Attribute Based Group Signatures

Dalia Khader
University of Bath

Abstract. An Attribute Based Group Signature (ABGS) allows a verifier to request a signature from a member of a group who possesses certain attributes. Therefore, a signature should authenticate a person in a group and prove ownership of certain properties. The major difference between our scheme and previous group signatures, is that the verifier can determine the role of the actual signer within the group.
In this paper we define the first ABGS scheme, and security notions such as anonymity and traceability. We then construct the scheme and prove it secure.

1 Introduction

Attribute Based Group Signature (ABGS) is a new paradigm of cryptography and a new generation of group signatures. The idea behind it is authenticating that a person has certain credentials. The following is a scenario where such a scheme is needed:
Alice wants a document to be signed by an employee in Bob's company. Alice requires that employee to have certain properties such as being part of the IT staff and at least a junior manager in the cryptography team or a senior manager in the biometrics team.
A possible solution for implementing such a scheme would be using Identity Based Group Signatures. An Identity Based Group Signature is a group signature where any member of the group could sign on behalf of the others and the signature could be verified using the identity of the group. For example, all members of the Cryptographic team in Bob's company will belong to the same group, where the public key is a template of the attribute "CryptoTeam" and each member gets his own private key. When a verifier requests a signature of an employee who satisfies certain attributes, a signer will use his different private keys to sign a document according to the verifier's request. However, there are problems in such a solution. First of all, the verification algorithm is run as many times as the number of attributes in the signature, thus compromising efficiency. Moreover, there is a security flaw in using Identity Based Group Signature; it is easy for different signers who do not satisfy the verifier's request to collude and create a valid signature if jointly they could satisfy the request. For example, John is part of the Cryptographic team and Smith is a junior manager. They could create a valid signature together on Alice's document. A possible fix would be to use the identity of the member but that compromises anonymity of the signer. The shortcoming of identity based schemes makes ABGS a new cryptographical problem that requires creating a new scheme.

Attribute based group signatures was designed to let the verifier request evidence from the signer that they own certain attributes. In our scenario, Alice starts building what we call an attribute tree. We adopt the idea of an attribute tree from Goyal et al's work in [16]. An attribute tree is a tree in which each interior node is a threshold gate and the leaves are linked with attributes. A threshold gate represents 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 1, which demonstrates an attribute tree for the scenario mentioned earlier.

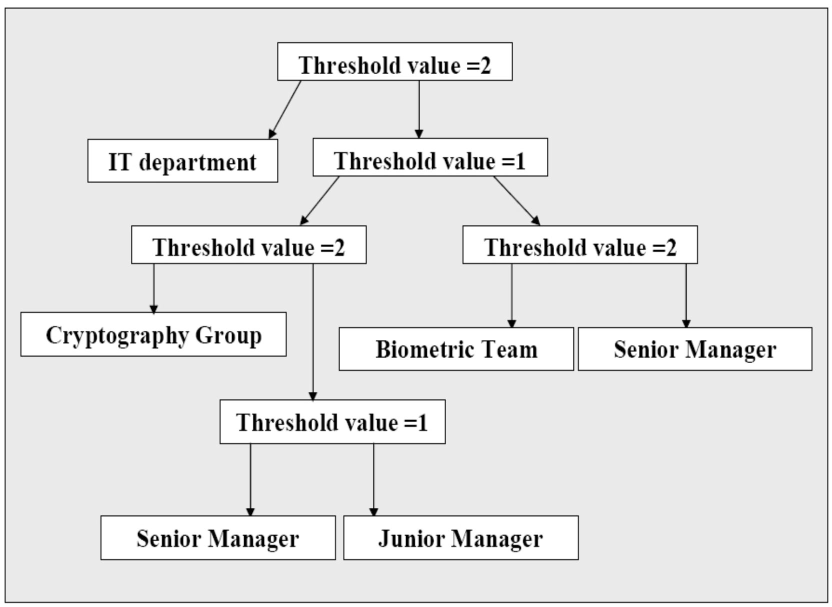

Fig. 1. Attribute Tree

Public keys used to verify signatures are labeled with such attribute trees. After Alice is done with building her attribute tree, she searches a corresponding public key in a lookup list provided by a third party, the Key Generator. If she doesn't find it, she could request generating such a key. The Key Generator adds every attribute tree and the corresponding verification key to the lookup list. Alice sends Bob's company the verification key. Only employees who satisfy the attribute tree could reply with a signature. The reason for that is each employee has a private key that implicitly contains the attributes he owns. For example, an employee who is a senior manager in the biometrics team would authenticate himself to the Key Generator. The Key Generator sends a private key that will

help him prove possession of the attributes when signing. Note that the Key Generator could be Bob's company if Alice trusts it as a company.
To argue about the security of our scheme we define full anonymity and full traceability in the ABGS context. Informally, a scheme is said to be anonymous if given a signature, it is computationally hard to identify the signer unless you are the manager, in this case the Key Generator. A scheme is traceable if the group manager is able to open a signature and trace it to a signer. More precisely, the major difference between our scheme and previous group signatures is that the verifier can determine the role of the actual signer within the group (i.e. which attributes are owned by the signer).

1.1 Related Work:

Since Chaum and van Heist's work, Group signatures have been of interest to cryptographers. Researchers have worked hard to add a variety of features to the scheme, defining different security notions and improving the performance of the scheme.
For instance, security notions such as Unforgeability, Anonymity, Unlinkability, Exculpability, Traceability, Coalition-Resistance, and Separatability were introduced. In [5] the authors tried to unify and simplify all these security notions by defining two core requirements: Full Anonymity and Full Traceability. They proved their definitions implicitly include all of the other security notions. Their proofs are specific to groups which have a group manager. In our work, since the group is also centralized, we prove the scheme to be Fully Anonymous and Fully Traceable.
A separate line of research was trying to add extra features to the scheme. In [12, 9, 18, 1, 10, 13, 8, 15, 21, 14] work was done to move group signatures from being static to more dynamic. In other words, we could add members anytime and revoke them if needed. Other cryptographers thought of creating Identity Based Group Signatures where the verification key is an identity of a group [23, 25, 27, 4]. In [19, 17, 28] Blind Group Signatures were proposed to be used in e-cash systems. In our paper we enable the verifier to decide the role of the signer within a group.

