Revocable Attribute-Based Signatures with Adaptive Security in the Standard Model

Àlex Escala, Javier Herranz, and Paz Morillo

Dept. Matemàtica Aplicada IV, Universitat Politècnica de Catalunya,
C. Jordi Girona 1-3, Mòdul C3, 08034, Barcelona, Spain.
e-mail: alexescala@gmail.com, jherranz@ma4.upc.edu, paz@ma4.upc.edu

Abstract. An attribute-based signature with respect to a signing policy, chosen ad-hoc by the signer, convinces the verifier that the signer holds a subset of attributes satisfying that signing policy. Ideally, the verifier must obtain no other information about the identity of the signer or the attributes he holds. This primitive has many applications in real scenarios requiring both authentication and anonymity/privacy properties.

We propose in this paper the first attribute-based signature scheme satisfying at the same time the following properties: (1) it admits general signing policies, (2) it is proved secure against fully adaptive adversaries, in the standard model, and (3) the number of elements in a signature depends only on the size of the signing policy. Furthermore, our scheme enjoys the additional property of revocability: an external judge can break the anonymity of a signature, when necessary. This property may be very interesting in real applications where authorities are unwilling to allow full anonymity of users.

Keywords: attribute-based signatures, Groth-Sahai proofs, unforgeability, non-linkability, revocability.

1 Introduction

Attribute-based cryptography has emerged in the last years as a very interesting and powerful paradigm. In an attribute-based cryptosystem, the secret operation (signing or decrypting) can be performed only by users who hold a subset of attributes that satisfy some policy. A successful execution of the secret operation should leak no information about the identity of the user or the attributes he holds, other than the fact that these attributes satisfy the given policy. Thanks to that property, attribute-based cryptography has a lot of applications in real-life scenarios where users want to preserve some level of privacy. For example, any attribute-based encryption or signature scheme can be used to implement a private access control mechanism: a server chooses and publishes an access policy, and then users who want to access some restricted resource (digital information, access to a building, etc.) must be able to decrypt or sign a fresh challenge chosen by the server. Attribute-based cryptosystems must satisfy a collusion-resistance property: if a set of users, each of them holding attributes that do not satisfy the given policy, collude and try to perform the secret operation, they must fail to do so, even if the union of all their attributes satisfies the policy.

The notion of attribute-based cryptography appeared explicitly in [6] for the first time, as an extension of fuzzy identity-based cryptography [15]. Since then, the notion of attribute-based encryption has received a lot of attention, see for example [2, 10, 12]. Regarding attribute-based signatures, they were introduced explicitly in the first version of [14]. After that, other attribute-based encryption schemes have been proposed in [17, 13].

In an attribute-based signature (we will use sometimes ABS, for short) scheme, users receive from a master entity a secret key which depends on the attributes that they hold. Later, a user can choose a signing policy (a monotone increasing family of subsets of attributes) satisfied by his attributes, and use his secret key to compute a signature on a message, for this signing policy. The verifier of the signature is convinced that some user holding a set of attributes satisfying the signing policy is the author of the signature, but does not obtain any other information about the actual identity of the signer or the attributes he holds. Besides the general applications of any attribute-based cryptosystem (such as private access control), this kind of signatures have many applications in specific scenarios where both authentication and privacy properties are desired. A typical example is the leakage of secrets; see [14] for other applications.

All the attribute-based signature schemes that have been proposed up to date have some drawback in their efficiency, functionality or security analysis. We propose in this paper a new attribute-based signature scheme which overcomes these drawbacks. Namely, our scheme is the first one enjoying at the same time the following properties:

(1) it admits general signing policies,

(2) its security against adaptive adversaries is proved in the standard model,

(3) the number of elements in a signature depends linearly on the size of the signing policy.

Table 1 summarizes the state of the art in attribute-based signatures, and the contribution of our new scheme. In the table, λ denotes a security parameter (the size of the underlying mathematical groups), and $|\Gamma|$ denotes the size of the mathematical object used to represent the signing policy. For example, in the case of (ℓ, n)-threshold signing policies, containing all subsets of at least ℓ attributes among a set of n attributes, we have $|\Gamma| = n$. Note that the difference between λ and $|\Gamma|$ can be quite significant; for example, in typical threshold scenarios the number of involved attributes can be $|\Gamma| = n \approx 20$, whereas $\lambda \approx 320$. Selective adversaries are those who choose the signing policy they want to attack at the very beginning, before having access to secret key or signing queries. In contrast, adaptive adversaries are more powerful: they can choose the attacked signing policy much later. Proving security against adaptive adversaries is obviously much better than proving security against selective adversaries.

ABS scheme	#elements in a signature	admitted policies	considered adversaries	model for the security proof
Instantiations 1,2 in [14]	$\mathcal{O}(\lambda)$	general	adaptive	standard
Instantiation 3 in [14]	$\mathcal{O}(\|\Gamma\|)$	general	adaptive	generic group
[17, 13]	$\mathcal{O}(\|\Gamma\|)$	threshold	selective	standard
Our scheme	$\mathcal{O}(\|\Gamma\|)$	general	adaptive	standard

Table 1. Comparison between existing attribute-based signature schemes.

We construct our scheme in different steps. First we concentrate on the case of threshold signing policies, and we start with a basic scheme which produces linkable signatures. The design of this first scheme is inspired by the ring signature scheme of Shacham-Waters [16]. Then we add more technical tools to provide non-linkability and anonymity to the signatures. Specifically, we use Groth-Ostrovsky-Sahai [8] and Groth-Sahai [9] proofs. Such proofs have been proved very useful in the design of signature schemes with some anonymity properties, such as ring signatures [4, 16] and group signatures [3, 7]. Our second scheme can be proved secure in the random oracle model. Finally, we explain which modifications can be applied to this second scheme in order to admit more general signing policies (possibly at the cost of an increase in the length of the signatures) and to achieve provable security in the standard model (at the cost of an increase in the length of the public parameters).

Interestingly, our schemes enjoy the additional property of revocability: the master entity can send some secret information to some special user, for example a judge. This user may then revoke the anonymity of an attribute-based signature, when needed, by tracing this signature to the user who computed it. To the best of our knowledge, previous attribute-based signature schemes do not satisfy this property. Revocability can be really useful when implementing the primitive of attribute-based signatures in real-life scenarios, because authorities do not usually like the idea of full anonymity.

2 Preliminaries

In this section we review some concepts, hardness assumptions and cryptographic primitives that will appear in the description and analysis of our schemes.

2.1 Symmetric Bilinear Groups and Hardness Assumptions

A symmetric bilinear group is a tuple $(n, \mathbb{G}, \mathbb{G}_T, e, g)$ where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order n (which can be prime or composite), g generates $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a pairing, i.e., an efficiently computable non-degenerate bilinear map.

The security of our schemes is based on different assumptions. Given a prime order symmetric bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, the *CDH assumption* states that any probabilistic polynomial time algorithm that takes as input a tuple $(g, g^a, g^b) \in \mathbb{G}^3$ outputs $g^{ab} \in \mathbb{G}$ only with negligible probability. We will use the CDH assumption in the subgroup of order p of a composite order symmetric bilinear group $\mathbb{G}$ to prove the unforgeability of our scheme.

Given a composite order symmetric bilinear group $(n, \mathbb{G}, \mathbb{G}_T, e, g)$ with $n = pq$ the product of two large primes, the *subgroup decision assumption* states that it is hard to distinguish an element in $\mathbb{G}$ from an element in $\mathbb{G}_q$, the subgroup of order q of $\mathbb{G}$. This assumption is needed to construct non-interactive witness indistinguishable proofs to provide anonymity to our scheme.

Finally, an automorphic signature scheme will be used in the design of the schemes, and the hardness assumptions ensuring the security of such automorphic signature scheme will be inherited by our scheme.

2.2 Automorphic Signatures

An automorphic signature scheme is a signature scheme that satisfies the following properties: the verification keys lie in the message space, messages and signatures consist of elements of a bilinear group, and verification is done by evaluating a set of pairing-product equations. We will use an automorphic signature in the design of our scheme, essentially as a black-box.

Instantiations of automorphic signature schemes can be found in [5, 1]. Therein, automorphic signature schemes using either symmetric or asymmetric bilinear groups are presented. For the symmetric case (the one that we consider here), the security of the scheme is based on the *q-DHSDH* (q-Double Hidden Symmetric Diffie-Hellman) and *WFCDH* (Weak Flexible Computational Diffie-Hellman) assumptions. These are non-standard but reasonable assumptions: under the Knowledge of the Exponent Assumption, the first assumption is equivalent to the *q-SDH-III* (q-Strong Diffie-Hellman III) assumption, which is a bit weaker than the quite standard *q-SDH* assumption. Under the same Knowledge of the Exponent Assumption, the asymmetric version of the *WFCDH* assumption is equivalent to the standard discrete logarithm assumption.

2.3 NIWI Proofs for Pairing Product Equations

Groth, Ostrovsky and Sahai [8] and Groth and Sahai [9] propose two different methodologies to construct non-interactive witness indistinguishable (NIWI) proofs for different statements. In our scheme we will use both kinds of proofs.

