Authenticating with Attributes

Dalia Khader

University of Bath

Abstract. Attribute based group signature (ABGS) is a a new generation of group signature schemes. In such a scheme the verifier could define the role of a signer within the group. He determines the attributes possessed by the member signing the document. In this paper we propose the first ABGS scheme with multi-authorities and define its security notions. We construct an ABGS that is proved to be fully traceable and fully anonymous. Our scheme is efficient and secure.

1 Introduction

Attribute Based Group Signature (ABGS) is a new paradigm of cryptography. The idea behind it is authenticating that a person has certain credentials. The following is a scenario were such a scheme is needed:
Alice requests a prescription from Bob's pharmacy. Bob is willing to give her a discount if she is a student, an elder, or an employee in his pharmacy. He also wants a prove that an authorized public or private medical practice prescribed the medicine for her. He needs to check that she has a national health insurance. Alice and Bob have no previous knowledge about each other. Alice is sensitive about her details. She does not want to tell Bob whether she goes to a private practice or a public one. She definitely does not want her identity to be revealed. She doesn't want Bob to know whether she bought any medicine from him before. Bob, on the other hand, is not willing to break his policy and wants to make sure that any information he gets from Alice is true.
From this scenario we could derive the properties of the ABGS scheme:

- **No previous knowledge.** We can not assume that the communicators, Alice and Bob, know information about each other.
- **Attributes can not be forged.** Alice should prove possession of attributes. That prove must guarantee that the attributes Bob needs are included and they belong to Alice only.
- **Anonymity.** Bob should not be able to reveal the identity of Alice.
- **Linkability.** If Alice buys twice from Bob. He should not be able to tell that the first purchase and the second were done by her.
- **Privacy.** It is enough for Bob to know that Alice satisfies his policy. He does not need to know how. For example, in our scenario Bob could accept a prescription from a private or public practice. Alice should prove that she does have such prescription without revealing who gave it.

A possible solution is to use digital signatures. Alice has a certificate for each attribute she owns signed by trusted third party. Bob announces the attributes he requires. Alice encloses the certificates she owns. Bob verifies each certificate and makes a decision on whether Alice is eligible to have the medicine with a discount. The drawback of such a solution is the need to run the verification algorithm many times. Alice is sending the certificates she owns even though Bob requires a proof of satisfying his policy only, therefore breaching here privacy. Moreover, a colluding attack is possible. Alice may have some certificates Bob needs. She could get together with another person who has the rest of the certificates and they could pretend to be one person when dealing with Bob. Therefore, we could conclude that this solution is not ideal.
Another suggestion would be using identity based signatures (IBS) which are digital signatures that use the identity of the person in the verification process rather than a public key. Such a scheme could be considered as the first attempt to create attribute based signatures. Assume we concatenate identities in IBS schemes to some attribute templates. This would limit the possibility of colluding attacks. However, the verifier will need an identity of the signer in order to create his verification request. The signer is forced to attach his identity to the signature which leads to breaking the anonymity and linkability of the scheme.
To overcome the anonymity problem in the previous solutions, group signatures could be used. Group signatures are digital signatures that allow any member of a group to sign anonymously on behalf of others. We could present all owners of a certain attribute as a group. For example, all students may belong to a group. Even though, group signature solves one problem, it still requires the number of verifications to be equal to the number of attributes needed from the verifier. In other words Alice still has to enclose all certificates Bob needs. It is also vulnerable to colluding attacks.
The shortcoming of all three suggestions, makes ABGS a new cryptographical problem that requires creating a new scheme. In attribute based group signature, a verifier could request the role of a signer within a group. That request is presented as a verification key sent from the verifier to the group. The verification key is built with a main public key base and many public attribute keys. The verifier is the one who creates the verification key, while the public keys used are created by one or more trusted third parties. Any member of the group could sign if and only if they have enough attribute private keys to satisfy the request. Trusted third parties are divided into two types: central authority and attribute authority. The central authority provides everyone with the public key base. It also gives each member of the group a different pair of private key base and registration key. The attribute authority uses the public key base in creating a public attribute key. It uses a members registration key to create a private attribute key for that user. Given our scenario, our central authority could be the National Health Insurance. Attribute authorities could be hospitals, clinics, ministry of education, pharmacy,...etc. Notice that a central authority could be an attribute authority too. Bob creates a verification request to Alice. Alice signs

and her signature implicitly proves that Bob's request has been satisfied(See Figure 1).

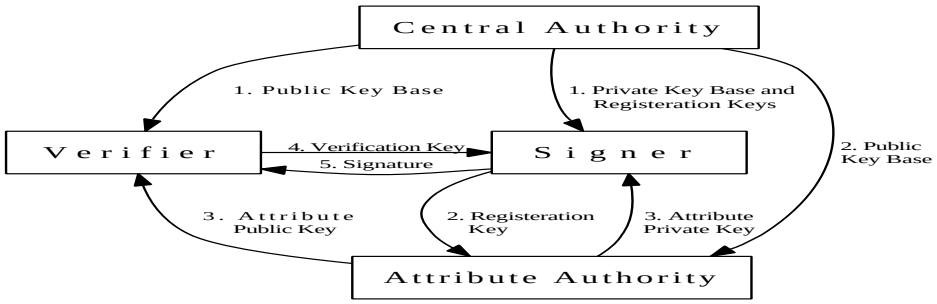

Fig. 1. ABGS

1.1 Related Work

In the early nineties Chaum and Heyst proposed the first group signature scheme [2]. Since then various research has been done on improving the schemes efficiency, security and features.
Many security notions were introduced such as unlinkability, unforgeability, exculpability, coalition-resistance, anonymity, and traceability [3–9]. In [9] the authors tried to unify and simplify all those security notions with defining the two core requirements: full anonymity and full traceability. They proved their definitions to implicitly include all of the other security notions. ABGS is required to be fully traceable and anonymous.
Cryptographers tried adding new features to group signatures. New ideas were born such as Identity based group signatures [10–13], blind group signatures [14, 15], and group signcryption [16, 17]. Our scheme is a continuation of that effort where we propose an attribute based group signature

1.2 Outline

The rest of the paper is organized as follows. We start with giving a precise definition of an ABGS in 2. Formal definitions of the security notions are presented in 2. Section 4 gives an example of an ABGS construction. Required preliminaries for the construction are given in section 3. We finally, conclude in section 5. In the Appendix a comprehensive proof of security is given.

2 Attribute Based Group Signature

The new scheme adopts the idea of an attribute tree from [1] in creating the verification key. In this section we will explain the tree. Then we will define the

algorithms of an ABGS scheme. Finally, we would provide the definitions of the required security notions.
An attribute tree is the structure used in presenting the verifier's request. It is a tree in which each interior node is a threshold gate and the leaves are linked to attributes. A threshold gate represents the fact that the number m of n children branching from the current node need to be satisfied for the parent to be considered satisfied. Satisfaction of a leaf is achieved by owning an attribute. For further explanation, consider the example in Figure 2, which demonstrates an attribute tree for the scenario mentioned earlier:

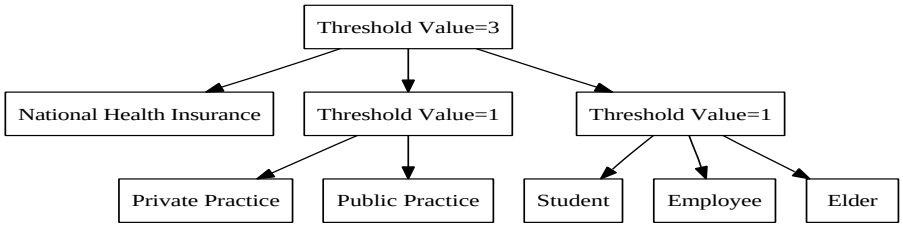

Fig. 2. Attribute Tree

Attribute Based Group Signature Schemes: An Attribute Based Group Signature (ABGS) scheme is specified by eight algorithms: $Setup$, $M.KeyGen$, $A.KeyGen_{pub}$, $A.KeyGen_{pri}$, $V.KeyGen$, $Sign$, $Verify$, and $Open$. As a prerequisite to describing the algorithms we define certain notations.
U is the universal set of attributes with m being the size of it. Γ will be used as a description to our attribute tree. For example, to represent the tree in Figure 2, $\Gamma = \{(3,3)$,National Insurance, $(1,2)$, $(1,3)$, Private Practice, Public Practice, Student, Employee, Elder$\}$, where (m,n) represents a threshold gate m of n. You read Γ in a Top-Down-Left-Right Manner. κ is the number of leaves in Γ. Υ_i is a set describing all private keys a member owns. For example, if Smith is a student and works in the pharmacy. $\Upsilon_{Smith} = \{$Student, Employee$\}$. The size of Υ_i is represented by μ.
After having defined the notations we require, we could describe the algorithms as follows:

- $Setup$: This algorithm is run by the central authority. It takes a security parameter as an input. It outputs two sets of parameters, S_{pri} and S_{pub}. The central authority keeps the system parameters S_{pri} to itself. It publishes S_{pub} so everyone else can use them.
- $M.KeyGen(S_{pri}, n)$: This algorithm is run by the central authority. It takes the inputs S_{pri} and n, where n is the number of members in the group. It generates n private key bases $bsk[i]$ and registration keys A_i. It distributes

them to the members of the group. The $bsk[i]$ is kept private to the user, while the A_i is given to trusted third parties, as shown in $A.KeyGen_{pri}$.

- $A.KeyGen_{pub}(S_{pub})$: This algorithm is run by the attribute authorities. Each authority is responsible of an attribute or more. The authority has for each attribute j, a master key t_j. That key will be used in creating attribute-related keys. Using the master keys the authority creates public keys bpk_j representing attributes it supports. Only the attribute authorities could produce such keys since it requires the knowledge of t_j.
- $A.KeyGen_{pri}(A_i, j)$: This algorithm is run by the attribute authorities. The idea of this algorithm is that member i registers with his key A_i to get a special private key $T_{i,j}$. That key is calculated using the attribute authority's master key t_j. Member i will be using his private key $gsk[i] = \langle bsk[i], A_i, T_{i,1}, ..., T_{i,\mu} \rangle$ to sign. Notice that not all $T_{i,j}$ have been generated by the same attribute authority.
- $V.KeyGen(bpk_1, ..., bpk_\kappa)$: This algorithm is run by the verifier. Verifier creates Γ. The verifier then creates a verification key, gpk. To do so he uses Γ and the attribute public keys bpk_1,...,bpk_κ.
- $Sign(gpk, gsk[i], M, \Gamma)$: This algorithm is run by the signer. The signer uses the verification key gpk, the attribute tree Γ, and his private key $gsk[i]$, to sign on the message M. The output is a signature σ
- $Verify(gpk, M, \sigma)$: This algorithm is run by the verifier. He verifies the signature σ using the verification key gpk. The output is either an acceptance or rejection of the signature.
- $Open(\sigma, M, S_{pri})$: This algorithm is run by the central authority. It traces a signature σ on a message M to the signer i.

Attribute based group signature schemes are considered an extension to group signature schemes. Therefore, it is natural to require the same security notions: full anonymity and full traceability which include all other notions. However, the additional property of an ABGS scheme makes it a necessarily to strengthen the definitions in [9]. The following sections provide adversarial models that define the security notions. These models give the adversary access to a private attribute key oracle and give him the opportunity to decide the universal set of attributes he would like to be challenged upon.

2.1 ABGS Anonymity Definition

We say that an Attribute Based Group Signature Scheme is anonymous under a specific set of attributes, if no polynomially bounded adversary *Adam* has a non-negligible advantage against the challenger *Charles* in the following game:

- **Init:** *Adam* decides the universal set of attributes U in which he would like to be challenged upon.
- **Setup:** *Charles* will play the role of the central authority and attribute authority. He will run the algorithms $Setup$, $M.KeyGen$, and $A.KeyGen_{pub}$. He will produce the systems S_{pub}, and S_{pri}. *Charles* also will generate i

private key bases $bsk[i]$ and i registration keys A_i. He chooses the master keys $\langle t_1,...,t_m\rangle$ randomly from Z_p. Finally, he will generate m public attribute keys $\langle bpk_1,...,bpk_m\rangle$. The tuple $\langle S_{pub}, bpk_1,...,bpk_m\rangle$ is sent to *Adam*.

- **Phase 1:** *Charles* will run three algorithms in this phase: Private Key Oracle, Private Key Base Oracle, and Signature Oracle.
  Adam could query the private key base oracle by sending an index i. He will get from *Charles* a private key base $bsk[i]$ and a registration key A_i. *Adam* could also query the private key oracle by sending a set of attributes Υ_i and the registration key A_i. Υ_i should have at least one attribute j. *Charles* responds back with $T_{i,1},...,T_{i,\mu}$.
  When *Adam* wants to issue a query to the signature oracle, he sends *Charles* an index i, a verification key gpk for an attribute tree Γ, and a message M. *Charles* responds with a signature σ.
- **Challenge:** *Adam* asks to be challenged on a message M, two indices i_0,i_1, and verification key gpk of tree Γ. *Charles* responds back with a signature σ_b, where $b \in \{0,1\}$. The signer could be either i_0 or i_1.
- **Phase 2:** This stage is similar to Phase 1.
- **Output:** *Adam* outputs a guess $\grave{b} \in \{0,1\}$. If $\grave{b} = b$, *Adam* wins the game.

If we prove that the previous game is anonymous, we implicitly prove that it is unlinkable. The reason behind such a claim is that the adversary is given the capability of requesting a challenge which he already queried in the signature oracle in phase 1. If the scheme is linkable then the adversary could easily win this game by linking the challenge to a previous queried signature.

2.2 ABGS Traceability Definition

We say that an Attribute Based Group Signature Scheme is traceable if no polynomially bounded adversary *Adam* has a non-negligible advantage against the challenger in the following game:

- **Init:** *Adam* decides the universal set of attributes U he would like to be challenged upon.
- **Setup:** *Charles* will play the role of the central authority and attribute authority. He will run the algorithms $Setup$, $M.KeyGen$, and $A.KeyGen_{pub}$. He will produce the systems S_{pub}, and S_{pri}. *Charles* also will generate i private key bases $bsk[i]$ and i registration keys A_i. Finally, he will generate m public attribute keys $\langle bpk_1,...,bpk_m\rangle$. The tuple $\langle S_{pub},bpk_1,...,bpk_m\rangle$ is sent to *Adam*.
- **Running Queries:** *Charles* runs three queries: Private Key Base, Private key, and finally Signature queries.
  Adam could request for a private key base. He sends *Charles* an index i. *Charles* replies back with the registration key A_i and private key base $bsk[i]$.
  Adam might want to issue a private key query. He sends a registration key A_i and a set Υ_i. The set Υ_i has one or more attributes. *Charles* responds with $T_{i,1},...,T_{i,\mu}$.