1.2 Outline:

The rest of the paper is organized as follows. We start with giving precise definitions and security models in 2. We describe certain preliminaries in Section 3. Our ABGS scheme is presented in Section 4. Section 5 gives conclusions and some open problems. The Appendix contains a more detailed discussion on the security proofs.

2 Definitions

Attribute Based Group Signature Schemes: An *Attribute Based Group Signature (ABGS)* scheme is specified by five algorithms: *Setup*, *KeyGen*, *Sign*,

$Verify$, *Open*. As a prerequisite to describing the algorithms we define certain notations.
$\varGamma$ will be used as a description to our attribute tree. The tree is read in a Top-Down-Left-Right manner. An interior node is written as (m, n) which represents a threshold gate m of n. For example, to represent the tree in Figure 1, $\varGamma = \{(2, 2)$, IT department, $(1, 2)$, $(2, 2)$, $(2, 2)$, Cryptography Team, $(1, 2)$, Biometric Team, Senior Manager, Senior Manager, Junior Manager$\}$. κ is the number of leaves in the tree.
$\varUpsilon_i$ is a set describing all private keys a member owns. For example, if Smith is a Junior Manager in the IT department, $\varUpsilon_{Smith} = \{$Junior Manager, IT department$\}$. The size of $\varUpsilon_i$ is represented by μ.
ζ_i is a set that describes the set of attributes which a signer uses to create his signature. In other words, $\zeta_i \subseteq \varUpsilon_i$, where elements in ζ_i are enough to satisfy $\varGamma$. For example, if the verifier is using $\varGamma = \{(1, 2)$, Junior Manger, Senior Manger$\}$, Smith could sign with $\zeta_{Smith} = \{$Junior Manager$\}$. τ is the size of ζ_i.
After having defined the notations we require, we could describe the algorithms as follows:

- *Setup*: A randomized algorithm that takes a security parameter as an input. It outputs a set of parameters S_{para} and a tracing key $gmsk$. S_{para} will be used in the $KeyGen$ algorithm. $gmsk$ will be used in the Open algorithm.

- $KeyGen(S_{para}, n)$: $KeyGen$ is an algorithm that takes the system parameters, and a number n that defines the number of users. It generates what is called private key bases $gsk[i]_{base}$ for any user i. It generates public keys and private keys using two sub-algorithms: $KeyGen_{public}$, $KeyGen_{priv}$.
  $KeyGen_{public}(\varGamma)$*:* This algorithm generates public keys gpk for attribute trees described in $\varGamma$(See Figure 1 as an example).
  $KeyGen_{priv}(gsk[i]_{base}, \varUpsilon_i)$*:* Creates the private key $gsk[i]$ for user i to enable him to authenticate himself and his properties which are described in $\varUpsilon_i$.

- $Sign(gpk, gsk[i], M)$: Given a public key of an attribute tree, a private key of a user i and a message, output a signature σ and ζ_i.

- $Verify(gpk, M, \sigma, \zeta_i)$: Given a message, a public key of a certain attribute tree, a signature and a set ζ_i, output either an acceptance or a rejection for the signature.

- $Open(S_{para}, gmsk, M, \sigma, \zeta_i)$: The Open algorithm is given a specific signature, a public key and the tracing key as inputs. Trace to the signer i even if it is a member in forging coalition. You could also trace the attributes that belong to ζ_i.

Definition 1. (ABGS Scheme is Correct:) *We say an ABGS Scheme is correct if and only if honestly-generated signatures verify and open correctly.*

2.1 Security Notions of the ABGS scheme

Anonymity and Traceability are the standard acceptable notions of security for Group Signatures [5, 7, 6]. Hence, it is natural to require that Attribute Based Group Signatures satisfy these security notions. However, the definition of those notions must be strengthened, to adjust to the fact that the verifier decides the role of a signer in a group. In our security model, the adversary could issue private key oracles for any attribute set Υ. The adversary chooses the attribute tree Γ in which he would like to be challenged upon. Finally, the adversary could issue signature oracles and decide the ζ_i of the signer. This section will describe the new definitions of Anonymity and Traceability.

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

- **Init:***Adam* chooses the attribute tree Γ he would like to be challenged upon.
- **Setup:** *Challenger* runs the *Setup* and *KeyGen* algorithms without running sub-algorithm $KeyGen_{priv}$.
  Challenger produces a public key for the attribute tree Γ and n private key bases gpk_{bases}.
- **Phase 1:***Challenger* runs a signature oracle and a private key oracle. *Adam* issues a certain number of queries to the signature oracle, sending in each time a message M, index of user i and a set of attributes ζ_i. *Challenger* responds with a signature σ. *Challenger* also runs a private key oracle. *Adam* sends an index i and a set of attributes Υ_i. *Challenger* responds with a private key. This oracle is equivalent to the $KeyGen_{priv}$.
- **Challenge:** *Adam* decides when to request his challenge. He sends the *Challenger* two indices (i_0, i_1), a message M and ζ. The triple $\langle i_0, M, \zeta\rangle$ and $\langle i_1, M, \zeta\rangle$ should not have been queried before in Phase 1 and should not be queried after this point in Phase 2. *Challenger* replies with a signature σ_b where $b \in \{0, 1\}$ and σ_b is the result of signing with the triple $\langle i_b, M, \zeta\rangle$
- **Phase 2:** Phase two is exactly the same as phase one.
- **Guess:***Adam* tries to guess $\grave{b} \in \{0, 1\}$. If $b = \grave{b}$, *Adam* succeeds otherwise he fails.

We refer to an adversary like *Adam* as the selective anonymity attack (SAA) adversary and we define the advantage of attacking the scheme as $Adv_{SAA} = Pr[b = \grave{b}] - 1/2$.