First, Groth,Ostrovsky and Sahai [8] propose a construction of NIWI proofs for all NP languages. More specifically, they constructed proofs for circuit satisfiability. We are just interested in a particular step of the construction: a NIWI proof that a commitment contains 0 or 1. The setup algorithm outputs a bilinear group $(n, \mathbb{G}, \mathbb{G}_T, e, g)$, where g is a generator of $\mathbb{G}$ and $n = pq$ is the product of two large primes, and also an element $h \in \mathbb{G}$ of order q. The commitment to $m \in \{0, 1\}$ is $c = g^m h^r$, the NIWI proof is computed as $\pi = (g^{2m-1} h^r)^r$ and the verifier must check if $e(c, c/g) = e(h, \pi)$. As proved in [8], this proof is correct, sound and witness indistinguishable. Instead of using a unique value g, we will use different values (the hash of some attributes).

Groth and Sahai [9] propose a construction of NIWI proofs of the satisfiability of equations in bilinear groups. They give three instantiations of their methodology based on three different assumptions. In our scheme, we will mainly use the instantiation based on the subgroup decision assumption. The methodology of Groth-Sahai applies to different kinds of equations, but we are only interested in pairing product equations, that is, those of the form

$$\prod_{i=1}^{r} e(g_i, X_i) \cdot \prod_{i=1}^{r} \prod_{j=1}^{s} e(X_i, Y_j)^{\gamma_{ij}} = t_T$$

where g_i, t_T and γ_{ij} are public constants in $\mathbb{G}$, G_T and $\mathbb{Z}_n$ respectively, and X_i and Y_j are secret variables in $\mathbb{G}$.

The setup algorithm outputs a bilinear group $(n, \mathbb{G}, \mathbb{G}_T, e, g)$, where g is a generator of $\mathbb{G}$ and $n = pq$ is the product of two large primes, and an element h of order q. To construct a NIWI proof, first all secret variables should be committed computing $\mathsf{Com}(X_i, \rho_i) = X_i h^{\rho_i}$ and $\mathsf{Com}(Y_j, \tau_j) = Y_i h^{\tau_j}$. After that, the proof π is computed using a protocol $Proof(ck, E, \{X_i, \rho_i\}, \{Y_j, \tau_j\})$, where E is the equation to be satisfied. Finally, the verifier must check that

$$\prod_{i=1}^{r} e(g_i, \mathsf{Com}(X_i)) \cdot \prod_{i=1}^{r} \prod_{j=1}^{s} e(\mathsf{Com}(X_i), \mathsf{Com}(Y_j))^{\gamma_{ij}} = t_T e(h, \pi).$$

The correctness, soundness and witness indistinguishability of these proof systems are proved in [9].

3 Revocable Attribute-Based Signatures: Protocols and Security

In this section we describe the protocols that form an attribute-based signature scheme, as well as the security properties that must be required to such a scheme. A difference with respect to previous definitions for this primitive (such as the one in [14]) is that we deal explicitly with the identity of the users, because of the revocability property of our scheme. An attribute-based signature is linked to a determined *signing policy* $(\mathcal{P}, \Gamma)$: a set $\mathcal{P}$ of attributes and a monotone increasing family $\Gamma \subset 2^{\mathcal{P}}$ of subsets of $\mathcal{P}$. A valid signature means that a signer possessing all the attributes of some of the subsets in Γ is the author of the signature. The monotonicity property ensures that $A_1 \subset A_2, A_1 \in \Gamma \Rightarrow A_2 \in \Gamma$. The most common and simple example of such a monotone increasing family of subsets is the threshold case: in a (ℓ, n)-threshold signing policy, the set $\mathcal{P}$ contains n attributes, and $\Gamma = \{A \subset \mathcal{P} \ : \ |A| \geq \ell\}$. That is, by verifying a threshold attribute-based signature, the verifier is convinced that the author of the signature holds at least ℓ of the attributes included in the set $\mathcal{P}$.

3.1 Syntactic Definition

A revocable attribute-based signature scheme consists of four probabilistic polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda)$. The setup algorithm takes as input a security parameter λ and outputs some public parameters params, a master secret key msk and a revocation key rk. The public parameters contain the possible universe of attributes $\tilde{\mathcal{P}} = \{\mathsf{at}_1, \ldots, \mathsf{at}_m\}$.
- $\mathsf{KeyGen}(\mathsf{id}, A, \mathsf{msk}, \mathsf{params})$. The key generation algorithm takes as input the master secret key msk, the public parameters params and then an identity id and a set of attributes $A \subset \tilde{\mathcal{P}}$ satisfied by the user with identity id. The output is a private key $\mathsf{sk}_{\mathsf{id},A}$. The master entity may store some information (for example, a table) relating the executions of this protocol with the identities id of the users. We refer to this information as st.
- $\mathsf{Sign}(M, \mathcal{P}, \Gamma, \mathsf{sk}_{\mathsf{id},A}, \mathsf{params})$. The signing algorithm takes as input a message M, a signing policy $(\mathcal{P}, \Gamma)$ where $\mathcal{P} \subset \tilde{\mathcal{P}}$ and $\Gamma \subset 2^{\mathcal{P}}$, a secret key $\mathsf{sk}_{\mathsf{id},A}$ and the public parameters params, and outputs a signature σ.
- $\mathsf{Verify}(\sigma, M, \mathcal{P}, \Gamma, \mathsf{params})$. The verification algorithm takes as input the signature σ, the message M, the signing policy $(\mathcal{P}, \Gamma)$ and the public parameters params, and outputs accept or reject, depending on the validity of the signature.
- $\mathsf{Revoke}(\sigma, \mathsf{rk}, \mathsf{params}, \mathsf{st})$. The revocation algorithm takes as input a signature σ, the revocation key rk, the public parameters params and possibly the information st stored by the master entity during the executions of KeyGen, and outputs an identity id or the special symbol $\perp$.

Of course, the usual properties of correctness for the verification and revocation algorithms must be required. Intuitively, a signature for a signing policy $(\mathcal{P}, \Gamma)$ that is computed by using $\mathsf{sk}_{\mathsf{id},A}$ such that $A \in \Gamma$ must be always accepted by the verification protocol. Analogously, for any signature σ that is accepted by the verification protocol, the revocation algorithm must output the identity of the signer who computed σ.

3.2 Security Definitions

Unforgeability. An attribute-based signature scheme must satisfy the property of existential unforgeability against chosen message and signing policy attacks. Such property is defined by the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{F}$.

Setup. $\mathcal{C}$ runs the setup algorithm and keeps the master secret key msk and the revocation key rk to itself, then gives the public parameters params to $\mathcal{F}$.

Queries. Adaptively, $\mathcal{F}$ can request any queries described below.

- Secret key query: $\mathcal{F}$ requests a private key on an identity id and a set of attributes $B \subset \tilde{\mathcal{P}}$.

– Signature query: $\mathcal{F}$ requests a signature for a message M and a signing policy $(\mathcal{P}, \Gamma)$, where $\mathcal{P} \subset \tilde{\mathcal{P}}$ and $\Gamma \subset 2^{\mathcal{P}}$.
– Revocation query: $\mathcal{F}$ sends a tuple $(M, \sigma, \mathcal{P}, \Gamma)$. If the signature is valid, then $\mathcal{F}$ expects to receive as answer an identity id for the author of the signature σ.

Output. Finally, $\mathcal{F}$ outputs a tuple $(\sigma^*, M^*, \mathcal{P}^*, \Gamma^*)$ and wins the game if (1) the signature is valid, (2) $\mathcal{F}$ has not made any secret key query for a set of attributes $A \subset \tilde{\mathcal{P}}$ such that $A \in \Gamma^*$, and (3) $\mathcal{F}$ has not made any signature query for the tuple $(M^*, \mathcal{P}^*, \Gamma^*)$.

Let $\mathsf{Succ}_{\mathcal{F}}$ be the event that $\mathcal{F}$ wins the above game. The advantage of $\mathcal{F}$ is defined as $\mathrm{Adv}_{\mathcal{F}}^{\text{ABS-EUF}} = Pr(\mathsf{Succ}_{\mathcal{F}})$ where the probability is taken over the coin tosses made by $\mathcal{F}$ and $\mathcal{C}$.

Definition 1. *An attribute-based signature scheme is unforgeable if, for any adversary $\mathcal{F}$ that runs in polynomial time, its advantage $Adv_{\mathcal{F}}^{ABS\text{-}EUF}$ is negligible in the security parameter λ.*

The above definition of unforgeability guarantees collusion resistance: a group of colluding users that pull their secret keys together will not be able to sign messages for a signing policy that none of the attribute sets of these users satisfies. The definition is in the *adaptive* setting where the attacker chooses the target signing policy $(\mathcal{P}^*, \Gamma^*)$ after making some queries. This is in contrast to the *selective* setting where the attacker must choose the target signing policy at the very beginning of the attack.

Non-Linkability and Anonymity. Intuitively, non-linkability means that an observer cannot distinguish if two valid signatures for the same signing policy have been computed by the same user. Non-linkability is defined via the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup. The setup is the same as the setup of the unforgeability game.

Queries. $\mathcal{A}$ can make the same queries as $\mathcal{F}$ in the unforgeability game.

Challenge. $\mathcal{A}$ submits a challenge tuple $(\mathsf{id}_0, M_0, \sigma_0, M_1, \mathcal{P}, \Gamma)$. If some of the following conditions fails, the challenger aborts:

– $\mathcal{A}$ has asked for a secret key for (id_0, A_0) such that $A_0 \cap \mathcal{P} \in \Gamma$,
– $\mathsf{Verify}(\sigma_0, M_0, \mathcal{P}, \Gamma, \mathsf{params}) = \mathsf{accept}$,
– $\mathsf{Revoke}(\sigma_0, \mathsf{rk}, \mathsf{params}) = \mathsf{id}_0$.