Finally, *Adam* could run $V.KeyGen$ to obtain a Γ of his choice and a gpk. He could send to *Adam* the output with a message M, and an index i, requesting a signature. $Charles$ will reply with a signature σ.

– **Challenge:** *Adam* asks to be challenged on a message M. $Charles$ generates a Γ and the corresponding gpk. He sends them to *Adam*. *Adam* replies with a forged signature σ. $Charles$ verifies the signature. If it turns up to be a valid signature, he tries tracing it to a signer. If it traces to a signer in which *Adam* did not query before or if it traces to a nonmember then *Adam* wins the game.

Full traceability includes unforgeability. One can reduce the challenge to be producing a valid pair of message and signature, where the message was not queried in phase 1. The new adversarial model is the definition of unforgeability. Therefore, full traceability implicitly proves unforgeability.
In the following section we will give a construction of an ABGS scheme. That scheme is proven to be fully traceable and anonymous in the appendix.

3 Preliminaries

In this section we will explain some of the preliminaries that are used in constructing the ABGS scheme and proving it secure. We will define q-Strong Diffie-Hellman which is used in building the scheme and proving it's traceability. Then we state the Decision Linear Diffie-Hellman Assumption used in constructing the scheme and proving it anonymous. Finally, we define Bilinear Maps which we base our scheme on to enable handling the attribute tree.

Definition 1. (q-Strong Diffie-Hellman Problem [18])
Let G_1, G_2 be cyclic groups of prime order p, with a computable isomorphism ψ or possibly $G_1 = G_2$. Assuming the generators $g_1 \in G_1$, and $g_2 \in G_2$. The q-SDH problem in (G_1, G_2) is defined as follows: given a $(q+2)$ tuple $(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}, ..., g_2^{\gamma^q})$ as an input, output what is called a SDH pair $(g_1^{1/(\gamma+x)}, x)$ where $x \in Z_p^*$. An algorithm A has an advantage ε in solving q-SDH in (G_1, G_2) if: $Pr[A(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}, ..., g_2^{\gamma^q}) = (g_1^{1/(\gamma+x)}, x)] \geq \varepsilon$, where the probability is over a random choice of a generator g_2 (with $g_1 \leftarrow \psi(g_2)$), of $\gamma \in Z_p^*$ and of random bits of A.

This problem is considered hard to solve in polynomial time and ε should be negligible [18].

Definition 2. (Decision Linear Problem in G_1 [19])
Let G_1 be a group of prime order p and u, v, h be generators in that group. Given $u, v, h, u^a, v^b, h^c \in G_1$ as an input,it is hard to decide whether or not $a + b = c$.

Definition 3. (Bilinear Maps [20]):
Let G_1,G_2 and G_T be three groups of order p for some large prime p. A bilinear map $\hat{e} : G_1 \times G_2 \rightarrow G_T$ must satisfy the following properties:

– *Bilinear: We say that a map* $\hat{e} : G_1 \times G_2 \rightarrow G_T$ *is bilinear if* $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ *for any generator* $g_1 \in G_1$, $g_2 \in G_2$ *and any* $a, b \in Z_p$.
– *Non-degenerate: The map does not send all pairs in* $G_1 \times G_2$ *to the identity in* G_T.
– *Computable: There is an efficient algorithm to compute* $\hat{e}(g_1, g_2)$ *for any* $g_1 \in G_1$ *and* $g_2 \in G_2$.

A bilinear map satisfying the three properties above is said to be an admissible bilinear map.

4 Construction of an ABGS

In this section we will construct an ABGS scheme based on Boneh and Shacham [21]. Notice that the *Open* algorithm has been replaced by *Revoke*. In case of a dispute, the central authority creates a fake revocation list and adds all members in the group to it. The authority verifies the signature and checks whether the signer is revoked. If the signature is not forged the signer should be a member of the group. Therefore, running the revocation algorithm on the fake list should output true. Otherwise, the signature is forged.

– *Setup*: Consider a bilinear pair (G_1, G_2) with a computable isomorphism ψ. Suppose that SDH assumption holds on (G_1, G_2) and the decision linear assumption holds on G_2. Define the bilinear map $\hat{e} : G_1 \times G_2 \rightarrow G_3$. All three groups G_1, G_2, G_3 are multiplicative and of a prime order p. Select a hash function $H : \{0,1\}^* \rightarrow Z_p$. Select a hash function H_0 with respected range G_2^2. Select a generator $g_2 \in G_2$ at random and then set $g_1 \leftarrow \psi(g_2)$. Select a random γ from Z_p and set $w = g_2^{\gamma}$. Let $S_{pub} = \langle G_1, G_2, G_T, \hat{e}, H, H_0, g_1, g_2, w \rangle$ and $S_{pri} = \gamma$.

– $M.KeyGen(S_{pri}, n)$: Using γ generate for each user $i, 1 \leq i \leq n$, a private key base $bsk[i] = \langle A_i, x_i \rangle$. All $bsk[i]$ should be SDH pairs, where x_i is chosen randomly from Z_p^* and $A_i = g_1^{1/(\gamma + x_i)} \in G_1$.

– $A.KeyGen_{pub}(S_{pub})$: The public key for attribute j is $bpk_j = g_2^{\gamma/t_j} = w^{1/t_j}$, where t_j is chosen randomly from Z_p

– $A.KeyGen_{pri}(A_i, j, R)$: User i wants to register attribute j. It contacts the attribute authority in charge, which checks the revocation list R. R contains all registeration keys of users who have been revoked. If the member i is not on the list, authority calculates $T_{i,j} = A_i^{t_j}$ and gives to user i. The private key for a user i will be the tuple $gsk[i] = \langle A_i, x_i, T_{i,1},...,T_{i,\mu} \rangle$.

– $V.KeyGen(bpk_1, ..., bpk_\kappa)$: Verifier chooses a Γ. Then he chooses a polynomial q_{node} of degree $d_{node} = k_{node} - 1$ for each node in the tree. k_{node} is the threshold gate value of every node. In other words k_{node} children need to be satisfied in order to consider the parent satisfied. Choosing the polynomials

is done in top-down manner. Starting from the root $q_{root}(0) = s_{rnd}$, where s_{rnd} is chosen randomly from Z_p and other points in the polynomial will be random. The other nodes we set $q_{node}(0) = q_{parent}(index(node))$ and choose the rest of the points of the polynomial randomly. Once all polynomials have been decided the verification key for a certain structure will be $gpk = \langle g_1, g_2, w, D_1,...,D_\kappa \rangle$ where $D_j = bsk[i]^{q_j(0)}$.

– $Sign(gpk, gsk[i], M, \Gamma)$: The signer picks randomly an r from Z_p and obtains $(\hat{u}, \hat{v})$ from $H_0(gpk, M, r)$. Then compute their images $u \leftarrow \psi(\hat{u})$ and $v \leftarrow \psi(\hat{v})$. User i chooses α and β randomly from Z_p.
Then he computes $C_1 = u^\alpha$, $C_2 = A_i v^\alpha$, $C_3 = \hat{e}(v^\alpha, w)^\beta$ and $C_4 = w^\beta$.