Definition 2. (Selective Anonymity:)
We say a scheme is secure under an SAA attack if for any polynomial time SAA-Adversary Adam, the advantage of winning the game is negligible. In other words, $Adv_{SAA} < \varepsilon$ *where* ε *is negligible.*

Traceability: 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* chooses the attribute tree Γ he would like to be challenged upon.
- **Setup:** *Challenger* runs the two algorithms: *Setup* and *KeyGen* algorithm except for the sub-algorithm $KeyGen_{priv}$. *Challenger* produces a public key gpk for the attribute tree and n private key bases $gsk[i]_{base}$.
- **Querying a Signature/Private key Oracle:**
  Challenger runs two oracles: a signature oracle and a private key oracle. *Adam* issues a number of queries to both oracles. He sends in every query to the signature oracle a message M, index of user i and a set of attributes ζ_i. *Challenger* responds back with a signature σ. When querying the private key oracle *Adam* sends an index i and a set of attributes Υ_i. *Challenger* responds with a valid private key $gsk[i]$.
- **Output:** If *Adam* is successful it outputs a forged signature σ that *Challenger* fails to trace using the open algorithm. Otherwise *Adam* fails.

We refer to *Adam*'s attack as the Un-Traceability Attack (UTA). We represent the advantage of the adversary in winning the attack as Adv_{UTA}.

Definition 3. (Traceability:) *An ABGS scheme is secure under a UTA attack if for any polynomial time UTA-Adversary, Adam, the advantage of winning the game is negligible. That is $Adv_{UTA} < \varepsilon$ where ε is negligible.*

In proving traceability, we need to show that a group of colluding members can not generate a valid signature, which does not trace to any member of the colluding group. That definition implicitly includes unforgeability and collision-resistance [5].

3 Preliminaries

In this section we will explain some of the preliminaries that are used in constructing the ABGS scheme and proving it secure.

3.1 The Strong Diffie-Hellman Assumption

This section defines q-Strong Diffie-Hellman and states the Boneh-Boyen Lemma which are two concepts that will be used in section 4.1 to prove traceability of the constructed scheme. 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 $g_2 \in G_2$ consider the following [6]:

Definition 4. (q-Strong Diffie-Hellman Problem)
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 [6].

This problem is considered hard to solve in polynomial time and ε should be negligible [6].

Theorem 1. (Boneh-Boyen SDH Equivalence)
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 [6] 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 [6].

3.2 Linear Encryption

In this section we will define an encryption scheme which depends on the difficulty of the Decision Linear Diffie-Hellman Assumption [7]. This scheme will be used in the construction of our ABGS scheme and will lead to ensuring anonymity(See Section 4.1) of the scheme.

Definition 5. (Decision Linear Problem in G_1)
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 $a + b = c$ [7].

Definition 6. (A Linear Encryption Scheme)
In a Linear Encryption scheme a user's public key is $u, v, h \in G_1$ [7]. 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 messsage 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.

3.3 Bilinear Maps

Bilinear Maps are used in constructing our ABGS in section 4.

Definition 7. (Bilinear Maps) [3]:
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.

3.4 Forking Lemma

Pointcheval and Stern [24], 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 Section 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 [24]:

Theorem 2. (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 $h \neq \grave{h}$.

4 Construction of an ABGS Scheme

In this section we construct an ABGS scheme based on Boneh et al's. work in Short Group Signatures in [7].

- *Setup*: Consider a bilinear pair (G_1, G_2) with a computable isomorphism ψ between them. Suppose that SDH assumption holds on (G_1, G_2) and the linear assumption holds on G_1. Define the bilinear map $\hat{e} : G_1 \times G_2 \to G_T$. All three groups G_1, G_2, G_T are multiplicative and of a prime order p. Select a hash function $H : \{0,1\}^* \to Z_p$. Select a generator $g_2 \in G_2$ at random and then set $g_1 \leftarrow \psi(g_2)$. Select $h \in G_1$ and ξ_1, ξ_2 randomly from Z_p. $gmsk = \langle \xi_1, \xi_2 \rangle$ will be used later in the open algorithm. Set $u, v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. Select a random γ from Z_p and set $w = g_2^{\gamma}$. Define a universe of attributes $U = \{1, 2, ..., m\}$ and for each attribute $j \in U$ choose a number t_j at random from Z_p. Let $S_{para} = \langle G_1, G_2, G_T, \hat{e}, H, g_1, g_2, h, u, v, gmsk, \gamma, w \rangle$.

- *KeyGen*(S_{para}, n): This algorithm generates a public key for a specific access structure and a private key for each user.
  Using γ generate for each user $i, 1 \leq i \leq n$ a private key base $gsk[i]_{base} = \langle A_i, x_i \rangle$. The $gsk[i]_{base}$ should be a SDH pair were x_i is chosen randomly from Z_p^* and $A_i = g_1^{1/(\gamma + x_i)} \in G_1$.
  KeyGen$_{public}$(Γ): To generate a public key for a certain attribute tree Γ we will need to choose a polynomial q_{node} of degree $d_{node} = k_{node} - 1$ for each

node in the tree, where k_{node} is the threshold gate. That is done in a top-down manner. Starting from the root $q_{root}(0) = \gamma$ 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 public key for a certain structure will be gpk=$\langle g_1, g_2, h, u, v, w, D_{leaf_1},...,D_{leaf_\kappa}, h_1,...,h_\kappa\rangle$ where $D_{leaf_j} = g_2^{q_{leaf_j}(0)/t_{leaf_j}}$, $h_j = h^{t_j}$.

$KeyGen_{priv}(gsk[i]_{base}, \Upsilon_i)$ For every attribute j that user i owns (i.e. $j \in \Upsilon_i$) calculate $T_{i,j} = g_1^{t_j/(\gamma+x_i)}$. The private key for a user i will be the tuple $gsk[i] = \langle A_i, x_i, T_{i,1}, ..., T_{i,\mu}\rangle$.