Otherwise, the challenger $\mathcal{C}$ recovers the secret key $\mathsf{sk}_{\mathsf{id}_0, A_0}$ that has been delivered to $\mathcal{A}$, chooses at random a different identity $\mathsf{id}_1 \neq \mathsf{id}_0$ and a subset of attributes $A_1 \in \Gamma$ and runs $\mathsf{sk}_{\mathsf{id}_1, A_1} \leftarrow \mathsf{KeyGen}(\mathsf{id}_1, A_1, \mathsf{msk}, \mathsf{params})$. Then $\mathcal{C}$ flips a random coin $b \in \{0, 1\}$ and computes $\sigma_1 \leftarrow \mathsf{Sign}(M_1, \mathcal{P}, \Gamma, \mathsf{sk}_{\mathsf{id}_b, A_b}, \mathsf{params})$. The values $\sigma_1, \mathsf{id}_1, A_1, \mathsf{sk}_{\mathsf{id}_1, A_1}$ are returned to $\mathcal{A}$.

Queries. $\mathcal{A}$ can make more queries, with the restriction that the signature σ_1 cannot be queried to the revocation oracle.

Output. Finally, $\mathcal{A}$ outputs a guess b' of b and wins the game if $b = b'$.

Let $\mathsf{Succ}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the above game. The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_{\mathcal{A}}^{\text{ABS-LIN}} = |2Pr(\mathsf{Succ}_{\mathcal{A}}) - 1|$, where the probability is taken over the coin tosses made by $\mathcal{A}$ and $\mathcal{C}$.

Definition 2. *An attribute-based signature scheme is non-linkable if, for any adversary $\mathcal{A}$ that runs in polynomial time, its advantage $Adv_{\mathcal{A}}^{ABS\text{-}LIN}$ is negligible in the security parameter λ.*

The more standard property of signer's anonymity can be defined in a very similar way. Since in both definitions the adversary can obtain secret keys for all identities of his choice, it is easy to see that non-linkability implies signer's anonymity.

Non-Frameability. Our schemes will enjoy the interesting property of revocability, which means that there exists an authority that can break the anonymity of a signature, when needed. This property brings new possibilities for an adversary to cheat the system, that must be dealt with by our security model. Specifically, we must consider *framing* attacks where an adversary tries to produce a signature that is later revoked to the identity of some honest user. This intuition is formalized by considering the following game, between a challenger $\mathcal{C}$ and a framing attacker $\mathcal{T}$.

Setup. The setup is similar to the setup of the unforgeability game, but now even the revocation key rk is given to the adversary $\mathcal{T}$.

Queries. $\mathcal{A}$ can make the same queries as $\mathcal{F}$ in the unforgeability game. Note that revocation queries make no sense now, since $\mathcal{T}$ knows the revocation key. Let $\mathcal{M} = \{M \text{ s.t. } (M, \mathcal{P}, \Gamma) \text{ is a signing query}\}$ be the set of messages queried to the signing oracle, and let $\mathcal{ID} = \{\mathsf{id} \text{ s.t. } (\mathsf{id}, B) \text{ is a secret key query}\}$ be the set of identities for which $\mathcal{T}$ obtains secret keys.

Output. At some point, $\mathcal{T}$ outputs a tuple $(\sigma^*, M^*, \mathcal{P}^*, \Gamma^*)$ and wins the game if (1) the signature is valid, (2) $M^* \notin \mathcal{M}$, and (3) $\mathsf{Revoke}(\sigma^*, \mathsf{rk}, \mathsf{params}) \notin \mathcal{ID}$.

Let $\mathsf{Succ}_{\mathcal{T}}$ be the event that $\mathcal{T}$ wins the above game. The advantage of $\mathcal{T}$ is defined as $\mathrm{Adv}_{\mathcal{T}}^{\text{ABS-FRA}} = Pr(\mathsf{Succ}_{\mathcal{T}})$ where the probability is taken over the coin tosses made by $\mathcal{T}$ and $\mathcal{C}$.

Definition 3. *A revocable attribute-based signature scheme is non-frameable if, for any adversary $\mathcal{T}$ that runs in polynomial time, its advantage $Adv_{\mathcal{T}}^{ABS\text{-}FRA}$ is negligible in the security parameter λ.*

4 The New Scheme

In this section, we construct our attribute-based signature scheme. We proceed in different steps. First, we construct a linkable scheme which works for threshold signing policies. Then we will introduce some changes in order to achieve non-linkability. The security of the resulting scheme will be proved in the random oracle model. After that, we will modify the scheme to admit more general signing policies. And finally, we will explain how to achieve security in the standard model.

4.1 The Intuition: a Linkable Scheme

Our basic construction is inspired by the ring signature scheme of Shacham-Waters [16].

$\mathsf{Setup}(1^\lambda)$. The setup algorithm first generates a symmetric bilinear group $(n, \mathbb{G}, \mathbb{G}_T, e, g)$ of composite order $n = pq$, where p and q are primes of bit size $\Theta(\lambda)$. Next, it chooses random $w \in \mathbb{G}$, $h \in \mathbb{G}_q$, where $\mathbb{G}_q$ is the subgroup of $\mathbb{G}$ of order q, $s \in \mathbb{Z}_n$ and cryptographic hash functions $H_1, H_2 : \{0,1\}^* \to \mathbb{G}$. It also generates a value $\delta_p \in \mathbb{Z}_n$ such that $\delta_p = 0 \mod q$ and $\delta_p = 1 \mod p$. An automorphic signature scheme is chosen, with public key $\mathsf{pk}_{\mathsf{aut}}$ and secret key $\mathsf{sk}_{\mathsf{aut}}$. Finally, a universe of attributes $\tilde{\mathcal{P}}$ is chosen. Then, the public parameters params, the master secret key msk and the revocation key rk are defined as

$$\mathsf{params} = (n, \mathbb{G}, \mathbb{G}_T, e, g, g_1 = g^s, h, h_1 = h^s, w, H_1, H_2, \mathsf{pk}_{\mathsf{aut}}, \tilde{\mathcal{P}})$$

$$\mathsf{msk} = (s, \mathsf{sk}_{\mathsf{aut}}) \qquad\qquad \mathsf{rk} = \delta_p$$

The master entity can erase the values (p, q, δ_p), because they are not needed to answer key generation queries.

$\mathsf{KeyGen}(\mathsf{id}, A, \mathsf{msk}, \mathsf{params})$. The key generation algorithm takes as input an identity id, a subset of attributes $A \subset \tilde{\mathcal{P}}$ satisfied by id, the master secret key msk and the public parameters params. The master entity chooses a random element $K_{\mathsf{id}} \in \mathbb{G}$ and signs this value with the automorphic signature, obtaining $\sigma_{K_{\mathsf{id}}}$. For each attribute $\mathsf{at}_i \in A$, the algorithm chooses a random $r_i \in \mathbb{Z}_n$ and defines the attribute secret key as $\mathsf{sk}_i = (E_i, G_i) = (H_1(\mathsf{at}_i)^s K_{\mathsf{id}}^{r_i}, g^{r_i})$. Finally, the global secret key is $\mathsf{sk}_{\mathsf{id},A} = (K_{\mathsf{id}}, \sigma_{K_{\mathsf{id}}}, \{\mathsf{sk}_i\}_{\mathsf{at}_i \in A})$. The master entity secretly stores the relation between id and K_{id} in a table st, that can be sent to the revocation judge.

$\mathsf{Sign}(M, \mathcal{P}, \ell, \mathsf{sk}_{\mathsf{id},A}, \mathsf{params})$. The signing algorithm takes as input a message M, a set of attributes $\mathcal{P} \subset \tilde{\mathcal{P}}$, a threshold ℓ, a secret key $\mathsf{sk}_{\mathsf{id},A}$ and the public parameters params. The algorithm selects a minimal authorized set A', this is, a subset of $A \cap \mathcal{P}$ of cardinality exactly ℓ. To generate the signature, it proceeds as follows:

1. First, for each $\mathsf{at}_i \in \mathcal{P}$ it chooses a random $z_i \in \mathbb{Z}_n$ and computes the commitment C_i of f_i and the corresponding proof π_i as
$$C_i = (H_1(\mathsf{at}_i)/w)^{f_i} h^{z_i} \text{ and } \pi_i = ((H_1(\mathsf{at}_i)/w)^{2f_i - 1} h^{z_i})^{z_i}$$
where $f_i = 1$ if $\mathsf{at}_i \in A'$ and $f_i = 0$ otherwise.
2. Then, it computes $H_m = H_2(M, \mathcal{P}, \ell)$. It also chooses a random $t \in \mathbb{Z}_n$ and computes $\sigma_1 = \left(\prod_{\mathsf{at}_i \in A'} E_i \right) H_m^t h_1^z$, $\sigma_2 = g^t$ and $\sigma_3 = \prod_{\mathsf{at}_i \in A'} G_i$, where $z = \sum_{\mathsf{at}_i \in \mathcal{P}} z_i$.
3. Finally, the signature is $\sigma = (\sigma_1, \sigma_2, \sigma_3, \{(C_i, \pi_i)\}_{\mathsf{at}_i \in \mathcal{P}}, K_{\mathsf{id}}, \sigma_{K_{\mathsf{id}}})$.