We need to define a recursive algorithm $Sign_{Node}$. If the node we are currently on is a leaf in the tree the algorithm returns the following:

$$Sign_{Node}(leaf) = \begin{cases} \text{If } (j \in \Gamma); \text{ return } \hat{e}(T_{i,j}^\beta, D_j) = \hat{e}(A_i, w)^{\beta q_j(0)} \\ \text{Otherwise return } \perp \end{cases}$$

For a node ρ which is not a leaf the algorithm proceeds as follows: For all children z of the node ρ it calls $Sign_{Node}$ and stores output as F_z. Let $\hat{S}_\rho$ be an arbitrary k_ρ sized set of children nodes z such that $F_z \neq \perp$ and if no such set exist return $\perp$.
Otherwise let $\Delta_{\hat{S}_\rho, index(z)} = \Pi_{\iota \in \{index(z): z \in \hat{S}_\rho - index(z)\}}(-\iota/(index(z) - \iota))$ and compute
$F_\rho = \Pi_{z \in \hat{S}_\rho} F_z^{\Delta_{\hat{S}_\rho, index(z)}} = \Pi_{z \in \hat{S}_\rho} \hat{e}(A_i^\beta, w)^{q_z(0).\Delta_{\hat{S}_\rho, index(z)}} = \hat{e}(A_i^\beta, w)^{q_\rho(0)}$

To create the signature calculate F_{root}. If the tree is satisfied then $F_{root} = \hat{e}(A_i^\beta, w)^{s_{rnd}}$.

Let $\delta = x_i \alpha$. Pick randomly r_α, r_x, and r_δ from Z_p.

Let $R_1 = u^{r_\alpha}$, $R_2 = \hat{e}(C_2, g_2)^{r_x} \hat{e}(v, w)^{-r_\alpha} \hat{e}(v, g_2)^{-r_\delta}$ and $R_3 = C_1^{r_x} u^{-r_\delta}$.

Compute $c = H(gpk, M, r, C_1, C_2, C_3, C_4, R_1, R_2, R_3)$, $s_\alpha = r_\alpha + c\alpha$, $s_x = r_x + cx_i$ and $s_\delta = r_\delta + c\delta$.

Finally, output the signature $\sigma = (r, C_1, C_2, C_3, C_4, c, s_\alpha, s_x, s_\delta, F_{root})$.

– $Verify(gpk, M, \sigma, R)$: The verifier could calculate $(\hat{u}, \hat{v}) = H_0(gpk, M, r)$ then $u \leftarrow \psi(\hat{u})$, and $v \leftarrow \psi(\hat{v})$. Verifier derives R_1,R_2 and R_3 by calculating

$\bar{R}_1 = u^{s_\alpha} / C_1^c$,

$\bar{R}_2 = \hat{e}(C_2, g_2)^{s_x} \hat{e}(v, w)^{-s_\alpha} \hat{e}(v, g_2)^{-s_\delta} . (\frac{\hat{e}(C_2, w)}{\hat{e}(g_1, g_2)})^c$.

$\bar{R}_3 = C_1^{s_x} u^{-s_\delta}$

If $c \neq H(gpk, M, r, C_1, C_2, C_3, C_4, \bar{R}_1, \bar{R}_2, \bar{R}_3)$ reject the signature. Otherwise verifier needs to check whether the signer is in the revocation list and whether he satisfies the attribute tree Γ.

If $F_{root}^{1/s_{rnd}}.C_3 = \hat{e}(C_2, C_4)$ that implies that user i has satisfied the attribute tree Γ. The revocation list R has all values of $A_{revoked}$ where $A_{revoked}$ is the registration key of revoked users. If for all revoked users $\hat{e}(C_2/A_{revoked}, \hat{u}) = \hat{e}(C_1, \hat{v})$ does not hold then user i still is a valid user.

– *Revoke*(i): Revoke is about building a revocation list R. The algorithm gets an index of a user i and adds A_i to the list.

Theorem 1. *The ABGS scheme is correct.*

Proof. To prove the scheme is correct we need to show that $R_1 = \bar{R}_1$, $R_2 = \bar{R}_2$, and $R_3 = \bar{R}_3$. That way we prove $H(gpk, M, r, C_1, C_2, C_3, C_4, R_1, R_2, R_3) = H(gpk, M, r, C_1, C_2, C_3, C_4, \bar{R}_1, \bar{R}_2, \bar{R}_3)$. The signature should be correctly verified unless the user is revoked. We start our proof with showing that the three equations hold:

$$\bar{R}_1 = u^{s_\alpha}/C_1^c = u^{r_\alpha + c\alpha}/u^{c\alpha} = u^{r_\alpha} = R_1$$

$$\bar{R}_3 = C_1^{s_x}.u^{-s_\delta} = (u^\alpha)^{r_x + cx_i}.u^{-(r_\delta + c\delta)} = u^{\alpha r_x + \alpha c x_i - r_\delta - c\delta} = C_1^{r_x}.u^{-r_\delta} = R_3$$

$$\bar{R}_2 = \hat{e}(C_2, g_2)^{s_x}\hat{e}(v, w)^{-s_\alpha}\hat{e}(v, g_2)^{-s_\delta}.(\frac{\hat{e}(C_2, w)}{\hat{e}(g_1, g_2)})^c$$

$$= (\hat{e}(C_2, g_2)^{r_x}.\hat{e}(v, w)^{-r_\alpha}.\hat{e}(v, g_2)^{-r_\delta}).(\hat{e}(C_2, g_2)^{x_i}.\hat{e}(v, w)^{-\alpha}.\hat{e}(v, g_2)^{-\alpha x_i}.\frac{\hat{e}(C_2, w)}{\hat{e}(g_1, g_2)})^c$$

$$= R_2((\hat{e}(C_2 v^{-\alpha}, w g_2^{x_i}))/\hat{e}(g_1, g_2))^c = R_2(\hat{e}(A_i, w g_2^{x_i})/\hat{e}(g_1, g_2))^c = R_2$$

In the verifying algorithm we check revoked users before accepting a signature. In the signature we have $C_1 = \psi(\hat{u})^\alpha$ and $C_2 = A_i\psi(\hat{v})^\alpha$ for some random α. We reject a signature when $(\hat{u}, \hat{v}, C_1, C_2/A_{revoked})$ is a co-Diffie Hellman tuple. We also check whether signer satisfies the tree Γ by checking the equality of $F_{root}^{1/s_{rnd}}.C_3 = \hat{e}(C_2, C_4)$. The correctness of that could be proved by starting with the fact that $F_{root} = \hat{e}(A_i^\beta, w)^{s_{rnd}}$ when the tree is satisfied. This implies $F_{root}^{1/s_{rnd}}.C_3 = \hat{e}(A_i^\beta, w).\hat{e}(v^\alpha, w)^\beta = \hat{e}(A_i v^\alpha, w^\beta) = \hat{e}(C_2, C_4)$.

In the appendix we use the adversial models in section 2 to prove the following theorems:

Theorem 2. *If SDH is hard on groups (G_1, G_2) then the selective model of the Attribute Based Group Signature Scheme is said to be traceable under the random oracle.*

Theorem 3. *If the decision linear assumption holds in group G_2 then the Attribute Based Group Signature Scheme is said to be anonymous under the random oracle.*

5 Conclusion

In this paper we define an ABGS with multi-authorities and its security notions. We construct an example and prove that to be fully anonymous and fully traceable. Our scheme enables authority segregation which implies having each authority responsible of giving out certain attributes. Moreover, our scheme is dynamic in the sense that you could add and remove members of the group. Furthermore, the ABGS scheme is efficient since the size of keys and signature is independent from the number of members of the group and number of attributes involved. However, the drawback of such a scheme is the verification key for two reasons. First of all, the key size depends on the number of attributes a verifier requests. This could be unpreventable since the verifier has to list down the attributes he requires. The second disadvantage is that any eavesdropper could guess the attributes the verifier needs. A possible solution using searchable encryption techniques to preserve confidentiality of the verifier.