- $Sign(gpk, gsk[i], M)$: For signing user i, needs to do the following:
  Choose randomly a $\alpha, \beta, rnd \in Z_p$
  Compute the linear encryption of A_i and $T_{i,j}$ where $j \in \zeta$. The ciphertext of the encryption will equal
  $C_1 = u^\alpha$, $C_2 = v^\beta$, $C_3 = A_i h^{\alpha+\beta}$, $CT_j = (T_{i,j} h_j^{\alpha+\beta})^{rnd}$.
  Let $\delta_1 = x_i\alpha, \delta_2 = x_i\beta$.
  Choose randomly r_α,r_β,r_x,r_{δ_1} and r_{δ_2}.
  Calculate $R_1 = u^{r_\alpha}$, $R_2 = v^{r_\beta}$, $R_4 = C_1^{r_x} u^{-r_{\delta_1}}$,
  $R_3 = \hat{e}(C_3, g_2)^{r_x} \hat{e}(h, w)^{-r_\alpha - r_\beta}.\hat{e}(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}$ and $R_5 = C_2^{r_x} v^{-r_{\delta_2}}$.
  Let $c = H(M, C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5) \in Z_p$.
  Construct the values $s_\alpha = (r_\alpha + c\alpha)$, $s_\beta = (r_\beta + c\beta)$, $s_x = (r_x + cx)$, $s_{\delta_1} = (r_{\delta_1} + c\delta_1)$, and$s_{\delta_2} = (r_{\delta_2} + c\delta_2)$.
  Let $\eta = w^{rnd}$ The Signature will be $\sigma = \langle C_1, C_2, C_3, c, CT_1, ..., CT_\tau, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}, \eta\rangle$

- Verify(gpk, M, σ, ζ): To verify the signature we first define a recursive algorithm Ver_{Node}. If the node we are currently on is a leaf in the tree the algorithm returns the following:

$$Ver_{Node}(leaf) = \begin{cases} \text{If } (j \in zeta); \text{return } \hat{e}(CT_{leaf_j}, D_{leaf_j}) \\ \text{Otherwise; return } \perp \end{cases}$$

Notice that $\hat{e}(CT_{leaf_j}, D_{leaf_j}) = \hat{e}(A_i h^{\alpha+\beta}, g_2^{rnd})^{q_{leaf_j}(0)}$.
For a node ρ which is not a leaf the algorithm proceeds as follows: For all children z of the node ρ it calls Ver_{Node} and stores output as F_z. Let S_ρ be an arbitrary k_ρ sized set of child nodes z such that $F_z \neq \perp$ and if no such set exist return $\perp$. Otherwise let $\Delta_{S_\rho, index(z)} = \Pi(-j/(index(z) - j))$, where $j \in \{index(z) : z \in S_\rho - index(z)\}$and compute

$$F_\rho = \Pi_{z \in S_\rho} F_z^{\Delta_{S_\rho, index(z)}}$$

$$F_\rho = \Pi_{z \in S_\rho} \hat{e}(A_i h^{\alpha+\beta}, g_2^{rnd})^{q_z(0).\Delta_{S_\rho, index(z)}}$$

$F_\rho = \Pi_{z \in S_\rho} \hat{e}(A_i h^{\alpha+\beta}, g_2^{rnd})^{q_{parent(z)}(index(z)).\Delta_{S_\rho, index(z)}}$

$F_\rho = \hat{e}(A_i h^{\alpha+\beta}, g_2^{rnd})^{q_\rho(0)}$

To verify the signature calculate F_{root}. If the tree is satisfied then $F_{root} = \hat{e}(C_3, \eta)$ according to Lagrange interpolation.
Calculate $\bar{R_1} = u^{s_\alpha} C_1^{-c}$, $\bar{R_2} = v^{s_\beta} C_2^{-c}$, $\bar{R_4} = C_1^{s_x} u^{-s_{\delta_1}}$, $\bar{R_5} = C_2^{s_x} v^{-s_{\delta_2}}$,
$\bar{R_3}{=}\hat{e}(C_3, g_2)^{s_x}.\hat{e}(h, w)^{-s_\alpha - s_\beta}.\hat{e}(h, g_2)^{-s_{\delta_1} - s_{\delta_2}}.(\frac{\hat{e}(C_3, w)}{\hat{e}(g_1, g_2)})^c$.

If $c = H(M, C_1, C_2, C_3, \bar{R_1}, \bar{R_2}, \bar{R_3}, \bar{R_4}, \bar{R_5})$ then accept the signature, otherwise reject it.

– *Open*$(S_{para}, gmsk, t_1, ..., t_\tau, M, \sigma, \zeta)$: This algorithm traces a signature to a signer. To do so the key generator(i.e. our Group Manager) will be using:
The $S_{para}{=}\langle G_1,\, G_2,\, G_T,\, \hat{e},\, H,\, g_1,\, g_2,\, h,\, u,\, v,\, gmsk,\, \gamma,\, w\rangle$. The group masters tracing key $gmsk = \langle \xi_1, \xi_2 \rangle$.
Step one in the tracing will be verifying the signature. Afterwards, the group manager could recover A_i by calculating $A_i = C_3/(C_1^{\xi_1} C_2^{\xi_2})$. Now the manager could look up the user with index A_i. After finding the user, the manager could further up verify the attributes. For each attribute, he checks the following equality $\hat{e}(CT_j, w) = \hat{e}((A_i C_1^{\xi_1} C_2^{\xi_2})^{t_j}, \eta)$. If the equality holds for an attribute j then the j is said to be traced to the same user i.
The reason behind limiting the possibility of being the group manager to the key generator is the need to use t_j when calculating $T_{i,j}$. This is a minor drawback in our system where it is preferable to have some kind of hierarchy. For example, it would be practical if a senior manager could trace junior employees in his department rather than refering to the company every time.

4.1 Security of the scheme

In this section we prove the scheme to be correct. We also prove it to be secure under UTA and SAA attack(See section 2.1).

