x) - k}$ .
- 5) Let  $leaf \in \{att_j \in Leaves\} \cup \{d_j \in D^{Leaves}\}$  be a leaf node of  $T^{Leaves}$ . For  $leaf$ , we define  $\Delta_{leaf}$  as follows:
  - a) Let  $Path_{leaf} := \{leaf, parent_1, \dots, parent_{n_{leaf}} = root\}$  be the set of nodes that appears in the path from  $leaf$  to root node.
  - b) Compute  $\Delta_{leaf} := \prod_{node \in Path_{leaf} \setminus root} L_{node}$ .
- 6) Output  $T^{Leaves}$ ,  $\Delta_j$  ( $att_j \in Leaves$ ),  $\Delta_{d_j}$  ( $d_j \in D^{Leaves}$ ).

Clearly,  $\sum_{att_j \in Leaves} \Delta_j s_j + \sum_{d_j \in D^{Leaves}} \Delta_{d_j} s_{d_j} = s_T$  holds.

### B. Proposed Attribute-Based Group Signature Scheme

In this subsection, we propose the ABGS by using our assignment (Section III-A). Our ABGS uses the Cramer-Shoup encryption scheme [6] for both CCA-Anonymity and Key-Exposure properties, and a concurrently secure Join algorithm proposed in [7]. Let  $NIZK$  be a Non-Interactive Zero-Knowledge proof,  $SPK$  be a Signature of Proof of Knowledge, and  $Ext-Commit$  be an extractable

commitment scheme. Let  $T_0$  be the initial access tree. Note that if an access tree is changed, then  $GM$  runs  $\text{BuildTree}(params, ik, T_{r+1})$ , and opens  $T_{r+1}$ , which is the public information associated with  $T_{r+1}$ .

- **Setup( $1^k$ )**
  - 1)  $GM$  selects cyclic groups of  $\mathbb{G}_1, \mathbb{G}_2$ , and  $\mathbb{G}_3$  with prime order  $p$ , an isomorphism  $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ , a bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ , and a hash function  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ .
  - 2)  $GM$  selects a generator  $g_2 \in \mathbb{G}_2$  and  $g_3, g_4 \in_R \mathbb{G}_1$ , and sets  $g_1 = \psi(g_2)$ .
  - 3)  $GM$  defines  $Att = \{att_1, att_2, \dots, att_m\}$ .
  - 4)  $GM$  outputs  $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \psi, \mathcal{H}, g_1, g_2, g_3, g_4, Att)$ .
- **KeyGen( $params$ )**
  - 1)  $GM$  selects  $\gamma \in_R \mathbb{Z}_p$ , and computes  $\omega = g_2^\gamma$ .
  - 2)  $GM$  selects  $x'_1, x'_2, y'_1, y'_2, z \in_R \mathbb{Z}_p$ , and computes  $C = g_3^{x'_1} g_4^{x'_2}, D = g_3^{y'_1} g_4^{y'_2}$  and  $E = g_3^z$ .
  - 3) For  $att_j \in Att$ ,  $GM$  selects  $s_j \in_R \mathbb{Z}_p^*$ , sets  $S = \{s_j\}_{att_j \in Att}$ , and computes  $g_{att_j} = g_2^{s_j}$  ( $att_j \in Att$ ).
  - 4) For  $att_j \in Att$ ,  $GM$  selects  $h_j \in_R \mathbb{G}_2$ , and sets  $\hat{h}_j = \psi(h_j)$ .
  - 5)  $GM$  outputs  $ok = (z), gpk = (\omega, C, D, E, \{h_j\}_{j=1}^m, \{g_{att_j}\}_{att_j \in Att})$  and  $ik = (\gamma, \{s_j\}_{att_j \in Att})$ .
- **BuildTree( $params, ik, T_0$ )**
  - 1)  $GM$  runs  $T_0^{ext} = \text{AddDummyNode}(T_0)$  and  $\text{AssignedValue}(p, S, T_0^{ext})$ , and gets  $\{s_{d_j}\}_{d_j \in D_{T_0}}$  and  $s_{T_0}$ .
  - 2)  $GM$  computes  $g_{d_j} = g_2^{s_{d_j}}$  ( $d_j \in D_{T_0}$ ) and  $v_0 = g_2^{s_{T_0}}$ .
  - 3)  $GM$  outputs  $\mathcal{T}_0 = (\{g_{d_j}\}_{d_j \in D_{T_0}}, v_0, T_0^{ext})$ .
- **Join( $\langle params, gpk, ik, upk_i, \Gamma_i \rangle, \langle params, gpk, upk_i, usk_i \rangle$ )**