$\mathsf{Verify}(\sigma, M, \mathcal{P}, \ell, \mathsf{params})$. The verification algorithm takes as input a message M, the signature σ on M, the threshold signing policy $(\mathcal{P}, \ell)$ and the public parameters params. It proceeds as follows:

1. For all $\mathsf{at}_i \in \mathcal{P}$, check if $e(C_i, C_i/(H_1(\mathsf{at}_i)/w)) \stackrel{?}{=} e(h, \pi_i)$.
2. Compute $H_m = H_2(M, \mathcal{P}, \ell)$ and check if $e(\sigma_1, g) \stackrel{?}{=} e(w^\ell \prod_{\mathsf{at}_i \in \mathcal{P}} C_i, g_1) e(H_m, \sigma_2) e(K, \sigma_3)$.
3. Check that $\sigma_{K_{\mathsf{id}}}$ is a valid automorphic signature on K_{id}.
4. Output accept if all the tests are successful, and reject otherwise.

$\mathsf{Revoke}(\sigma, \mathsf{rk}, \mathsf{params}, \mathsf{st})$. Since the value K_{id} is included in the signature σ, the judge only needs to recover the relation $(\mathsf{id}, K_{\mathsf{id}})$ from the secret table st.

4.2 Achieving Non-Linkability

It is easy to see that the scheme described in the previous section works correctly. However, the values σ_3, K_{id} and $\sigma_{K_{\mathsf{id}}}$ allow any verifier to link two signatures issued by the same signer, even if the relation between K_{id} and the identity of the signer is unknown. To solve this drawback, we will use Groth-Sahai proofs to commit to K_{id} and $\sigma_{K_{\mathsf{id}}}$, and we will randomize σ_3.

Let $\mathsf{Com}(K_{\mathsf{id}})$ and $\mathsf{Com}(\sigma_{K_{\mathsf{id}}})$ be the commitments to K_{id} and $\sigma_{K_{\mathsf{id}}}$ respectively. Let π_σ be the NIWI proof of the satisfiability of the second verification equation, and $\pi_{K_{\mathsf{id}}}$ the NIWI proof of the satisfiability of the verification equation of the automorphic signature of K_{id}. We remind the reader that the verification of an automorphic signature is done by evaluating a set of pairing-product equations, so we can build Groth-Sahai proofs of satisfiability for these equations.

In addition, we randomize the value σ_3 by choosing a random $r' \in \mathbb{Z}_n$ and multiplying σ_3 with $g^{r'}$. We will also need to multiply σ_1 with $K_{\mathsf{id}}^{r'}$ in order to satisfy the verification equation. So, we redefine $\sigma_3 = \prod_{\mathsf{at}_i \in A'} G_i g^{r'}$ and $\sigma_1 = \left(\prod_{\mathsf{at}_i \in A'} E_i \right) K_{\mathsf{id}}^{r'} H_m^t h_1^z$.

Now, the signature will be the tuple $\sigma = (\sigma_1, \sigma_2, \sigma_3, \{(C_i, \pi_i)\}_{\mathsf{at}_i \in \mathcal{P}}, \mathsf{Com}(K_{\mathsf{id}}), \mathsf{Com}(\sigma_{K_{\mathsf{id}}}), \pi_{K_{\mathsf{id}}}, \pi_\sigma)$. The verification algorithm $\mathsf{Verify}(\sigma, M, \mathcal{P}, \ell, \mathsf{params})$ proceeds now as follows:

1. For all $\mathsf{at}_i \in \mathcal{P}$, check if $e(C_i, C_i/(H_1(\mathsf{at}_i)/w)) \stackrel{?}{=} e(h, \pi_i)$.
2. Check if $e(\sigma_1, g) \stackrel{?}{=} e(w^l \prod_{\mathsf{at}_i \in \mathcal{P}} C_i, g_1) e(H_m, \sigma_2) e(\mathsf{Com}(K_{\mathsf{id}}), \sigma_3) e(h, \pi_\sigma)$
3. Checks that $\sigma_{K_{\mathsf{id}}}$ is a valid signature on K_{id} using the proof $\pi_{K_{\mathsf{id}}}$ and the commitments $\mathsf{Com}(K_{\mathsf{id}}), \mathsf{Com}(\sigma_{K_{\mathsf{id}}})$.
4. Outputs accept if all the tests are successful, and reject otherwise.

Finally, the revocation algorithm must be modified in the following way.

$\mathsf{Revoke}(\sigma, \mathsf{rk}, \mathsf{params}, \mathsf{st})$. The revocation algorithm takes as input the revocation key $\mathsf{rk} = \delta_p$, a valid signature σ, the public parameters params and the table

st. It computes $\mathsf{Com}(K_{\mathsf{id}})^{\delta_p} = K_{\mathsf{id}}^{\delta_p}$. This value $K_{\mathsf{id}}^{\delta_p}$ can be detected in the secret table st, in order to obtain the identity id of the signer. Note that this process can be made more efficient if a third value $K_{\mathsf{id}}^{\delta_p}$ is added to each entry $(\mathsf{id}, K_{\mathsf{id}})$ in the table st.

Security Analysis Now we prove that the scheme described in this section achieves the properties of non-linkability, non-frameability and unforgeability. The proofs for the first two property are just sketched. The unforgeability proof, which is done in the random oracle model, is detailed in Appendix A.

Theorem 1. *If the subgroup decision assumption holds in $\mathbb{G}$, then our threshold attribute-based signature scheme is non-linkable.*

Proof (sketch). The challenger can use the Setup algorithm to choose the parameters of the security game. As he knows the secret and revocation keys, he can answer all the queries made by the adversary using the algorithms KeyGen, Sign and Revoke.

The advantage of $\mathcal{A}$ is negligible because σ_1, σ_2 and σ_3 are randomized elements. All the commitments and proofs to the attributes $\{(C_i, \pi_i)\}$ and the commitments $\mathsf{Com}(K_{\mathsf{id}}), \mathsf{Com}(\sigma_{K_{\mathsf{id}}})$, and the proofs $\pi_{K_{\mathsf{id}}}, \pi_\sigma$ do not reveal any information about f_i or K_{id} because they are commitments and NIWI proofs. $\square$

It is easy to see that our scheme also enjoys non-linkability (and anonymity) with respect to the subset of attributes employed to compute a signature.

Theorem 2. *Assuming that the underlying automorphic signature scheme is secure, our threshold attribute-based signature scheme is non-frameable.*

Proof (sketch). The challenger can use the knowledge of all the elements in msk and rk, excepting $\mathsf{sk}_{\mathsf{aut}}$, and also its access to a signing oracle for the automorphic signature scheme, to answer the different queries that a framing adversary $\mathcal{T}$ makes.

If $\mathcal{T}$ succeeds in forging a signature σ^* for which $\mathsf{Com}(K^*)^{\delta_p} \neq K_{\mathsf{id}}^{\delta_p}$ for all the values K_{id} that have been generated in the secret key queries, then the values in the forged attribute-based signature can be used to obtain a valid forgery against the automorphic signature scheme. $\square$

Theorem 3. *Suppose that the CDH assumption holds in $\mathbb{G}_p$ and the subgroup decision assumption holds in $\mathbb{G}$. Then, our threshold attribute-based signature scheme is existentially unforgeable under chosen message and signing policy attacks.*

The proof of this theorem can be found in Appendix A.

4.3 Admitting More General Signing Policies

The previous scheme admits only threshold signing policies. A first generalization of such policies are k-multi-threshold policies: the universe of attributes $\tilde{\mathcal{P}}$ is divided into disjoint subsets, $\tilde{\mathcal{P}} = B_1 \cup \ldots \cup B_k$. A $(\ell_1, \ldots, \ell_k)$-multi-threshold policy $(\mathcal{P}, \Gamma)$ in this set is defined by $\Gamma = \{A \subset \mathcal{P} \, : \, |A \cap B_j| \geq \ell_j, \text{ for } j = 1, \ldots, k\}$. Our scheme can be easily adapted to admit such multi-threshold policies, without any increase in the length of the signatures. In the setup phase, instead of choosing a single value w, now k random values $w_1, \ldots, w_k \in \mathbb{G}$ must be chosen and added to params. The commitments will be now $C_i = (H_1(\mathsf{at}_i)/w_j)^{f_i} h^{z_i}$, if $\mathsf{at}_i \in B_j$. The NIWI proofs π_i are changed so that they satisfy the verification equation $e(C_i, C_i/(H_1(\mathsf{at}_i)/w_j)) \stackrel{?}{=} e(h, \pi_i)$. Finally, the value H_m must be defined as $H_m = H_2(M, \mathcal{P}, \ell_1, \ldots, \ell_k)$ and the verification equation that involves all the commitments C_i is now

$$e(\sigma_1, g) \stackrel{?}{=} \left(\prod_{1 \leq j \leq k} e(w_k^{\ell_j} \prod_{\mathsf{at}_i \in B_j} C_i, g_1) \right) \cdot e(H_m, \sigma_2) \cdot e(\mathsf{Com}(K_{\mathsf{id}}), \sigma_3) \cdot e(h, \pi_\sigma)$$