Theorem 3. *The ABGS scheme is correct.*

Proof. In order to do so we need to prove that $\bar{R_1} = R_1, \bar{R_2} = R_2, \bar{R_3} = R_3, \bar{R_4} = R_4, \bar{R_5} = R_5$ because that leads $c = H(M, C_1, C_2, C_3, \bar{R_1}, \bar{R_2}, \bar{R_3}, \bar{R_4}, \bar{R_5})$ which means the signature is accepted.
$\bar{R_1}{=}u^{s_\alpha} C_1^{-c}{=}u^{r_\alpha + c\alpha}.(u^\alpha)^{-c}{=}u^{r_\alpha}{=}R_1$
$\bar{R_2}{=}v^{s_\beta} C_2^{-c}{=}u^{r_\beta + c\beta}.(v^\beta)^{-c}{=}v^{r_\beta}{=}R_2$
$\bar{R_4}{=}C_1^{s_x}.u^{-s_{\delta_1}}{=}u^{\alpha(r_x + cx)}.u^{(-r\delta_1 - c\delta_1)}{=}C_1^{r_x}.u^{-r\delta_1}{=}R_4$
$\bar{R_5}{=}C_2^{s_x}.v^{-s_{\delta_2}}{=}v^{\beta(r_x + cx)}.v^{(-r\delta_2 - c\delta_2)}{=}C_2^{r_x}.v^{-r\delta_2} = R_5$
Finally,$\bar{R_3} = R_3$ holds for the following reasons:
$\hat{e}(C_3, g_2)^{s_x}.\hat{e}(h, w)^{-s_\alpha - s_\beta}.\hat{e}(h, g_2)^{-s_{\delta_1} \cdot - s_{\delta_2}}$

$= \hat{e}(C_3 h^{-\alpha-\beta}, wg_2^x)^c.\hat{e}(C_3, w)^{-c}(R_3)$
$= (\hat{e}(A, wg_2^x)/\hat{e}(C_3, w))^c R_3$
$= (\hat{e}(g_1, g_2)/\hat{e}(C_3, w))^c R_3$

Theorem 4. *If the linear encryption is IND-CPA secure then the ABGS scheme is fully anonymous, under the same attribute set, under the Random Oracle Assumption.*

In other words, if there is an adversary *Adam* that breaks the scheme's SSA security then there exists an adversary *Eve* that breaks into the linear encryption IND-CPA security. It makes sense to assume anonymity under the same attribute set, otherwise you could easily distinguish between signatures from attributes owned by each signer.
To prove Theorem 4, we run the adversarial model defined in section 2. We will assume we have an adversary *Adam* performing an SSA attack on the ABGS scheme. Let *Eve* be the adversary threatening the linear encryptions IND-CPA security. *Eve* will play a role of a challenger with *Adam*. She will make use of his talent to break the IND-CPA security. When *Adam* wants to Challenge, he sends i_0, i_1, a Message M and a set ζ to *Eve*. *Eve* has the values A_{i_0}, A_{i_1} since she is the one who ran the setup. She will give A_{i_0}, A_{i_1} as messages to challenge the IND-CPA security of the linear encryption. She will get back a ciphertext of one of them, A_{i_b}. The ciphertext is in the form $\bar{C} = \langle C_1, C_2, C_3\rangle$, where $C_1 = u^\alpha$, $C_2 = v^\beta$, and $C_3 = A_{i_b} h^{\alpha+\beta}$. *Eve* could calculate $CT_j = C_3^{rnd.t_j}$. She could then calculate $c, \eta, s_\alpha, s_\beta, s_x, s_{\delta_1}$, and s_{δ_2} as done in section 2. *Eve* sends *Adam* the signature of i_b as $\sigma_b = \langle C_1, C_2, C_3, c, CT_1, ..., CT_\mu, s_\alpha,\ s_\beta,\ s_x,\ s_{\delta_1},\ s_{\delta_2}, \eta\rangle$. Notice that *Eve* herself does not know b. If *Adam* could break the ABGS anonymity, he will send *Eve* the right value of b. *Eve* will use it to know whether A_{i_0} or A_{i_1} has been encrypted. Therefore, *Eve* breaks the IND-CPA security of linear encryption. In Appendix C we describe more details about the proof.

Theorem 5. *If SDH is hard on group (G_1, G_2) then the selective model of the Attribute Based Group Signature Scheme is fully-traceable under the Random Oracle assumption.*

In other words, if there is an adversary *Adam* that attacks the UTA security of the scheme then the SDH problem is solved. The proof of Theorem 5 is detailed in Appendix B. A simplified version will be explained in this section.
In our proof we use the game described in section 2, the Forking Lemma(Theorem 2), and Boneh-Boyen Lemma (Theorem 1). A signature will be represented as $\langle M, \sigma_0, c, \sigma_1, \sigma_2\rangle$. M is the signed message. $\sigma_0 = \langle C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5\rangle$. c is the value derived from hashing σ_0. $\sigma_1 = \langle s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}\rangle$ which are values used to calculate the missing inputs for the hash function. Finally $\sigma_2 = \langle CT_1, ..., CT_\tau, \eta\rangle$ the values that depend on the set of attributes in each signature oracle.
We will run the game in section 2 twice. In both simulated runs, the *Challenger* is given an (n)SDH instance, $(\grave{g}_1, \grave{g}_2, \grave{g}_2^\gamma, \grave{g}_2^{\gamma^2}, ..., \grave{g}_2^{\gamma^q})$. By applying the Boneh and Boyen's Lemma found in [6], *Challenger* could obtain $g_1 \in G_1, g_2 \in G_2, w = g_2^\gamma$

and $(n-1)$ SDH pairs (A_i, x_i) which he will use as the private key bases $gsk[i]_{base}$. The next step is showing how the Forking Lemma (Section 2) could be applied here to prove that we could generate new SDH pairs, if a forgery of any type exists. The difference between the two simulated runs is the response to the hash oracle (See Appendix B). According to the Forking Lemma, if *Adam* could find with non-negligible probability a valid signature $\langle M, \sigma_0, c, \sigma_1, \sigma_2\rangle$, then with a replay another valid signature $\langle M, \sigma_0, \grave{c}, \grave{\sigma}_1, \sigma_2\rangle$ is outputted with a non-negligible probability.
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_β,Δs_x,Δs_{δ_1}, and Δs_{δ_2}.
Divide two instances of the equations used previously(See Theorem 3 proof) where one instance is with $\grave{c}$ and the other is with c to get the following:

- Dividing $R_1/\grave{R}_1$ we get
  $u^{\tilde{\alpha}} = C_1$; where $\tilde{\alpha} = \Delta s_\alpha/\Delta c$
- Dividing $R_2/\grave{R}_2$ we get
  $v^{\tilde{\beta}} = C_2$; where $\tilde{\beta} = \Delta s_\beta/\Delta c$
- Dividing $C_1^{s_x}/C_1^{\grave{s}_x} = u^{s_{\delta_1}}/u^{\grave{s}_{\delta_1}}$ will lead to
  $\Delta s_{\delta_1} = \tilde{\alpha}\Delta s_x$
- Dividing $C_2^{s_x}/C_2^{\grave{s}_x} = v^{s_{\delta_2}}/u^{\grave{s}_{\delta_2}}$ will lead to
  $\Delta s_{\delta_2} = \tilde{\beta}\Delta s_x$
- Calculating the following equality:
  $(\hat{e}(g_1, g_2)/\hat{e}(C_3, w))^{\Delta c}$
  $= \hat{e}(C_3, g_2)^{\Delta s_x}.\hat{e}(h, w)^{-\Delta s_\alpha - \Delta s_\beta}.\hat{e}(h, g_2)^{-\Delta s_{\delta_1} - \Delta s_{\delta_2}}$
  $= \hat{e}(C_3, g_2)^{\Delta s_x}.\hat{e}(h, w)^{-\Delta s_\alpha - \Delta s_\beta}.\hat{e}(h, g_2)^{-\tilde{\alpha}\Delta s_x - \tilde{\beta}\Delta s_x}$

From the equations above if we let $\tilde{x} = \Delta s_x/\Delta c$ and $\tilde{A} = C_3 h^{-(\tilde{\alpha}+\tilde{\beta})}$ we get the following equation:
$\hat{e}(g_1, g_2)/\hat{e}(C_3, w)=\hat{e}(C_3, g_2)^{\tilde{x}}.\hat{e}(h, w)^{-\tilde{\alpha}-\tilde{\beta}}\hat{e}(h, g_2)^{-\tilde{x}(\tilde{\alpha}+\tilde{\beta})}$
$\hat{e}(g_1, g_2)=\hat{e}(\tilde{A}, w g_2^{\tilde{x}})$
Hence we obtain a new SDH pair $(\tilde{A}, \tilde{x})$ breaking Boneh and Boyens Lemma(See Section 1).

5 Conclusion

We proposed a new group signature scheme that enables a verifier to decide the character of the signer within the group, which we refer to as the Attribute Based Group Signature(ABGS). We have defined security models for the notions Anonymity and Traceability. We construct the first ABGS and prove it to be secure against SSA and UTA attacks. The next step would be to have signatures and keys within our scheme, independent on the attributes. It is an open problem to construct a scheme that could be proven secure in a standard model.

References

1. G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *Proceedings of Financial Cryptography'01*, volume 2357 of *Lecture Notes in Computer Science*, pages 183–197. Springer-Verlag, 2001.
2. G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Financial Crypto'99*, volume 1648, pages 196–211, 1999.
3. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
4. 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.
5. 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.
6. 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.
7. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *In Proceedings Crypto'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.
8. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *In Proc. of the 11th ACM Conference on Computer and Communications Security*, pages 168–177, 2004.
9. 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.
10. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings of Crypto'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer-Verlag, 2002.
11. 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.
12. L. Chen and T.P. Pedersen. New group signature schemes. In *Proceedings of Eurocrypt 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1994.
13. Z. Chen, J. Wang, Y. Wang, J. Huang, and Daren Huang. An efficient revocation algorithm in group signatures. In *Information Security and Cryptology - ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 339–351. Springer-Verlag, 2004.
14. C. Delerable and D. Pointcheval. Dynamic fully anonymous short group signatures. In *In: VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer-Verlag, 2006.
15. X. Ding, G. Tsudik, and S. Xu. Leak-free group signatures with immediate revocation. In *Proc. of 24th International Conference on Distributed Computing Systems*, pages 608–615, 2004.

16. 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.
17. 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.
18. H. Kim, J. Lim, and D. Lee. Efficient and secure member deletion in group signature schemes. In *Information Security and Cryptology*, volume 2015 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
19. 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.
20. M. Manulis, A. Sadeghi, and J. Schwenk. Linkable democratic group signatures. In *Information Security Practice and Experience*, volume 3903 of *Lecture Notes in Computer Science*, pages 187–201. Springer-Verlag, 2006.
21. T. Nakanishi and Y. Sugiyama. A group signature scheme with efficient membership revocation for reasonable groups. In *Information Security and Privacy*, volume 3108 of *Lecture Notes in Computer Science*, pages 336–347. Springer-Verlag, 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. S. Park, S. Kim, and D. Won. Id-based group signature. In *Electronics Letters*, pages 1616–1617, 1997.
24. 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.
25. 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.
26. V Wei. Tracing-by-linking group signatures. In *Proceedings Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 149–163. Springer-Verlag, 2005.
27. 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.
28. F. Zhang and K. Kim. Id-based blind signature and ring signature from pairings. In *ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer-Verlag, 2002.