$U_i$  gets  $sk_i = ((A_i, x_i, y_i), \{T_{i,j}\}_{att_j \in \Gamma_i})$ , where  $(A_i, x_i, y_i)$  is a member certificate and  $\{T_{i,j}\}_{att_j \in \Gamma_i}$  is the set of attribute certificates as follows:

  - 1)  $U_i$  picks  $y_i \in_R \mathbb{Z}_p$ , and computes  $c_i = \text{Ext-Commit}(y_i)$ ,  $F_i = E^{y_i}$  and  $\pi_1 = \text{NIZK}\{y_i : F_i = E^{y_i} \wedge c_i = \text{Ext-Commit}(y_i)\}$ .
  - 2)  $U_i$  sends  $F_i, c_i$  and  $\pi_1$  to  $GM$ .
  - 3)  $GM$  checks  $\pi_1$ . If  $\pi_1$  is not valid, then abort.
  - 4)  $GM$  selects  $x_i \in_R \mathbb{Z}_p$ , and computes  $A_i = (g_1 F_i)^{1/(\gamma+x_i)}$ ,  $B_i = e(g_1 F_i, g_2)/e(A_i, w)$ ,  $D_i = e(A_i, g_2)$ ,  $T_{i,j} = A_i^{s_j}$  ( $att_j \in \Gamma_i$ ) and  $\pi_2 = \text{NIZK}\{x_i, s_j$  ( $att_j \in \Gamma_i$ ) :  $B_i = D_i^{x_i} \wedge T_{i,j} = A_i^{s_j}$  ( $att_j \in \Gamma_i$ )  $\wedge g_{att_j} = g_2^{s_j}$  ( $att_j \in \Gamma_i$ )\}.
  - 5)  $GM$  sends  $A_i, B_i, D_i, \{T_{i,j}\}_{att_j \in \Gamma_i}$  and  $\pi_2$  to  $U_i$ .
  - 6)  $U_i$  checks  $\pi_2$ . If  $\pi_2$  is not valid, then abort.
  - 7)  $U_i$  makes  $S_{i,A_i} = \text{DSig}_{usk_i}(A_i)$ , and sends  $S_{i,A_i}$  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  $A_i^{(x_i+\gamma)} = g_1 E^{y_i}$  to verify whether  $e(A_i, g_2)^{x_i} e(A_i, w) e(E, g_2)^{-y_i} \stackrel{?}{=} e(g_1, g_2)$ .

- **Sign**( $param, gpk, sk_i, M, \zeta_i, T_r$ )

A signer  $U_i$  signs a message  $M \in \{0, 1\}^*$  as follows:

- 1)  $U_i$  chooses  $\zeta_i \subseteq \Gamma_i$  ( $\zeta_i \models T_r$ ) to associate  $\zeta_i$  with a group signature. Let  $|\zeta_i| = \phi$ .
- 2)  $U_i$  runs **MakeSimplifiedTree**( $\zeta_i, T_r^{ext}$ ), and gets  $T_r^{\zeta_i}$ ,  $\Delta_j$  ( $att_j \in \zeta_i$ ) and  $\Delta_{d_j}$  ( $d_j \in D_r^{\zeta_i}$ ).
- 3)  $U_i$  computes  $g_d = \prod_{d_j \in D_r^{\zeta_i}} g_{d_j}^{\Delta_{d_j}}$ .
- 4)  $U_i$  selects  $\alpha, \delta \in_R \mathbb{Z}_p$ , and computes  $C_1 = A_i E^\alpha$ ,  $C_2 = g_3^\alpha$ ,  $C_3 = g_4^\alpha$  and  $C_4 = (CD^\beta)^\alpha$ , where  $\beta = \mathcal{H}(C_1, C_2, C_3)$ .
- 5)  $U_i$  computes  $CT_j = T_{i,j} \hat{h}_j^\delta$  ( $att_j \in \zeta_i$ ).
- 6)  $U_i$  sets  $\tau = \alpha x_i + y_i$ , and computes  $V = SPK\{(\alpha, x_i, \tau, \delta) : \frac{e(C_1, \omega)}{e(g_1, g_2)} = \frac{e(E, g_2)^{\tau} \cdot e(E, \omega)^{\alpha}}{e(C_1, g_2)^{x_i}} \wedge C_2 = g_3^\alpha \wedge C_3 = g_4^\alpha \wedge C_4 = (CD^\beta)^\alpha \wedge \frac{e(\prod_{att_j \in \zeta_i} CT_j^{\Delta_j}, g_2)}{e(C_1, v_r/g_d)} = \frac{e(\prod_{att_j \in \zeta_i} \hat{h}_j^{\Delta_j}, g_2)^\delta}{e(E, v_r/g_d)^\alpha}\}(M)$ .