Let us consider now more general signing policies, not necessarily defined by any threshold. We consider $\mathbb{Z}_n$-monotone span programs [11]. A signing policy $(\mathcal{P}, \Gamma)$ is a $\mathbb{Z}_n$-monotone span program if there exist a $m_1 \times m_2$ matrix Ψ with entries in $\mathbb{Z}_n$, being $m_1 \geq |\mathcal{P}|$, and a function $\tau : \{1, \ldots, m_1\} \to \{1, \ldots, |\mathcal{P}|\}$ that associates each row of Ψ to an attribute in $\mathcal{P}$, such that

$$A \in \Gamma \iff \Big(\exists \boldsymbol{\lambda} \in (\mathbb{Z}_n)^{m_1} : \boldsymbol{\lambda}\Psi = (1, 0, \ldots, 0)\Big) \text{ and } \Big(\forall j = 1, \ldots, m_1, \ \mathsf{at}_{\tau(j)} \notin A \Rightarrow \lambda_j = 0\Big)$$

An ℓ-threshold policy can be represented as a $\mathbb{Z}_n$-monotone span program if $\ell < p, \ell < q$, by considering Vandermonde-type matrices. We have to modify the Sign and Verify protocols of our threshold scheme in order to admit $\mathbb{Z}_n$-monotone span program signing policies. The signer has to convince the verifier that he possesses a secret key for a set A of authorized attributes, i.e., that there exists a vector $\boldsymbol{\lambda} \in (\mathbb{Z}_n)^{m_1}$ such that $\boldsymbol{\lambda}\Psi = (1, 0, \ldots, 0)$ and such that $\lambda_j = 0$ for any index $j \in \{1, \ldots, m_1\}$ for which $\mathsf{at}_{\tau(j)} \notin A$. In order to guarantee the anonymity of the attributes, the signer will commit to the components λ_j of this vector $\boldsymbol{\lambda}$. In addition, NIWI proofs will be added in the signature to convince the verifier that the commitments are well-formed.

Namely, the signer will prove for all $\mathsf{at}_i \in \mathcal{P}$ that there exist a value $\tilde{\lambda}_i \neq 0$ and an index $j \in \tau^{-1}(i)$ satisfying the equality $\lambda_j = \tilde{\lambda}_i f_i$. Remember that $f_i = 1$ if $\mathsf{at}_i \in A$ and $f_i = 0$ otherwise, being A the authorized subset of attributes held by the signer. On the one hand, if $f_i = 1$, then $\lambda_j \neq 0$ for some $j \in \tau^{-1}(i)$ (we can assume this without loss of generality; otherwise, the attribute $\mathsf{at}_i \in A$ would be useless, and $A - \{\mathsf{at}_i\} \in \Gamma$). In this case, $\tilde{\lambda}_i = \lambda_j$ satisfies the previous equality. On the other hand, if $f_i = 0$ then $\lambda_j = 0$ for all $j \in \tau^{-1}(i)$, and any value $\tilde{\lambda}_i \in \mathbb{Z}_n^*$ satisfies the desired equality. Note that commitments $\mathsf{Com}(f_i)$ to the values f_i will have to be added to the signature, as well.

To prove that these values $\tilde{\lambda}_i$ are different to 0, the signer will prove that they are invertible. That is, he will prove that there exists $\mu_i \in \mathbb{Z}_n^*$ such that $\mu_i \tilde{\lambda}_i = 1$, for all $\mathsf{at}_i \in \mathcal{P}$. The probability that $\tilde{\lambda}_i$ is different to zero but is not invertible is very small because n is the product of two large primes. Note that it would be more efficient to prove and check that the product of all the values $\tilde{\lambda}_i$ is invertible; this is not possible to do by using Groth-Sahai proofs, because they only apply to quadratic equations.

Summing up, given the monotone span program (Ψ, τ) defining the signing policy $(\mathcal{P}, \Gamma)$, and given the commitments $C_i = (H_1(\mathsf{at}_i))^{f_i} h^{z_i}$ and $\mathsf{Com}(f_i)$, for each $\mathsf{at}_i \in \mathcal{P}$, the signer makes NIWI proofs to convince the verifier that:

1. $\exists \boldsymbol{\lambda} \in (\mathbb{Z}_n)^{m_1}$ such that $\boldsymbol{\lambda}\Psi = (1, 0, \ldots, 0)$. Note that this means m_2 NIWI proofs, one for each component in this vector equality. We denote such proofs as $\{\pi_{\boldsymbol{\lambda},k}\}_{k=1}^{m_2}$.
2. $\exists \tilde{\lambda}_i \in \mathbb{Z}_n$, $\exists j \in \tau^{-1}(i)$ such that $\lambda_j = \tilde{\lambda}_i f_i$ for all $\mathsf{at}_i \in \mathcal{P}$. We denote such proofs as $\{\pi_{\tilde{\lambda}_i}\}_{\mathsf{at}_i \in \mathcal{P}}$, whose global length is linear in m_1.
3. $\exists \mu_i \in \mathbb{Z}_n^*$ such that $\mu_i \tilde{\lambda}_i = 1 \bmod n$, for all $\mathsf{at}_i \in \mathcal{P}$. We denote such proofs as $\{\pi_{\mu_i}\}_{\mathsf{at}_i \in \mathcal{P}}$.
4. The commitment $\mathsf{Com}(f_i)$ and C_i commit to the same value, for all $\mathsf{at}_i \in \mathcal{P}$. We denote such proofs as $\{\pi_{f_i}\}_{\mathsf{at}_i \in \mathcal{P}}$.

The signatures that result from this process have the form

$$\sigma = \Big(\{C_i, \pi_i, \mathsf{Com}(f_i), \mathsf{Com}(\tilde{\lambda}_i), \mathsf{Com}(\lambda_i), \pi_{f_i}, \pi_{\lambda_i}, \pi_{\mu_i}\}_{\mathsf{at}_i \in \mathcal{P}}, \\ \{\pi_{\boldsymbol{\lambda},k}\}_{k=1}^{m_2}, \sigma_1, \sigma_2, \sigma_3, \mathsf{Com}(K_{\mathsf{id}}), \mathsf{Com}(\sigma_{K_{\mathsf{id}}}), \pi_{K_{\mathsf{id}}}, \pi_\sigma \Big)$$

Therefore, the length of the signature depends linearly on the size $m_1 + m_2$ of the monotone span program Ψ. Using analogous arguments to those used for the threshold case, one can prove that the resulting attribute-based signature scheme enjoys the properties of unforgeability and non-linkability.

4.4 Security in the Standard Model

We have proved the security properties of our schemes in the random oracle model, but there are well-known techniques that can be applied to our schemes so that security can be proved in the standard model.

The hash function H_1 is used to transform attributes into elements in $\mathbb{G}$. Since the universe of attributes $\tilde{\mathcal{P}}$ is chosen in the Setup algorithm, a different element $Q_i \in \mathbb{G}$ can be chosen at random and associated to each attribute $\mathsf{at}_i \in \tilde{\mathcal{P}}$. These elements will be included in the public parameters params. In the security proofs, we would define $Q_i = g^{c_i}$ with some probability ρ and $Q_i = g_2^{c_i}$ with probability $1 - \rho$, for some random value $c_i \in \mathbb{Z}_n$ (as it is done in the proof of Lemma 2, in Appendix A).

The other hash function, H_2, is used to transform tuples $(M, \mathcal{P}, \Gamma)$ into elements of $\mathbb{G}$. A well-known solution for this case is to consider a collusion resistant hash function $H : \{0,1\}^* \to \{0,1\}^m$ and elements $v_1, v_2, \ldots, v_m \in \mathbb{G}$, which are included in params. Then the value of $H_2(M, \mathcal{P}, \Gamma)$ is replaced with $\prod_{j=1}^{m} v_j^{H(M,\mathcal{P},\Gamma)_j}$, where $H(M, \mathcal{P}, \Gamma)_j$ denotes the j-th bit of $H(M, \mathcal{P}, \Gamma)$. In the security proof the elements $v_1, v_2, \ldots, v_m$ will be chosen by algorithm $\mathcal{B}$ so that $\mathcal{B}$ knows their discrete logarithm with respect to the base g.

4.5 Prime Order Bilinear Groups

Our scheme uses composite order groups and the security is based (among others) on the subgroup decision assumption. Schemes designed in this setting are considered to be inefficient because the size of the elements in such groups must be very large.

Alternatively, we can consider a prime order symmetric bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, because Groth-Sahai proofs can be implemented there and revocation keys rk can be defined in this case, as well. The underlying hardness assumption is then the *decisional linear assumption*: given $(g^\alpha, g^\beta, g^{r\alpha}, g^{s\beta}, g^t)$ for random $\alpha, \beta, r, s \in \mathbb{Z}_p$ it is hard to tell whether $t = r+s$ or t is random. However, whereas in the composite order group setting both commitments and proofs consist of a single element, in the prime order group setting commitments and proofs consist of multiple elements (1 element for each variable, and 6 or 9 elements for each equation). Anyway, considering that computing a pairing has complexity of $O(n^3)$ in time, and that the size of composite order groups is about ten times larger than the size of prime order groups, the scheme in the prime order group setting might be more efficient. We have described our constructions in the setting of composite order groups, though, because this simplifies notation and understandability in a significant way.