A Extra Preliminaries

A.1 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 8. (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 9. (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.*

B ABGS scheme Traceablity

Theorem 6. *If SDH is hard on group (G_1, G_2) then the selective model of the Attribute Based Group Signature Scheme is fully-traceable under the Random Oracle assumption. In other words, if there exists an adversary that attacks the UTA security of the scheme then there exist an adversary that could solve the SDH problem.*

Proof. In order to prove that we need three steps. Defining a security model for proving full-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 full traceability of the signature scheme. The security model will be defined as an interacting framework between the *Challenger* and *Adam* as follows:

- **Init:** The *Challenger* runs *Adam*. *Adam* chooses the attribute tree Γ it would like to be challenged upon.
- **Setup:** The *Challenger* runs the setup algorithm as in section 2 with a bilinear pair (G_1, G_2). It selects the generators g_1, g_2, a hash function H, ξ_1, ξ_2, u,v,h, and γ such that they all satisfy properties mentioned in section 2. It also chooses a t_j for all attributes j in the tree *Adam* gave. The *Challenger* has to come up with the pairs $\langle A_i, x_i \rangle$ for an $i = 1, ..., n$. Some of those pairs have $x_i = \star$ which implies that x_i corresponding to A_i is not known; Other pairs are a valid SDH pair. In the Setup *Challenger* creates a public key for the same attribute tree. So *Adam* is given $gpk = \langle g_1, g_2, h, u, v, w, D_{leaf_1}, ..., D_{leaf_\kappa}, h_1, ..., h_\kappa \rangle$ and (ξ_1, ξ_2).
- **Hash Queries:** When the *Challenger* asks *Adam* for the hash of $(M, C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5)$, *Adam* responds with a random element in G_1 and saves the answer just incase the same query is requested again.
- **Signature Queries:** *Adam* asks for a signature on a message M by a key index i and a set of attributes ζ; where ζ satisfies the attribute tree chosen in Setup. If $x_i \neq \star$ *Challenger* calculates $T_{i,j} = A_i^{t_j}$ for all attributes in ζ and signs the message normally to obtain σ and give it to *Adam*. If $x_i = \star$ then *Challenger* picks randomly $\alpha, \beta, rnd \in Z_p$ sets $C_1 = u^\alpha, C_2 = v^\beta, C_3 = A_i g_1^{\alpha+\beta}$, and $CT_j = (A_i g_1^{\alpha+\beta})^{rnd.t_j}$ for every attribute in ζ. Now *Challenger* could get σ as shown in the signature algorithm and give it to *Adam*
- **Private Key Queries:** *Adam* asks for the private key in a certain index i for an attribute set Υ. If $x_i \neq \star$, *Challenger* returns back $\langle A_i, x_i, T_{i,1}, ..., T_{i,\tau} \rangle$ where $T_{i,j} = A_i^{t_j}$ otherwise *Challenger* declares failure.
- **Output:** If *Adam* is successful, it outputs a forged signature on a message M. The signature should verify correctly yet not trace to a member that has been queried. *Challenger* runs the verify then the open algorithm. He then tests the

A^* he calculated through the open algorithm. If $A^* \neq A_i$ for all i output σ. If $A^* = A_{i^*}$ for some $i^*$and if $s_{i^*} = \star$ output σ. The only possibility left is having $A^* = A_{i^*}$ but $s_i \neq \star$ *Challenger* declares failure.

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 $A^* = A_{i^*}$ 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 Lemma 1 for a $(n+1)$SDH, we could obtain g_1,g_2 and w. We could also use the n pairs (A_i, x_i) to calculate the private keys $\langle A_i, x_i, A_i^{t_1}, ..., A_i^{t_\mu}\rangle$. 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 Lemma 1 but for an nSDH this time, we could obtain g_1, g_2 and w. Then we could also use the $n-1$ pairs (A_i, x_i) to calculate the private keys $\langle A_i, x_i, A_i^{t_1}, ..., A_i^{t_\mu}\rangle$. In a random index i^*, we could choose the missing pair randomly where $A_{i^*} \in G_1$ and set $x_{i^*} = \star$. The random private key will be $\langle A_{i^*}, x_{i^*}, A_{i^*}^{t_1}, ..., A_{i^*}^{t_\mu}\rangle$. *Adam* in the security model will fail if he queries the private key 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.
The next step is showing how the Forking Lemma (Section 2) could be applied here to prove that we could generate new SDH pairs if a forgery of any type exists. 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. $\sigma_0 = \langle C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5\rangle$. c is the value derived from hashing σ_0. $\sigma_1 = \langle s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}\rangle$ which are values used to calculate the missing inputs for the hash function. Finally $\sigma_2 = \langle CT_1, ..., CT_\tau, \eta\rangle$ the values that depend on the set of attributes in each signature oracle.
One simulated run of the adversary is described by a random string ω used by the adversary *Adam* and a vector ℓ of the responses made by the hash oracle. Let S be a set of the pairs $\langle \omega, \ell\rangle$ where *Adam* successfully forges the signature $(M, \sigma_0, c, \sigma_1, \sigma_2)$. Let $Ind(\omega, \ell)$ be the index of ℓ on which *Adam* queried (M, σ_0). Let $\nu = Pr[S] = \grave{\varepsilon} - 1/p$ which represents the probability of the security model ending with a success subtracting the possibility that *Adam* guessed the hash of (M, σ_0) without the help of the hash oracle. For each χ, $1 \leq \chi \leq q_H$, let S_χ be a set of pairs $\langle \omega, \ell\rangle$ where $Ind(\omega, \ell) = \chi$. Let $\varPhi$ be the set of indices χ where $Pr[S_\chi|S] \geq 1/2q_H$ causing $Pr[Ind(\omega, \ell) \in \varPhi|S] \geq 1/2$. Let $\ell|_a^b$ be the restriction of ℓ to its elements at indices $a, a+1, ..., b$. For each $\chi \in \varPhi$ consider the heavy row lemma (See Section A.1) with a matrix with rows indexed with$(\omega, \ell|_1^{\chi-1})$ 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 $\varOmega_\chi$ be the ones such that $\forall(x, y) \in \varOmega_\chi : Pr_{\grave{y}}[(x, \grave{y}) \in S_\chi] \geq \nu/(4q_H)$. By the heavy row lemma $Pr[\varOmega_\chi|S_\chi] \geq 1/2$ which leads to $Pr[\exists\chi \in \varPhi : \varOmega_\chi \cap S_\chi|S] \geq 1/4$. Therefore *Adam*'s probability in forging a signature is about $\nu/4$. That signature derives from the heavy row $(x, y) \in \varOmega_\chi$ for some $\chi \in \varPhi$, hence execution (ω, ℓ) such that the $Pr_{\grave{\ell}}[(\omega, \grave{\ell}) \in S_j|\grave{\ell}|_1^{j-1} = \ell|_1^{j-1}] \geq \nu/(4q_H)$. 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 $\varDelta c = c - \grave{c}$, $\varDelta s_\alpha = s_\alpha - \grave{s}_\alpha$, and similarly for $\varDelta s_\beta$,$\varDelta s_x$,$\varDelta s_{\delta_1}$, and $\varDelta s_{\delta_2}$.