Concretely,  $U_i$  computes  $V$  as follows:

- a)  $U_i$  selects  $r_\alpha, r_{x_i}, r_\tau, r_\delta \in_R \mathbb{Z}_p$ .
  - b)  $U_i$  computes  $R_1 = \frac{e(E, g_2)^{r_\tau} e(E, \omega)^{r_\alpha}}{e(C_1, g_2)^{r_{x_i}}}$ ,  $R_2 = g_3^{r_\alpha}$ ,  $R_3 = g_4^{r_\alpha}$ ,  $R_4 = (CD^\beta)^{r_\alpha}$  and  $R_{Att} = \frac{e(\prod_{att_j \in \zeta_i} \hat{h}_j^{\Delta_j}, g_2)^{r_\delta}}{e(E, v_r/g_d)^{r_\alpha}}$ .
  - c)  $U_i$  computes  $c = \mathcal{H}(gpk, M, \{C_i\}_{i=1}^4, \{CT_i\}_{i=1}^\phi, \{R_i\}_{i=1}^4, R_{Att})$ .
  - d)  $U_i$  computes  $s_\alpha = r_\alpha + c\alpha$ ,  $s_{x_i} = r_{x_i} + cx_i$ ,  $s_\tau = r_\tau + c\tau$  and  $s_\delta = r_\delta + c\delta$ .
- 7)  $U_i$  outputs  $\sigma = (\{C_i\}_{i=1}^4, c, s_\alpha, s_{x_i}, s_\tau, s_\delta, \{CT_i\}_{i=1}^\phi)$

A signer  $U_i$  proves the knowledge of  $(\alpha, x_i, \tau, \delta)$  which satisfies the 5 above relations described in  $SPK$   $V$ . 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 associated with the set of attributes  $\zeta_i \models T_r$  or not.

- **Verify**( $param, gpk, M, \sigma, \zeta, T_r$ )

A verifier verifies a group signature  $\sigma$  associated with the set of attributes  $\zeta$ .

- 1) The verifier runs **MakeSimplifiedTree**( $\zeta, T_r^{ext}$ ), and gets  $T_r^\zeta$ ,  $\Delta_j$  ( $att_j \in \zeta$ ) and  $\Delta_{d_j}$  ( $d_j \in D_r^\zeta$ ). Let  $|\zeta| = \phi$ .
- 2) The verifier computes  $g_d = \prod_{d_j \in D_r^\zeta} g_{d_j}^{\Delta_{d_j}}$  and  $\beta = \mathcal{H}(C_1, C_2, C_3)$ .
- 3) The verifier computes  $\tilde{R}_1 = \frac{e(E, g_2)^{s_\tau} \cdot e(E, \omega)^{s_\alpha}}{e(C_1, g_2)^{s_{x_i}}}$ ,  $\tilde{R}_2 = \left(\frac{1}{C_2}\right)^c$ ,  $\tilde{R}_3 = g_4^{s_\alpha} \left(\frac{1}{C_3}\right)^c$ ,  $\tilde{R}_4 = (CD^\beta)^{s_\alpha} \left(\frac{1}{C_4}\right)^c$  and  $\tilde{R}_{Att} =$

$$\frac{e(\prod_{att_j \in \zeta_i} \hat{h}_j^{\Delta_j}, g_2)^{s_\delta}}{e(E, v_r/g_d)^{s_\alpha}} \left( \frac{e(C_1, v_r/g_d)}{e(\prod_{att_j \in \zeta_i} CT_j^{\Delta_j}, g_2)} \right)^c.$$

- 4) The verifier checks  $c \stackrel{?}{=} \mathcal{H}(gpk, M, gpk, M, \{C_i\}_{i=1}^4, \{CT_i\}_{i=1}^\phi, \{\tilde{R}_i\}_{i=1}^4, \tilde{R}_{Att})$ .