References

1. V. Goyal, O. Pandeyy, A. Sahaiz, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89 – 98, 2006.
2. D. Chaum and V. Heyst. Group signatures. In *Proceedings of Eurocrypt 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.
3. X. Boyen and B. Waters. Compact group signatures without random oracles. In *In Proceedings of EUROCRYPT'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer-Verlag, 2006.
4. Y. Komano, K. Ohta, A. Shimbo, and S. Kawamura. Toward the fair anonymous signatures: Deniable ring signatures. In *Topics in Cryptology - CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 174–191. Springer-Verlag, 2006.
5. D. X. Song. Practical forward secure group signature schemes. In *In: Proc. of the 8th ACM Conference on Computer and Communications Security*, volume 2229, pages 225–234, 2001.
6. I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication. In *Proceedings ASIACRYPT'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, 2004.
7. J. Camenisch. Efficient and generalized group signatures. In *Proceedings of Eurocrypt 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479. Springer-Verlag, 1997.
8. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Proceedings of Crypto'00*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, 2000.

9. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Proceedings EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer-Verlag, 2003.
10. S. Park, S. Kim, and D. Won. Id-based group signature. In *Electronics Letters*, pages 1616–1617, 1997.
11. S. Popescu. An efficient id-based group signature scheme. In *Studia Univ. Babes-Bolyai Informatica*, http://www.cs.ubbcluj.ro/ studia-i/2002-2/, pages 29–36, 2002.
12. V. Wei, T. Yuen, and F. Zhang. Group signature where group manager members and open authority are identity-based. In *Proceedings of Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 468–480. Springer-Verlag, 2005.
13. M. Au, J. Liu, T. Yuen, and D. Wong. Id-based ring signature scheme secure in the standard model. In *Proceedings of Advances in Information and Computer Security*, volume 4266 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2006.
14. A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Proceedings of Financial Cryptography'98*, volume 1465 of *Lecture Notes in Computer Science*, pages 184–197. Springer-Verlag, 1998.
15. J. Herranz and F. Laguillaumie. Blind ring signatures secure under the chosen-target-cdh assumption. In *In Proceedings of Information Security*, volume 4176 of *Lecture Notes in Computer Science*, pages 117–130. Springer-Verlag, 2006.
16. G. Wang, R. Deng, D. Kwak, and S. Moon. Security analysis of two signcryption schemes. In *Information Security*, volume 3225 of *Lecture Notes in Computer Science*, pages 123–133. Springer-Verlag, 2004.
17. D. Kwak, S. Moonb, G. Wangc, and R. Dengd. A secure extension of the kwak'moon group signcryption scheme. In *Computers and Security*, volume 25, pages 435–444. ELSEVIER, 2006.
18. D. Boneh and X. Boyen. Short signatures without random oracles. In *Proceedings of Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 382–400. Springer-Verlag, 2004.
19. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proceedings of Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41 – 55. Springer-Verlag, 2004.
20. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
21. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *In proceedings of the 11'th ACM conference on Computer and Communications Security*, pages 168–177, 2004.
22. Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In *Advances in Cryptology - CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 223–242, 1998.
23. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. In *Journal of Cryptography*, volume 13 of *Number 3*, pages 361–396. Springer-Verlag, 2000.

A Extra Preliminaries

A.1 Forking Lemma

Pointcheval and Stern [23], developed the Forking Lemma as a method to prove certain security notions of a digital signature scheme. We will be using it in proving our scheme to be traceable(See Appendix B). Assume a signature scheme produces the triple $\langle \sigma_1, h, \sigma_2 \rangle$ where σ_1 takes its values randomly from a set. h is the result of hashing the message M together with σ_1. σ_2 depends only on (σ_1, h, M). The Forking Lemma is as follows [23]:

Theorem 4. (The Forking Lemma)
Let A be a Probabilistic Polynomial Time Turing machine, given only the public data as input. If A can find, with non-negligible probability, a valid signature $(M, \sigma_1, h, \sigma_2)$ then, with non-negligible probability, a replay of this machine, with the same random tape but a different oracle, outputs new valid signatures $(M, \sigma_1, h, \sigma_2)$ and $(M, \sigma_1, \grave{h}, \grave{\sigma_2})$ such that $h \neq \grave{h}$.

A.2 Heavy Row Lemma

In this section we define a Boolean Matrix. We define a Heavy Row in that matrix [22]. Both definitions are used in the Heavy Row Lemma [22] which will be used in proving traceability of our scheme together with the Forking lemma(See Section B).

Definition 4. (Boolean Matrix of Random Tapes) *Consider a hypothetical matrix M whose rows consists of all possible random choices of an adversary and the columns consist of all possible random choices of a challenger. Let each entry be either $\perp$ when the adversary fails or $\top$ if the adversary manages to win the game.*

Definition 5. (Heavy Row) *A row in M is called heavy if the fraction of $\top$ along the row is less than $\varepsilon/2$ where ε is the advantage of the adversary succeeding in attack.*

Lemma 1. (Heavy Row Lemma) *Let M be a boolean matrix, given any entry that equals $\top$, the probability that it lies in a heavy row is at least 1/2.*

A.3 The Strong Diffie-Hellman Assumption

In addition to the q-SDH problem mentioned earlier in section 3, we would need the Boneh and Boyen SDH Equivalence theorem to prove traceability of the scheme.

Theorem 5. (Boneh-Boyen SDH Equivalence [18])
Given a q-SDH instance $(\grave{g}_1, \grave{g}_2, \grave{g}_2^{\gamma}, \grave{g}_2^{\gamma^2}, ..., \grave{g}_2^{\gamma^q})$, and then applying the Boneh and Boyen's Lemma found in [18] we can obtain $g_1 \in G_1, g_2 \in G_2, w = g_2^{\gamma}$ and $(q-1)$ SDH pairs (A_i, x_i) (such that $e(A_i, wg_2^{x_i}) = e(g_1, g_2)$) for each i. Any SDH pair besides these $(q-1)$ ones can be transformed into a solution to the original q-SDH instance.

A.4 Linear Encryption

In this section we will define an encryption scheme which depends on the difficulty of the Decision Linear Diffie-Hellman Assumption. The IND-CPA security of the following encryption scheme would be used to prove our scheme anonymous.

Definition 6. (A Linear Encryption Scheme [19])
In a Linear Encryption scheme a user's public key is $u, v, h \in G_1$. *The private key is the exponents* $\xi_1, \xi_2 \in Z_p$ *such that* $u^{\xi_1} = v^{\xi_2} = h$. *To encrypt a message* M *choose random elements* $\alpha, \beta \in Z_p$ *and output the triple* $\langle C_1, C_2, C_3 \rangle = \langle u^{\alpha}, v^{\beta}, Mh^{\alpha+\beta} \rangle$. *To decrypt compute* $C_3/(C_1^{\xi_1} C_2^{\xi_2})$.

LE has been proven to be IND-CPA secure under the Decision Linear Problem.

B Traceability

Theorem 6. *If SDH is hard on groups* (G_1, G_2) *then the selective model of the Attribute Based Group Signature Scheme is said to be traceable under the random oracle.*

Proof. In order to prove that we need three steps. Defining a security model for proving traceability, introducing two types of signature forger, and then we show that the existence of such forgers implies that SDH is easy. Suppose we are given an adversary *Adam* that breaks the traceability of the signature scheme. The security model will be defined as an interacting framework between the *Charles* and *Adam* as follows:

- **Init:** *Adam* decides the universal set of attributes U it would like to be challenged upon.