Divide two instances of the equations used previously (See Theorem 3 proof) where one instance is with $\grave{c}$ and the other is with c to get the following:

- Dividing $R_1/\grave{R}_1$ we get
  $u^{\tilde{\alpha}} = C_1$; where $\tilde{\alpha} = \Delta s_\alpha/\Delta c$
- Dividing $R_2/\grave{R}_2$ we get
  $v^{\tilde{\beta}} = C_2$; where $\tilde{\beta} = \Delta s_\beta/\Delta c$
- Dividing $C_1^{s_x}/C_1^{\grave{s}_x} = u^{s_{\delta_1}}/u^{\grave{s}_{\delta_1}}$ will lead to
  $\Delta s_{\delta_1} = \tilde{\alpha}\Delta s_x$
- Similarly dividing $C_2^{s_x}/C_2^{\grave{s}_x} = v^{s_{\delta_2}}/u^{\grave{s}_{\delta_2}}$ will lead to
  $\Delta s_{\delta_2} = \tilde{\beta}\Delta s_x$
- Calculating the following equality:
  $(\hat{e}(g_1, g_2)/\hat{e}(C_3, w))^{\Delta c}$
  $= \hat{e}(C_3, g_2)^{\Delta s_x}.\hat{e}(h, w)^{-\Delta s_\alpha - \Delta s_\beta}.\hat{e}(h, g_2)^{-\Delta s_{\delta_1} - \Delta s_{\delta_2}}$
  $= \hat{e}(C_3, g_2)^{\Delta s_x}.\hat{e}(h, w)^{-\Delta s_\alpha - \Delta s_\beta}.\hat{e}(h, g_2)^{-\tilde{\alpha}\Delta s_x - \tilde{\beta}\Delta s_x}$

From the equations above if we let $\tilde{x} = \Delta s_x/\Delta c$ and $\tilde{A} = C_3 h^{-(\tilde{\alpha}+\tilde{\beta})}$ we get the following equation:
$\hat{e}(g_1, g_2)/\hat{e}(C_3, w) = \hat{e}(C_3, g_2)^{\tilde{x}}.\hat{e}(h, w)^{-\tilde{\alpha}-\tilde{\beta}}\hat{e}(h, g_2)^{-\tilde{x}(\tilde{\alpha}+\tilde{\beta})}$
$\hat{e}(g_1, g_2) = \hat{e}(\tilde{A}, w g_2^{\tilde{x}})$
Hence we obtain a new SDH pair $(\tilde{A}, \tilde{x})$ breaking Boneh and Boyens Lemma (See Section 1). Now putting things together we get the following claims:

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

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

C ABGS Scheme Anonymity

Theorem 9. *If the linear encryption is IND-CPA secure then the ABGS scheme is fully anonymous under the same attribute tree under the Random Oracle Assumption. In other words, if there exists an adversary that breaks the scheme's SSA security then there exists an adversary that breaks into the linear encryption IND-CPA security.*

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

- **Init:** *Adam* decides the attribute tree Γ he would like to be challenged upon and gives it to *Eve*.
- **Setup:** *Eve* is given the public key $LE_{PK} = \langle u, v, h\rangle$ from the *Challenger*. *Eve* chooses a random γ from Z_p and $t_1, ..., t_m$. Using the LE_{PK} key and the random values, *Eve* could calculate an ABGS public key for the attribute tree $gpk = \langle u, v, h, w, D_{leaf_1}, ..., D_{leaf_\kappa}, h_1, ..., h_\kappa\rangle$ for the ABGS scheme. *Eve* also calculates n private key bases $gsk[i]_{base} = \langle A_i, x_i\rangle$ where $1 \le i \le n$.

- **Phase 1:** *Eve* runs three oracles: a signature oracle, private key oracle and a hash oracle. The hash oracle has a list that saves a unique random value for each 9-element tuple. That random value is the response of the oracle. The hash oracle should guarantee that no 9-element tuple have the same random value and that each time it responds with the same random value for the same 9-element tuple. In the signature oracle *Adam* sends an index i, a random message M and a set of attributes ζ to *Eve* where ζ satisfies the tree. *Eve* responds back with a signature $\sigma = \langle C_1, C_2, C_3, c, CT_1, ...CT_\tau, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}, \eta\rangle$ on that message from user i. c is the response of the hash oracle for the tuple $\langle M, C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5\rangle$. Finally, the private key oracle *Adam* sends an index i and a set of attributes Υ and *Eve* responds back with $\langle A_i, x_i, A_i^{t_1}, ..., A_i^{t_\mu}\rangle$.
- **Challenge:** Now *Adam* could request from *Eve* his anonymity challenge by choosing two indices (i_0 and i_1), set of attributes ζ and a message M asking for a signature of one of them. *Eve* sends *Challenger* both $\langle A_{i_0}, A_{i_1}\rangle$ as messages requesting a challenge. *Challenger* responds back with the ciphertext $\bar{C} = \langle C_1, C_2, C_3\rangle$ of A_{i_b} where $b \in \{0, 1\}$. *Eve* generates a signature from $\bar{C}$=$\langle C_1$,C_2,C_3,$C_3^{rnd.t_1}$,...,$C_3^{rnd.t_\tau}$,$w^{rnd}\rangle$ and sends it to *Adam*.
- **Phase 2:** *Adam* goes back to issuing further queries as done in Phase one.
- **Guess:***Adam* returns a $\grave{b}$ to *Eve*.

Eve outputs $\grave{b}$ as her answer to the *Challenger*. *Eve* has a high advantage on guessing the right $\grave{b} = b$ if and only if *Adam* could break into the anonymity of the ABGS scheme.