- **Open**( $param, gpk, ok, \sigma, \zeta, T_r, M, reg$ )

- 1)  $GM$  verifies the validity of  $\sigma$  by using **Verify**( $param, gpk, M, \sigma, \zeta, T_r$ ). If  $\sigma$  is not a valid signature, then  $GM$  outputs  $\perp$ .
- 2)  $GM$  computes  $A_i = C_1/C_2^z$ .
- 3)  $GM$  searches  $A_i$  from  $reg$ , and outputs identity  $i$ . If there is no entry in  $reg$ , then  $GM$  outputs 0.

#### IV. SECURITY

Let  $p, q_H$  and  $q_S$  be order of bilinear groups, and the number of hash queries and signature queries, respectively. Our scheme is based on the discrete logarithm (DL), eXternal Diffie-Hellman (XDH) [5] (it is the Decision Diffie-Hellman (DDH) assumption over  $\mathbb{G}_1$ ), and  $q$ -strong Diffie-Hellman ( $q$ -SDH) [4] assumptions.

**Theorem 1:** The proposed scheme satisfies Anonymity under the XDH assumption (namely DDH assumption over  $\mathbb{G}_1$ ), i.e.,  $Adv^{anon}(\mathcal{A}) \leq \frac{q_S q_H}{p} + m \cdot \epsilon_{\text{ddh}}$  holds, where  $\epsilon_{\text{ddh}}$  is the DDH-advantage of some algorithms and  $m = |Att|$ .

**Theorem 2:** We suppose an adversary  $\mathcal{A}$  breaks the Traceability of the proposed scheme with the advantage  $\epsilon$ . Then, we can construct an algorithm  $\mathcal{B}$  that breaks the  $q$ -SDH assumption with the advantage  $\frac{1}{6} \left(1 - \frac{1}{p}\right) \left(1 - \frac{q_S q_H}{p}\right) \epsilon$ .

**Theorem 3:** We suppose an adversary  $\mathcal{A}$  breaks the Non-Frameability of the proposed scheme with the advantage  $\epsilon$ . Then, we can construct an algorithm  $\mathcal{B}$  that breaks the DL assumption with the advantage  $\frac{1}{12} \left(1 + \frac{1}{n}\right) \left(1 - \frac{q_S q_H}{p}\right) \epsilon$ .

**Theorem 4:** The probability that a signature by forged attribute certificates passes the verification,  $\Pr(\text{Verify}(param, s, gpk, M, \sigma, \zeta, T) = 1 \wedge \zeta \not\models T)$ , is  $\frac{p^{\phi-1}-1}{p^\phi} = \frac{1}{p} \left(1 - \frac{1}{p^{\phi-1}}\right)$ , where  $\phi$  is the number of attributes associated with a signature.

**Theorem 5:** Even if some malicious participants  $U_{i_1}, \dots, U_{i_k}$  ( $k > 1$ ) with the set of attributes  $\zeta_{i_1}, \dots, \zeta_{i_k}$  collude, they cannot make a valid signature associated with an attribute tree  $T_r$ , where  $(\cup_{j=1}^k \zeta_{i_j}) \models T_r$  and  $\zeta_{i_j} \not\models T_r$  ( $j = 1, \dots, k$ ) with non-negligible probability.

We omit these proofs, and put them in the full version of this paper.

#### V. COMPARISONS

Let  $\zeta$  ( $|\zeta| = \phi$ ) be the set of attributes which is associated with a signature,  $D_\zeta$  be the set of dummy nodes which is defined as  $\zeta$ , and  $|Att| = m$ . Moreover, let  $r$  be the number of revoked members [9]. We assume that the computational estimations are made according to [11]. In our scheme, Verification costs are the lowest, because the number of calculations in a pairing does not depend on the number of attributes associated with a signature. There is room for argument regarding the Signing costs. Moreover, our scheme provides Dynamic property,

**Table 1. Comparisons**

	[10]	[9]	Our Scheme
Dynamic property	no	no	yes
CCA-Anonymity	no	no	yes
Non-Frameability	no	no	yes
Key-Exposure	no	no	yes
Signature Length	$1633 + 171\phi$	$1192 + 1191\phi$	$1634 + 171\phi$
Signing	$(12 + 2\phi)\mathbb{G}_1 + 5\mathbb{G}_3 + e$	$(7 + 2\phi)\mathbb{G}_1 + (5 + \phi)\mathbb{G}_3 + (\phi + 1)e$	$(9 + 3\phi)\mathbb{G}_1 + (\phi + 1)\mathbb{G}_2 + 8\mathbb{G}_3 + 3e$
Verification	$12\mathbb{G}_1 + (\phi + 8)\mathbb{G}_3 + (\phi + 1)e$	$(6 + 2r)\mathbb{G}_1 + (8 + 2\phi)\mathbb{G}_3 + (\phi + 2r + 1)e$	$(11 + 2\phi)\mathbb{G}_1 + (\phi + 1)\mathbb{G}_2 + 14\mathbb{G}_3 + 6e$