- **Setup:** *Charles* is given a bilinear pair (G_1, G_2) with respective generators g_1, and g_2. It is also given a value $w = g_2^{\gamma}$ and n private key bases $bsk[i] = \langle A_i, x_i \rangle$ for an $1 \leq i \leq n$. Some of those pairs have $x_i = \star$ which implies that x_i corresponding to A_i is not known; Other pairs are valid SDH pairs(Theorem 1). *Charles* chooses randomly t_1,...,t_m from Z_p, where m is the size of the set U. He then runs the $A.KeyGen_{pub}$ and gives *Adam* the tuple $\langle bpk_1,...,bpk_m \rangle$ where $bpk_j = w^{1/t_j}$. *Charles* sends $S_{pub} = \langle G_1, G_2, G_T, \hat{e}, g_1, g_2, w \rangle$ and keeps $S_{pri} = \gamma$. Hash functions H_0 and H are presented as Random Oracles later on.

- **Private Key Base Queries:** *Adam* could ask for certain private keys by sending the *Charles* an index i. If *Adam* queries an index where $x_i = \star$ abort the game and declare failure otherwise respond with sending $\langle A_i, x_i \rangle$.

– **Hash Queries:** When the *Adam* asks *Charles* for the hash of $(gpk, M, r, C_1, C_2, C_3, C_4, R_1, R_2, R_3)$, *Charles* responds with a random element in G_1 and saves the answer just incase the same query is requested again. That represents the hash function H. When the *Adam* asks *Charles* for the hash of (gpk, M, r), *Charles* responds with two random elements in G_2 and saves the answer.

– **Revocation Queries:** *Adam* picks a random index i to revoke. A_i is added to the list R. The size of the list should be reasonable enough to run the rest of the oracles in the next steps.

– **Signature Queries:** *Adam* runs the $V.KeyGen$ for an attribute tree Γ of his choice. He creates $gpk = \langle g_1, g_2, w, D_1, ..., D_\kappa \rangle$ where $D_j = bpk_j^{q_j(0)}$.
Adam asks for a signature on a message M by the member i. If $x_i \neq \star$ the *Charles* follows the same signing procedure done in section[4].

If $x_i = \star$, *Charles* simulates a signature. He picks a random r from Z_p. He then gets $(\hat{u}, \hat{v}) = H_0(gpk, M, r)$. Sets $u \leftarrow \psi(\hat{u})$ and $v \leftarrow \psi(\hat{v})$. Picks a random α and β from Z_p.

Charles then calculates $C_1 = u^\alpha$, $C_2 = A_i v^\alpha$, $C_3 = \hat{e}(v^\alpha, w)^\beta$ and $C_4 = w^\beta$. It randomly picks c, s_δ, s_x, and s_α from Z_p. Calculates $R_1 = u^{s_\alpha}/C_1^c$, $R_3 = C_1^{s_x} u^{-s_\delta}$ and $R_2 = \hat{e}(C_2, g_2)^{s_x} \hat{e}(v, w)^{-s_\delta} \hat{e}(v, g_2)^{-s_\delta} . (\hat{e}(C_2, w)/\hat{e}(g_1, g_2))^c$.
Charles adds c to the list of the hash oracle H to maintain consistency.

Charles takes every D_j in gpk and calculates $\hat{e}(A_i^{t_j}, D_j)^\beta$.
Charles could now calculate F_{root} using the elements it calculated and Γ.

Charles returns the signature $\sigma = (r, C_1, C_2, C_3, C_4, c, s_\alpha, s_x, s_\delta, F_{root})$ to *Adam*.

– **Private Key Queries:** *Adam* issues a query for a private key by sending *Charles* an A_i, and a set Υ_i. Υ_i should have at least one element j. If $x_i \neq \star$, *Charles* responds back with $gsk[i] =< A_i, x_i, A_i^{t_1}, ..., A_i^{t_\mu} >$. Otherwise, *Charles* fails and terminates the game. In other words, *Charles* runs the $A.KeyGen_{pri}$ algorithm.

– **Challenge:** *Adam* asks to be challenged and sends *Charles* a message M. *Charles* responds back with a gpk for a certain Γ. If *Adam* is successful, he will output a forged signature $\sigma = (r, C_1, C_2, C_3, C_4, c, s_\alpha, s_x, s_\delta, F_{root})$ on a message M. C_1 and C_2 should not have any of the revocation list elements $A_{revoked}$ encoded in them. Let A_i^* be the value used in signing the forged signature. For $i = 1, ..., n$, *Charles* checks whether $\hat{e}(C_2/A_i, \hat{u}) = \hat{e}(C_1, \hat{v})$. If the equality holds then that implies that $A_i^* = A_i$. In that case check if $s_{i^*} = \star$ to output σ or otherwise declare failure. If the for loop goes through all the A_is and there was no equality output σ.

From this model of security there are two types of forgery. Type-I outputs a signature that could be traced to some identity which is not part of $\{A_1,...,A_n\}$. Type-II has $A_i^* = A_i$ where $1 \leq i \leq n$ but *Adam* did not do a private key query on i. We should prove that both forgeries are hard.

Type-I: If we consider Theorem 5 for a $(n+1)$SDH, we could obtain g_1,g_2 and w. We could also use the i pairs (A_i, x_i) to calculate the private key bases $\langle A_i, x_i \rangle$. We use these values in interacting with *Adam*. *Adam*'s success leads to forgery of Type-I and the probability is ε.

Type-II: Using the same Theorem 5 but for a (i)SDH this time, we could obtain g_1, g_2 and w. Then we could also use the $i-1$ pairs (A_i, x_i) to calculate the private key bases $\langle A_i, x_i \rangle$. In a random index i^*, we could choose the missing pair randomly where $A_{i^*} \in G_1$ and set $x_{i^*} = \star$. *Adam* in the security model will fail if he queries the private key or private key bases oracle in index i^*. Other private key queries will succeed. In the signature oracle and because the hashing oracle is used it will be hard to distinguish between signatures with a SDH pair and ones without. As for the output algorithm the probability of tracing to a forged signature that leads to index i^* is equal to ε/n.