The previous comment also affects the discussion about the $\mathbb{Z}_n$-monotone span programs that we consider in Section 4.3. Usually, monotone span programs are defined over a finite field, whereas we are considering the ring $\mathbb{Z}_n$. Note that n is a product of two large primes, so the ring $\mathbb{Z}_n$ almost behaves as a field. Anyway, if we move to the scenario based on the decisional linear assumption, where the bilinear groups have prime order p, then we can consider the standard notion of $\mathbb{Z}_p$-monotone span programs, for a field $\mathbb{Z}_p$.

References

1. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev and M. Ohkubo. Structure-preserving signatures and commitments to group elements. *Proceedings of Crypto'10*, LNCS **6223**, Springer-Verlag, pp. 209–236 (2010)
2. J. Bethencourt, A. Sahai and B. Waters. Ciphertext-policy attribute-based encryption. *Proceedings of IEEE Symposium on Security and Privacy*, IEEE Society Press, pp. 321–334 (2007)
3. X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. *Proceedings of PKC'07*, LNCS **4450**, Springer-Verlag, pp. 1–15 (2007)

4. N. Chandran, J. Groth and A. Sahai. Ring signatures of sub-linear size without random oracles. *Proceedings of ICALP'07*, LNCS **4596**, Springer-Verlag, pp. 423–434 (2007)
5. G. Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Manuscript available at `http://eprint.iacr.org/2009/320` (2009)
6. V. Goyal, O. Pandey, A. Sahai and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of Computer and Communications Security, CCS'06*, ACM Press, pp. 89–98 (2006)
7. J. Groth. Fully anonymous group signatures without random oracles. *Proceedings of Asiacrypt'07*, LNCS **4833**, Springer-Verlag, pp. 164–180 (2007)
8. J. Groth, R. Ostrovsky and A. Sahai. Perfect non-interactive zero knowledge for NP. *Proceedings of Eurocrypt'06*, LNCS **4004**, Springer-Verlag, pp. 339–358 (2006)
9. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. *Proceedings of Eurocrypt'08*, LNCS **4965**, Springer-Verlag, pp. 415–432 (2008)
10. J. Herranz, F. Laguillaumie and C. Ràfols. Constant size ciphertexts in threshold attribute-based encryption. *Proceedings of PKC'10*, LNCS **6056**, Springer-Verlag, pp. 19–34 (2010)
11. M. Karchmer and A. Wigderson. On span programs. *Proceedings of SCTC'93*, IEEE Computer Society Press, pp. 102–111 (1993).
12. A. Lewko, T. Okamoto, A. Sahai, K. Takashima and B. Waters. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. *Proceedings of Eurocrypt'10*, LNCS **6110**, Springer-Verlag, pp. 62–91 (2010)
13. J. Li, M.H. Au, W. Susilo, D. Xie and K. Ren. Attribute-based signature and its applications. *Proceedings of ASIACCS'10*, ACM Press, pp. 60–69 (2010)
14. H.K. Maji, M. Prabhakaran and M. Rosulek. Attribute-based signatures. To appear in *Proceedings of CT-RSA'11*, available at `http://eprint.iacr.org/2010/595` (2010). A preliminary version is available at `http://eprint.iacr.org/2008/328`
15. A. Sahai and B. Waters. Fuzzy identity-based encryption. *Proceedings of Eurocrypt'05*, LNCS **3494**, Springer-Verlag, pp. 457–473 (2005)
16. H. Shacham and B. Waters. Efficient ring signatures without random oracles. *Proceedings of PKC'07*, LNCS **4450**, Springer-Verlag, pp. 166–180 (2007)
17. S.F. Shahandashti and R. Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. *Proceedings of Africacrypt'09*, LNCS **5580**, Springer-Verlag, pp. 198–216 (2009)

A Proof of Theorem 3

We construct an algorithm $\mathcal{B}$ that solves the CDH problem in $\mathbb{G}_p$ running an adversary $\mathcal{F}$ attacking the unforgeability of our scheme. Note that each proof (C_i, π_i) in a forged signature $(\sigma^*, M^*, \mathcal{P}^*, \ell^*)$ generated by $\mathcal{F}$ must pass the verification equation $e(C_i, C_i/(H(\mathsf{at}_i))/w)) = e(h, \pi_i)$. This implies that C_i has the form $(H_1(\mathsf{at}_i)/w)^{f_i} h^{z_i}$ for some $f_i \in \{0, 1\}$ and $z_i \in \mathbb{Z}_n$. According to the value of $\sum_{\mathsf{at}_i \in \mathcal{P}^*} f_i$, we consider two types of adversaries as follows.

1. A type-1 adversary $\mathcal{F}_1$ is one such that $\sum_{\mathsf{at}_i \in \mathcal{P}^*} f_i \neq \ell^*$, where ℓ^* is the threshold defining the signing policy of the forged signature.
2. A type-2 adversary $\mathcal{F}_2$ is one such that $\sum_{\mathsf{at}_i \in \mathcal{P}^*} f_i = \ell^*$.

For each type of adversary $\mathcal{F}_1$ and $\mathcal{F}_2$, we will construct algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively, to solve the CDH problem in $\mathbb{G}_p$. We do this in the following two lemmas.

Lemma 1. *Assume that the inherent automorphic signature scheme is secure. If there exists a type-1 adversary $\mathcal{F}_1$ against our threshold attribute-based signature scheme, then there exists an algorithm $\mathcal{B}_1$ that solves the CDH problem in $\mathbb{G}_p$.*

Proof. Suppose there exists a type-1 adversary $\mathcal{F}_1$ that breaks unforgeability of our scheme. Let us construct an algorithm $\mathcal{B}_1$ that solves the CDH problem in $\mathbb{G}_p$. $\mathcal{B}_1$ is given the description of the bilinear group $\mathbb{G}$, the factorization p, q of n, which is the order of $\mathbb{G}$, and a random CDH challenge $(g_p, g_p^\alpha, g_p^\beta) \in \mathbb{G}_p^3$, where g_p is a generator of $\mathbb{G}_p$. Its goal is to compute $g_p^{\alpha\beta}$. The algorithm $\mathcal{B}_1$ interacts with $\mathcal{F}_1$ as follows:

Setup. $\mathcal{B}_1$ selects a generator $h \in \mathbb{G}_q$ and chooses random values $r_1 \in \mathbb{Z}_q^*, r_2, r_3 \in \mathbb{Z}_q$. It also chooses the keys $(\mathsf{sk}_{\mathsf{aut}}, \mathsf{pk}_{\mathsf{aut}})$ of an automorphic signature scheme and a universe of attributes $\tilde{\mathcal{P}}$. Next it defines the public parameters as $\mathsf{params} = (n, \mathbb{G}, \mathbb{G}_T, e, g = g_p h^{r_1}, g_1 = g_p^\alpha h^{r_2}, h, h_1 = h^{r_2/r_1}, w = g_p^\beta h^{r_3}, H_1, H_2, \mathsf{pk}_{\mathsf{aut}}, \tilde{\mathcal{P}})$ and gives params to $\mathcal{A}_1$. The public parameters are correctly distributed because $e(g_1, h) = e(g_p^\alpha h^{r_2}, h) = e(h^{r_1}, h^{r_2/r_1}) = e(g_p h^{r_1}, h^{r_2/r_1}) = e(g, h_1)$. This is because $g_p \in \mathbb{G}_p$ and $h \in \mathbb{G}_q$ imply that $e(g_p^\alpha, h) = 1$.

Queries. Adaptively, $\mathcal{F}_1$ can make H_1-hash queries, H_2-hash queries, secret key queries, signature queries or revocation queries at any time. For hash queries, $\mathcal{B}_1$ creates and maintains two lists H_1-list and H_2-list storing the information of all the queries; these lists are consulted before answering any new query, for consistency.

For a H_1-hash query on at_i, $\mathcal{B}_1$ generates a random $c_i \in \mathbb{Z}_n$ and responds with $H_1(\mathsf{at}_i) = g^{c_i}$. For a H_2-hash query on a message M, a set of attributes $\mathcal{P}$ and a threshold ℓ, $\mathcal{B}_1$ generates a random $d \in \mathbb{Z}_n$ and responds with $H_2(M, \mathcal{P}, \ell) = g^d$. For a secret key query for an identity id and attribute set $A \subset \tilde{\mathcal{P}}$, $\mathcal{B}_1$ generates a random $e_j \in \mathbb{Z}_n$, computes $K_{\mathsf{id}} = g^{e_j}$ and the signature $\sigma_{K_{\mathsf{id}}}$ and for each $\mathsf{at}_i \in A$ computes $\mathsf{sk}_i = (g_1^{c_i} K_{\mathsf{id}}^{r_i}, g^{r_i})$, where r_i is chosen at random in $\mathbb{Z}_n$. It responds with $\mathsf{sk}_{\mathsf{id},A} = (K_{\mathsf{id}}, \{\mathsf{sk}_i\}_{\mathsf{at}_i \in A})$. Using these secret keys, $\mathcal{B}_1$ can answer signature queries properly, as well. Since $\mathcal{B}_1$ knows p, q, it can easily answer revocation queries, as well.