VI. APPLICATION OF ABGS IN ANONYMOUS SURVEY FOR COLLECTION OF ATTRIBUTE STATISTICS

In this section, we discuss how our ABGS can be applied to an anonymous survey for collection of attribute statistics. An anonymous survey is used as follows: When we apply the GS to a business system offering some services to group members, each member’s personal information is not exposed. Moreover, a service provider can verify whether each user is valid or not. However, it is difficult for a service provider to obtain a collection of user’s attribute statistics to improve service contents. In [13], an anonymous survey has been proposed which is a protocol executed among trusted third parties (TTPs). Moreover, the relationships among some attributes, e.g., (female  $\wedge$  20s), can be handled in the statistics information. However, a distributor cannot verify whether users properly construct the ciphertext or not. In [12], an anonymous survey has been proposed using the Open algorithm of Ateniese et. al. GS [1]. A distributor can verify whether users properly make the ciphertext or not, to verify the validity of group signatures. Because one attribute certificate is issued for an attribute type, it is difficult as the relationships among some attributes to be handled in the statistics information. There is an obvious solution: new attribute types such as  $att_C = att_A \wedge att_B$  are defined. However, the number of all attribute types are represented by  $O(2^m)$ , where  $m$  is the number of all attributes. We solve this attribute increase problem to apply an ABGS.

- 1) A user makes a group signature  $\sigma$  associated with the set of attributes  $\zeta$  to use our ABGS.
- 2) The user encrypts  $\zeta$  to use the public key of a distributor, and sends both  $\sigma$  and the encrypted  $\zeta$  to the distributor.
- 3) The distributor decrypts  $\zeta$ , and verifies whether  $\sigma$  is valid or not.
- 4) The statistics information is the collection of  $\zeta$ .

To collect the set of attributes  $\zeta$ , the distributor can obtain the statistics of attributes without any other information, because the distributor does not know the opening key which is used for the opening procedure to reveal the signer’s identification from the group signature. Moreover, the distributor can verify whether users properly made the ciphertext or not, to verify that the validity of group signatures is the same as in [12]. Moreover, the relationships among some attributes can be handled in the statistics information in the same way as in [13], without increasing the number of attribute certificates of each user. Indeed, the number of attribute certificates of each user is represented by  $O(m)$ . Of course, relationships

among some attributes which one wants to reflect with the statistics information are different in each case. Our scheme is suitable for use in the anonymous survey because the change of relationships is indispensable in the anonymous survey for the collection of attribute statistics.

VII. CONCLUSION

In this paper, we propose a Dynamic ABGS scheme that enables an access tree to be changed. Our ABGS is efficient in that re-issuing of the attribute certificate previously issued for each user is not necessary. Moreover, the number of calculations in a pairing does not depend on the number of attributes associated with a signature. A service provider obtains a collection of anonymous user’s attribute statistics to improve service contents by using our ABGS.

REFERENCES

- [1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, pages 255–270.
- [2] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, pages 136–153.
- [3] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography Conference, TCC 2006*, pages 60–79.
- [4] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, pages 56–73.
- [5] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO 2004*, pages 41–55.
- [6] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO ’98*, pages 13–25.
- [7] C. Delerablée and D. Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT 2006*, pages 193–210.
- [8] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006*, pages 89–98.
- [9] D. Khader. Attribute based group signature with revocation. Cryptology ePrint Archive, Report 2007/241, 2007.
- [10] D. Khader. Attribute based group signatures. Cryptology ePrint Archive, Report 2007/159, 2007.
- [11] T. Nakanishi and N. Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. In *International Workshop on Security, IWSEC 2006*, pages 17–32.
- [12] T. Nakanishi and Y. Sugiyama. An efficient anonymous survey for attribute statistics using a group signature scheme with attribute tracing. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 86(10):2560–2568, 2003.
- [13] K. Sako. Generating statistical information in anonymous surveys. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 79(4):507–512, 1996.