Now we need to prove that the existence of any of the two forgeries contradicts with SDH assumption. For that we use the Forking Lemma (See Theorem[4]). Let *Adam* be a forger of any type in which the security model succeeds with probability $\grave{\varepsilon}$. A signature will be represented as $\langle M,\sigma_0,c,\sigma_1,\sigma_2 \rangle$. M is the signed message. $\sigma_0 = \langle r,gpk,C_1,C_2,C_3,C_4,R_1,R_2,R_3 \rangle$. c is the value derived from hashing σ_0. $\sigma_1 = \langle s_\alpha, s_x, s_\delta \rangle$ which are values used to calculate the missing inputs for the hash function. Finally $\sigma_2 = F_{root}$ the values that depend on the set of attributes in each signature oracle.
We require *Adam* to query H_0 before H to ensure that by rewinding the game we could change values of $H(M, r, ..)$, while values of $H_0(M, r)$ should remain the same. Therefore the arguments u, v used in H remain unchanged too.
One simulated run of the adversary is described by the randomness string ω(used by *Adam* and *Charles*),by the vector ℓ_0 of responses made by H_0 and by the vector ℓ of responses made by H. Let S be the set of tuple (ω, ℓ_0, ℓ) where *Adam* successfully forges the signature $(M, \sigma_0, c, \sigma_1, \sigma_2)$ and he queried H on (M, σ_0). Let $Ind(\omega, \ell_0, \ell)$ be the index of ℓ at which *Adam* queried (M, σ_0). Let $\nu = Pr[S] = \grave{\varepsilon} - 1/p$ where $1/p$ term represents the possibility that *Adam* guessed the hash of (M, σ_0) without quering it. For each χ, $1 \leq \chi \leq q_H$, let S_χ be a set of the tuple $(\omega, \ell_0, \ell)$where $Ind(\omega, \ell_0, \ell) = \chi$. Let Φ be the set of indices χ where $Pr[S_\chi|S] \geq 1/2q_H$ causing $Pr[Ind(\omega, \ell_0, \ell) \in \Phi|S] \geq 1/2$.
Let $\ell|_a^b$ be the restriction of ℓ to its elements at indices $a, a+1, ..., b$. For each $\chi \in \Phi$ consider the heavy row lemma (See Section[A.2]) with a matrix with rows indexed with$(\omega, \ell_0, \ell|_1^{\chi-1})$ and columns $(\ell|_\chi^{q_H})$. If (x, y) is a cell, then $Pr[(x, y) \in S_\chi] \geq \nu/2q_H$. Let the heavy rows Ω_χ be the rows such that $\forall (x, y) \in \Omega_\chi : Pr_{\grave{y}}[(x, \grave{y}) \in S_\chi] \geq \nu/(4q_H)$. By the heavy row lemma $Pr[\Omega_\chi|S_\chi] \geq 1/2$

which leads to $Pr[\exists \chi \in \Phi : \Omega_\chi \cap S_\chi | S] \geq 1/4$.
Therefore *Adam*'s probability in forging a signature is about $\nu/4$. That signature derives from the heavy row $(x, y) \in \Omega_\chi$ for some $\chi \in \Phi$, hence execution (ω, ℓ_0, ℓ) such that the $Pr_{\grave{\ell}}[(\omega, \ell_0, \grave{\ell}) \in S_j | \grave{\ell}|_1^{j-1} = \ell|_1^{j-1}] \geq \nu/(4q_H)$. In other words if we have another simulated run of the adversary with $\grave{\ell}$ that differs from ℓ starting the jth query *Adam* will forge another signature $\langle M, \sigma_0, \grave{c}, \grave{\sigma_1}, \sigma_2 \rangle$ with the probability $\nu/(4q_H)$.
Now we show how we could extract from $\langle \sigma_0, c, \sigma_1, \sigma_2 \rangle$ and $\langle \sigma_0, \grave{c}, \grave{\sigma_1}, \sigma_2 \rangle$ a new SDH tuple. Let $\Delta c = c - \grave{c}$, $\Delta s_\alpha = s_\alpha - \grave{s}_\alpha$, and similarly for Δs_x, and Δs_δ.
Divide two instances of the equations used previously in proving correctness of the scheme (See section 4). One instance is with $\grave{c}$ and the other is with c to get the following:

- Dividing $C_1^c / C_1^{\grave{c}} = u^{s_\alpha} / u^{\grave{s}_\alpha}$ we get $u^{\tilde{\alpha}} = C_1$; where $\tilde{\alpha} = \Delta s_\alpha / \Delta c$
- Dividing $C_1^{s_x} / C_1^{\grave{s}_x} = u^{s_\delta} / u^{\grave{s}_\delta}$ will lead to $\Delta s_\delta = \tilde{\alpha} \Delta s_x$
- Dividing $(\hat{e}(g_1, g_2) / \hat{e}(C_2, w))^{\Delta c}$ will lead to $\hat{e}(C_2, g_2)^{\Delta s_x} \hat{e}(v, w)^{-\Delta s_\alpha} \hat{e}(v, g_2)^{-\tilde{\alpha} \Delta s_x}$

Letting $\dot{x} = \Delta s_x / \Delta c$ we get $\hat{e}(g_1, g_2) / \hat{e}(C_2, w) = \hat{e}(C_2, g_2)^{\dot{x}} \hat{e}(v, w)^{-\dot{\alpha}} \hat{e}(v, g_2)^{-\dot{x}\dot{\alpha}}$ this could be rearranged as $\hat{e}(g_1, g_2) = \hat{e}(C_2 v^{-\dot{\alpha}}, w g_2^{\dot{x}})$. Let $\grave{A} = C_2 v^{-\dot{\alpha}}$ we get $\hat{e}(\grave{A}, w g_2^{\dot{x}}) = \hat{e}(g_1, g_2)$. Hence we obtain a new SDH pair $(\grave{A}, \dot{x})$ breaking Boneh and Boyens Lemma(See Section[5]). Now putting things together we get the following theorems:

Theorem 7. *We could solve an instance of* $(n + 1)$ *SDH with a probability* $(\varepsilon - 1/p)^2 / 16 q_H$ *using a Type-I forger Adam*

Theorem 8. *We could solve an instance of* n *SDH with a probability* $(\varepsilon/n - 1/p)^2 / 16 q_H$ *using a Type-II forger Adam*

C ABGS Scheme Anonymity

Theorem 9. *If the decision linear assumption holds in group* G_2 *then the Attribute Based Group Signature Scheme is said to be anonymous under the random oracle.*

Assuming *Adam* is an adversary that breaks the anonymity of the ABGS scheme. We will prove that there is an adversary *Eve* that solves the decisional linear assumption using *Adam*'s talent. Note that *Eve* in this game plays a challenger's role when it comes to interacting with *Adam* and an adversary's role when she interacts with *Charles*. So the game is demonstrated below:

- **Init:** *Adam* decides the universal set of attributes U, in which he would like to be challenged upon and gives it to *Eve*.

– **Setup:** *Charles* gives *Eve* the tuple $\langle u_0,u_1,u_2,h_0 = u_0^a,h_1 = u_1^b,Z\rangle$ where $u_0,u_1,u_2 \in G_2$ and $a, b \in Z_p$. Z is either random or $Z = u_2^{a+b}$. *Eve* should decide which Z she was given. Recall that g_1, g_2 are in G_1 and G_2 respectively. *Eve* chooses a random γ from Z_p. *Eve* also chooses $t_1,...,t_m$ for attributes in U, where m is the size of U. *Eve* assigns $w = g_2^{\gamma}$. She creates the $n-2$ private key bases $bsk[i] = \langle A_i, x_i\rangle$ as in section[4]. She will then choose a random $W \in G_2$. The missing private key bases of user i_0 and i_1 are chosen randomly. They are defined as $A_{i_0} = ZW/u_2^a$ and $A_{i_1} = Wu_2^b$ for some x_{i_0},x_{i_1}. Notice that if $Z = u_2^{a+b}$ then $A_{i_0} = A_{i_1}$. *Eve* does not know the values of either $bsk[i_0]$ or $bsk[i_1]$. We will show later in our security model how she could still interact with *Adam* pretending she does know them. *Eve* gives *Adam* the $S_{pub} = \langle G_1,G_2, G_T, \hat{e}, g_1, g_2, w\rangle$ keeping $S_{pri} = \gamma$. It also runs the $A.KeyGen_{pub}$ algorithm and gives *Adam* the values $\langle bpk_1,...,bpk_m\rangle$.

– **Phase 1:** *Eve* runs five oracles, a signature oracle, a private key oracle, a private key base oracle, revocation oracle and a hash oracle. If *Adam* queries the hash oracle, *Eve* should keep a list of her responces to ensure randomness and consistancy for both hash functions H and H_0. In the rest of the oracles *Eve*'s reaction will be divided into three depending whether *Adam* queried i_0,i_1 or neither.