Output. Finally, $\mathcal{F}_1$ outputs a signature $(\sigma^*, M^*, \mathcal{P}^*, \ell^*)$, where

$$\sigma^* = (\sigma_1, \sigma_2, \sigma_3, \{(C_i, \pi_i)\}_{\mathsf{at}_i \in \mathcal{P}^*}, \mathsf{Com}(K_{\mathsf{id}^*}), \mathsf{Com}(\sigma_{K_{\mathsf{id}^*}}), \pi_{K_{\mathsf{id}^*}}, \pi_\sigma).$$

If (1) $\mathcal{F}_1$ did request a private key $\mathsf{sk}_{\mathsf{id},A}$ such that $A \cap \mathcal{P}^*$ has cardinality at least ℓ^*, or (2) $\mathcal{F}_1$ did request a signature on the tuple $(M^*, \mathcal{P}^*, \ell^*)$, or (3) if the forged signature is not valid; then $\mathcal{B}_1$ stops the simulation because $\mathcal{F}_1$ has not been successful.

Otherwise, $\mathcal{B}_1$ solves the given instance of the CDH problem as follows: let δ_p be such that $\delta_p = 0 \mod q$ and $\delta_p = 1 \mod p$. We have $u^{\delta_p} = 1$ if, and only if, $u \in \mathbb{G}_q$. We obtain $C_i^{\delta_p} = (H_1(\mathsf{at}_i)^{\delta_p} / w^{\delta_p})^{f_i} = (g_p^{c_i} / g_p^\beta)^{f_i}$ for all

$\mathsf{at}_i \in \mathcal{P}^*$, and so $C^{\delta_p} = \prod_{\mathsf{at}_i \in \mathcal{P}^*} C_i^{\delta_p} = g_p^c/(g_p^\beta)^f$, where $c = \sum_{\mathsf{at}_i \in \mathcal{P}^*} c_i f_i$ and $f = \sum_{\mathsf{at}_i \in \mathcal{P}^*} f_i$. From the second verification equation of the scheme (see Section 4.2), we obtain $e(g_p, \sigma_1^{\delta_p}) = e(g_p^\alpha, (g_p^\beta)^{\ell^*} g_p^c/(g_p^\beta)^f) e(\sigma_2^{\delta_p}, g_p^d) e(\sigma_3^{\delta_p}, g_p^e)$, where $H_2(M^*, \mathcal{P}^*, \ell^*)^{\delta_p} = g_p^d$ and $K^{\delta_p} = g_p^e$. This equation comes from the fact that $\mathsf{Com}(K_{\mathsf{id}^*})^{\delta_p} = K_{\mathsf{id}^*}^{\delta_p}$ and $e(h, \pi_\sigma)^{\delta_p} = 1$. By rewriting this equation, we have $e(g_p^\alpha, g_p^\beta)^{\ell^* - f} = e(g_p, \sigma_1^{\delta_p} (\sigma_2^{\delta_p})^{-d} (\sigma_3^{\delta_p})^{-e} (g_p^\alpha)^{-c})$. $\mathcal{B}_1$ recovers from H_1-list the values c_i corresponding to all the attributes in $\mathcal{P}^*$ and the value d corresponding to the H_2-query $(M^*, \mathcal{P}^*, \ell^*)$. It also recovers the value e corresponding to K_{id^*}. As the automorphic signature scheme is secure, we can be sure that the value of K_{id^*} used in the forgery comes from a query, so it is known to $\mathcal{B}_1$. Finally, $\mathcal{B}_1$ recovers $\{f_i\}_{i \in \mathcal{P}^*}$ using that $C_i^{\delta_p} = 1$ if, and only if, $f_i = 0$. By assumption $f = \sum_{\mathsf{at}_i \in \mathcal{P}^*} f_i \neq \ell^*$, and so the value $(\ell^* - f)^{-1} \mod p$ exists. Therefore, $\mathcal{B}_1$ can solve the CDH problem as follows:

$$g_p^{\alpha\beta} = \left[(\sigma_1 \sigma_2^{-d} \sigma_3^{-e})^{\delta_p} g_p^{-\alpha c}\right]^{\frac{1}{\ell^* - f}} .$$

□

Lemma 2. *If there exists a type-2 adversary $\mathcal{F}_2$ against our threshold attribute-based signature scheme, then there exists an algorithm $\mathcal{B}_2$ that solves the CDH problem in $\mathbb{G}_p$.*

Proof. Suppose there exists a type-2 adversary $\mathcal{F}_2$ that breaks the unforgeability of our scheme. We construct an algorithm $\mathcal{B}_2$ that solves the CDH problem in $\mathbb{G}_p$. $\mathcal{B}_2$ is given the description of the bilinear group $\mathbb{G}$, the factorization p, q of n, which is the order of $\mathbb{G}$, and a random CDH challenge $(g_p, g_p^\alpha, g_p^\beta) \in \mathbb{G}_p^3$, where g_p is a generator of $\mathbb{G}_p$. Its goal is to compute $g_p^{\alpha\beta}$. The algorithm $\mathcal{B}_2$ interacts with $\mathcal{F}_2$ as follows:

Setup. $\mathcal{B}_2$ selects a generator $h \in \mathbb{G}_q$ and chooses random values $r_1 \in \mathbb{Z}_q^*, r_2, r_3, r_4 \in \mathbb{Z}_q, r_5 \in \mathbb{Z}_p$. It also chooses keys $(\mathsf{sk}_{\mathsf{aut}}, \mathsf{pk}_{\mathsf{aut}})$ of an automorphic signature scheme and a universe of attributes $\tilde{\mathcal{P}}$. Next it sets the public parameters $\mathsf{params} = (n, \mathbb{G}, \mathbb{G}_T, e, g = g_p h^{r_1}, g_1 = g_p^\alpha h^{r_2}, h, h_1 = h^{r_2/r_1}, w = g_p^{r_5} h^{r_3}, H_1, H_2, \mathsf{pk}_{\mathsf{aut}}, \tilde{\mathcal{P}})$, defines $g_2 = g_p^\beta h^{r_4}$ and gives params to $\mathcal{F}_2$. The public parameters are correctly distributed because $e(g_1, h) = e(g_p^\alpha h^{r_2}, h) = e(h^{r_1}, h^{r_2/r_1}) = e(g_p h^{r_1}, h^{r_2/r_1}) = e(g, h_1)$. These equalities hold because $g_p \in \mathbb{G}_p$ and $h \in \mathbb{G}_q$ imply that $e(g_p^\alpha, h) = 1$. With these parameters, $s = \log_g g_1$ is implicitly defined.

Queries. Adaptively, $\mathcal{F}_1$ can make H_1-hash queries, H_2-hash queries, secret key queries, signature queries or revocation queries at any time. $\mathcal{B}_2$ creates and maintains lists H_1-list, H_2-list and K-list, for consistency. Again, since $\mathcal{B}_2$ knows p, q, it can easily answer revocation queries.

For a H_1-hash query on at_i, $\mathcal{B}_2$ responds as follows: if at_i was in a previous H_1-hash query, it recovers $(\mathsf{at}_i, H_1\text{-}\mathsf{coin}_i, c_i)$ from H_1-list; otherwise, it generates a random $H_1\text{-}\mathsf{coin}_i \in \{0, 1\}$ so that $\Pr[H_1\text{-}\mathsf{coin}_i = 1] = \rho_1$, for ρ_1 to be determined later. It generates a random $c_i \in \mathbb{Z}_n^*$ and stores $(\mathsf{at}_i, H_1\text{-}\mathsf{coin}_i, c_i)$ in H_1-list. If

H_1-$\mathsf{coin}_i = 0$, then it responds with $H_1(\mathsf{at}_i) = g^{c_i}$; otherwise, it responds with $H_1(\mathsf{at}_i) = g_2^{c_i}$.

For a H_2-hash query on a tuple $(M, \mathcal{P}, \ell)$, $\mathcal{B}_2$ responds as follows: if $(M, \mathcal{P}, \ell)$ already exists in H_2-list, $\mathcal{B}_2$ recovers $(M, \mathcal{P}, \ell, d)$ from its H_2-list; otherwise, it generates a random $d \in \mathbb{Z}_n$, stores $(M, \mathcal{P}, \ell, d)$ in H_2-list and responds with $H_2(M, \mathcal{P}, \ell) = g^d$.

For a secret key query for an identity id and a set of attributes $A \subset \tilde{\mathcal{P}}$, $\mathcal{B}_2$ responds as follows: it generates a random K_{id}-$\mathsf{coin} \in \{0,1\}$ so that $\Pr[K_{\mathsf{id}}$-$\mathsf{coin} = 1] = \rho_2$, for ρ_2 to be determined later. If K_{id}-$\mathsf{coin} = 0$, $\mathcal{B}_2$ generates a random $e \in \mathbb{Z}_n$ and sets $e' = 0$; otherwise, it generates two random elements $e, e' \in \mathbb{Z}_n^*$. It defines $K_{\mathsf{id}} = g^e g_1^{e'}$ and stores $(K_{\mathsf{id}}\text{-}\mathsf{coin}, e, e', K_{\mathsf{id}})$ in K-list. Next, for each attribute $\mathsf{at}_i \in A$, it recovers $(\mathsf{at}_i, H_1\text{-}\mathsf{coin}_i, c_i)$ from H_1-list.