If *Adam* queries the signature oracle he should send an index i, a verifying key gpk for a certain attribute tree Γ, and a message M. If $(i \neq i_0, i_1)$; *Eve* will reply with a signature $\sigma = \langle r,C_1,C_2,C_3,C_4,c,s_\alpha,s_x,s_\delta, F_{root}\rangle$ as done in section[4]. If $(i = i_0)$, *Eve* picks a random $s,t,l,\beta \in Z_p$ and makes the following assignments:
$C_1 = h_0u_0^s$; $C_2 = ZWu_2^sh_0^tu_0^{st}$; $\hat{u} = u_0^l$; $\hat{v} = (u_2u_0^t)^l$.
Let $\alpha = (a+s)/l \in Z_p$. Then $C_1 = \hat{u}^\alpha$ and $C_2 = A_{i_0}\hat{v}^\alpha$.
Eve assigns $C_3 = \hat{e}(ZW,w)^\beta$ and $C_4 = w^\beta$. It calculates F_{root} by replacing the recursive algorithm $Sign_{Node}$ with $Fake-Sign_{Node}$, which is described below:

$$Fake-Sign_{Node}(leaf) = \begin{cases} \text{If } (j \in \Gamma); \text{ return } \hat{e}((u_2^sh_0^tu_0^{st})^{t_j}, D_j)^\beta \\ \qquad\qquad = \hat{e}(u_2^sh_0^tu_0^{st}, w)^{\beta q_j(0)} \\ \text{Otherwise return } \perp \end{cases}$$

F_{root} in this case will equal $\hat{e}(u_2^sh_0^tu_0^{st}, w)^{\beta s_{rnd}}$. Notice that $F_{root}^{1/s_{rnd}}.C_3 = \hat{e}(C_2, w)^\beta$. If β_1, β_2 are random elements in Z_p, it is hard to distinguish between the following two triples:
$\langle \hat{e}(v^\alpha, w)^{\beta_1}, w^{\beta_1}, \hat{e}(A_i, w)^{\beta_1 s_{rnd}}\rangle$ and $\langle \hat{e}(ZW, w)^{\beta_2}, w^{\beta_2}, \hat{e}(u_2^sh_0^tu_0^{st}, w)^{\beta_2 s_{rnd}}\rangle$.

If $(i = i_1)$, *Eve* picks a random $s,t,l,\beta \in Z_p$ and makes the following assignments:
$C_1 = h_1u_1^s$; $C_2 = Wh_1^tu_1^{st}/u_2^s$; $\hat{u} = u_1^l$; $\hat{v} = (u_1^t/u_2)^l$
Let $\alpha = (b+s)/l \in Z_p$. Then $C_1 = \hat{u}^\alpha$ and $C_2 = A_{i_1}\hat{v}^\alpha$.
Eve assigns $C_3 = \hat{e}(W,w)^\beta$ and $C_4 = w^\beta$. It calculates F_{root} by replacing

the recursive algorithm $Sign_{Node}$ with $Fake - Sign_{Node}$ which is described below:

$$Fake - Sign_{Node}(leaf) = \begin{cases} \text{If } (j \in \Gamma); \text{ return } \hat{e}((h_1^t u_1^{st}/u_2^s)^{t_j}, D_j)^{\beta} \\ \qquad\qquad = \hat{e}(h_1^t u_1^{st}/u_2^s, w)^{\beta q_j(0)} \\ \text{Otherwise return } \perp \end{cases}$$

F_{root} in this case will equal $\hat{e}(h_1^t u_1^{st}/u_2^s, w)^{\beta s_{rnd}}$. Notice that $F_{root}^{1/s_{rnd}}.C_3 = \hat{e}(C_2, w)^{\beta}$. If β_1, β_2 are random elements in Z_p, it is hard to distinguish between the triple :
$\langle \hat{e}(v^{\alpha}, w)^{\beta_1}, w^{\beta_1}, \hat{e}(A_i, w)^{\beta_1 s_{rnd}} \rangle$ and $\langle \hat{e}(W, w)^{\beta_2}, w^{\beta_2}, \hat{e}(h_1^t u_1^{st}/u_2^s, w)^{\beta_2 s_{rnd}} \rangle$

Eve chooses random values $r, c, s_{\alpha}, s_x, s_{\delta}$ from Z_p. *Eve* sets the values
$R_1 = u^{s_{\alpha}}/\psi(C_1)^c$, $R_3 = \psi(C_1)^{s_x}\psi(u)^{-s_{\delta}}$, and
$R_2 = \hat{e}(\psi(C_2), g_2)^{s_x}\hat{e}(\psi(\hat{v}), w)^{-s_{\alpha}}\hat{e}(\psi(\hat{v}), g_2)^{-s_{\delta}}(\hat{e}(\psi(C_2), w)/\hat{e}(g_1, g_2))^c$.
The probability that $H(gpk, M, \psi(C_1), \psi(C_2), \psi(C_3), \psi(C_4), R_1, R_2, R_3)$ or $H_0(gpk, M, r)$ have been queried before is at most q_H/p where q_H is the numbers of queries. If a collusion happens *Eve* reports a failure. Otherwise we add $H(gpk, M, \psi(C_1), \psi(C_2), \psi(C_3), \psi(C_4), R_1, R_2, R_3) = c$ and $H_0(gpk, M, r) = (\hat{u}, \hat{v})$ to the hash oracle's list.
Eve sends back the signature $\sigma = \langle r, \psi(C_1), \psi(C_2), \psi(C_3), \psi(C_4), c, s_{\alpha}, s_x, s_{\delta}, F_{root} \rangle$

When *Adam* issues a query on the private key oracle he needs to send *Eve* an attribute set Υ_i and an index i, where Υ_i should have at least an element j representing an attribute. *Eve* responds back with $\langle A_i, x_i, A_i^{t_1}, ..., A_i^{t_{\mu}} \rangle$. If *Adam* queries i_0, i_1, *Eve* reports failure. Finally when querying the revocation oracle *Adam* sends a users index i to revoke. *Eve* replies with adding A_i to R. If *Adam* queries i_0, i_1, *Eve* reports failure.

- **Challenge:** *Adam* asks to be challenged on message M, verification key gpk for a certain Γ and indexes i_0^* plus i_1^*. If $\{i_0^*, i_1^*\} \neq \{i_0^*, i_1^*\}$ then *Eve* reports failure. Otherwise, *Eve* picks randomly $b \in \{0, 1\}$ and generates a signature the same way it would have done in the signature query. So *Eve* responces back with a signature σ_b.

- **Phase 2:** Is exactly like phase 1.

- **Output :** *Adam* outputs a guess $\grave{b} \in \{0, 1\}$. If $b = \grave{b}$ then Z is random, otherwise $Z = u_2^{a+b}$.

There are two ways this game could end. Case one is when *Eve* does not abort. If Z is random then $Pr[b = \grave{b}] > 1/2 + \varepsilon$ otherwise if $Z = u_2^{a+b}$ then both signatures should be identical and therefore challenge is independent of b hence $Pr[b = \grave{b}] = 1/2$. So the advantage of *Eve* solving the linear challenge is at least $\varepsilon/2$.

The second case is Eve aborts and fails. Eve could abort in the signature queries with probability $q_S q_H/p$ where q_S is the number of signature queries and q_H are hash queries. The probability that all queries in phase 1 and the challenge do not cause Eve to abort is $1/n^2$. Concatenating both cases together the probability of Eve solving the linear challenge is $(\varepsilon/2)((1/n^2) - (q_S q_H)/p)$ as required.