- If H_1-$\mathsf{coin}_i = 0$, then it generates a random $r_i \in \mathbb{Z}_n$ and defines $\mathsf{sk}_i = (g_1^{c_i} K_{\mathsf{id}}^{r_i}, g^{r_i})$.
- If H_1-$\mathsf{coin}_i = 1$ and K-$\mathsf{coin} = 1$, it generates a random $r_i \in \mathbb{Z}_n$ and sets $\mathsf{sk}_i = ((g_2^{c_i})^{-\frac{e}{e'}} (g^e g_1^{e'})^{r_i}, g^{r_i} g_2^{-\frac{c_i}{e'}})$. The secret key is correctly distributed (we denote by β^* the value $\log_g g_2$) because: $E_i = (g_2^{c_i})^{-\frac{e}{e'}} (g^e g_1^{e'})^{r_i} = (g_2^{c_i})^s (g^e g_1^{e'})^{(r_i - \frac{c_i}{e'}\beta^*)} = H(\mathsf{at}_i)^s K_{\mathsf{id}}^{r_i'}$ and, on the other hand, $G_i = g^{r_i} g_2^{-\frac{c_i}{e'}} = g^{(r_i - \frac{c_i}{e'}\beta^*)} = g^{r_i'}$.
- Otherwise, if H_1-$\mathsf{coin}_i = 1$ and K_{id}-$\mathsf{coin}_j = 0$, then $\mathcal{B}_2$ cannot create sk_i.

If $\mathcal{B}_2$ can create sk_i for all $\mathsf{at}_i \in A$, then it uses $\mathsf{sk}_{\mathsf{aut}}$ to compute an automorphic signature $\sigma_{K_{\mathsf{id}}}$ on K_{id} and responds to $\mathcal{F}_2$'s query with $\mathsf{sk}_{\mathsf{id},A} = (K_{\mathsf{id}}, \sigma_{K_{\mathsf{id}}}, \{\mathsf{sk}_i\}_{\mathsf{at}_i \in A})$. Otherwise, it aborts.

For a signature query for the tuple $(M, \mathcal{P}, \ell)$, $\mathcal{B}_2$ acts as follows. It recovers $(M, \mathcal{P}, \ell, d)$ from H_2-list and recovers $(H(\mathsf{at}_i), H_1\text{-}\mathsf{coin}_i, c_i)$ from H_1-list, for all the attributes $\mathsf{at}_i \in \mathcal{P}$. $\mathcal{B}_2$ creates a K value by generating at random $e, e' \in \mathbb{Z}_n$ and computing $K = g^e g_1^{e'}$. With this value of K, $\mathcal{B}_2$ can create secret keys for any attribute at, so a secret key $\mathsf{sk}_{\mathcal{P}}$ can be generated and used to sign the message M, by following the algorithm Sign of the scheme. Note that the adversary $\mathcal{F}_2$ cannot distinguish which value of K has been used in the signature (due to the anonymity properties of the scheme).

Output. Finally, $\mathcal{F}_2$ outputs a signature $(\sigma^*, M^*, \mathcal{P}^*, \ell^*)$ where

$$\sigma^* = (\sigma_1, \sigma_2, \sigma_3, \{(C_i, \pi_i)\}_{\mathsf{at}_i \in \mathcal{P}^*}, \mathsf{Com}(K_{\mathsf{id}^*}), \mathsf{Com}(\sigma_{K_{\mathsf{id}^*}}), \pi_{K_{\mathsf{id}^*}}, \pi_\sigma).$$

If (1) $\mathcal{F}_2$ did request a private key $\mathsf{sk}_{\mathsf{id},A}$ such that $A \cap \mathcal{P}^*$ has cardinality at least ℓ^*, or (2) $\mathcal{F}_2$ did request a signature for the tuple $(M^*, \mathcal{P}^*, \ell^*)$, or (3) if the forged signature is not valid; then $\mathcal{B}_2$ stops the simulation because $\mathcal{F}_2$ has not been successful.

Otherwise, $\mathcal{B}_2$ solves the given instance of the CDH problem as follows: let δ_p be such that $\delta_p = 0 \mod q$ and $\delta_p = 1 \mod p$. Computing $C_i^{\delta_p}$ for all $\mathsf{at}_i \in \mathcal{P}^*$, $\mathcal{B}_2$ recovers $\{f_i\}_{\mathsf{at}_i \in \mathcal{P}^*}$, so it recovers the subset A' of attributes that has been used in the forged signature. Next it recovers $(M^*, \mathcal{P}^*, \ell^*, d^*)$ from H_2-list and $(\mathsf{at}_i, H_1\text{-}\mathsf{coin}_i, c_i)$ from H_1-list, for every attribute $\mathsf{at}_i \in A'$. On the other hand, it computes $\mathsf{Com}(K_{\mathsf{id}^*})^{\delta_p} = K_{\mathsf{id}^*}^{\delta_p}$. Since the automorphic signature scheme is secure, we can be sure that the value of K_{id^*} used in the forgery has been obtained in a secret key query, so K_{id^*} is known to $\mathcal{B}_2$, that can recover $(K_{\mathsf{id}^*}\text{-}\mathsf{coin}, e, e', K_{\mathsf{id}^*})$ from K-list. If $K_{\mathsf{id}^*}\text{-}\mathsf{coin} = 1$, then $\mathcal{B}_2$ aborts. Otherwise, let B_0 be the set of indices $i \in A'$ such that $H_1\text{-}\mathsf{coin}_i = 0$ and let B_1 be the set of indices i such that $H_1\text{-}\mathsf{coin}_i = 1$. We have $A' = B_0 \cup B_1$. Due to the non-malleability properties of the employed hash functions, σ_1 must be of the form

$$\begin{aligned}
\sigma_1 &= \left(\prod_{\mathsf{at}_i \in A'} (H(\mathsf{at}_i))^s \right) K_{\mathsf{id}^*}^r H_m^t h_1^z \\
&= \left(\prod_{i \in B_0} (H(\mathsf{at}_i))^s \prod_{i \in B_1} (H(\mathsf{at}_i))^s \right) (g^e)^r (g^{d^*})^t h_1^z \\
&= \left(\prod_{i \in B_0} (g^{c_i})^s \prod_{i \in B_1} (g_2^{c_i})^s \right) \sigma_3^e \sigma_2^{d^*} h_1^z \\
&= g_1^{\sum_{i \in B_0} c_i} (g_2^s)^{\sum_{i \in B_1} c_i} \sigma_3^e \sigma_2^{d^*} h_1^z
\end{aligned}$$

Let $c_{B_0} = \sum_{i \in B_0} c_i$ and $c_{B_1} = \sum_{i \in B_1} c_i$. If $c_{B_1} \bmod p = 0$, $\mathcal{B}_2$ aborts because it cannot solve the CDH problem. Otherwise, we have $g_2^s = (\sigma_1 \sigma_2^{-d^*} \sigma_3^{-e} g_1^{-c_{B_0}} h_1^{-z})^{1/c_{B_1}}$. We also have $(g_2^s)^{\delta_p} = g_p^{\alpha\beta}$, that comes from the fact that $g_p^\alpha = g_1^{\delta_p} = (g^s)^{\delta_p}$. Using that $\mathsf{Com}(\sigma_3)^{\delta_p} = \sigma_3^{\delta_p}$, we have that $\mathcal{B}_2$ can solve the CDH problem as follows:

$$g_p^{\alpha\beta} = (g_2^s)^{\delta_p} = \left[\sigma_1^{\delta_p} (\sigma_2^{\delta_p})^{-d^*} (\sigma_3^{\delta_p})^{-e} (g_p^\alpha)^{-c_{B_0}} \right]^{\frac{1}{c_{B_1}}}$$

Analysis. Let abort be the event that $\mathcal{B}_2$ aborts during the simulation and let forge be the event that $\mathcal{F}_2$ produces a valid forgery according to the definition of the unforgeability game. We have

$$\begin{aligned}
\mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{CDH}} &\geq \Pr[\mathsf{forge} \wedge \neg\mathsf{abort}] \\
&= \Pr[\mathsf{forge} | \neg\mathsf{abort}] \Pr[\neg\mathsf{abort}] \\
&= \mathrm{Adv}_{\mathcal{F}_2}^{\mathrm{ABS}} \Pr[\neg\mathsf{abort}]
\end{aligned}$$

The last equality comes from the fact that, if abort does not occur, then $\mathcal{B}_2$ simulates perfectly the environment of $\mathcal{F}_2$. Let abort_E be the event that $\mathcal{B}_2$ aborts at a secret key query, let abort_K be the event that $K_{\mathsf{id}^*}\text{-}\mathsf{coin} = 1$ when $\mathcal{F}_2$ outputs a forgery, and let abort_C be the event that $c_{B_1} \bmod p = 0$.

$$
\begin{aligned}
\Pr[\neg\mathsf{abort}] &= \Pr[\neg\mathsf{abort}_E \wedge \neg\mathsf{abort}_K \wedge \neg\mathsf{abort}_C] \\
&= \Pr[\neg\mathsf{abort}_E] \Pr[\neg\mathsf{abort}_K \wedge \neg\mathsf{abort}_C | \neg\mathsf{abort}_E] \\
&= \Pr[\neg\mathsf{abort}_E] \Pr[\neg\mathsf{abort}_K | \neg\mathsf{abort}_E] \Pr[\neg\mathsf{abort}_C | \neg\mathsf{abort}_E] \\
&\geq \Big[((1-\rho_2)(1-\rho_1)^{q_E} + \rho_2)\Big] \cdot \Big[(1-\rho_2)(1-\rho_1)^{q_E}\Big] \cdot \\
&\cdot \Big[(1-1/p)(1-(1-\rho_1)^{q_E})\rho_2\Big]
\end{aligned}
$$

The third equality follows from the fact that the events abort_K and abort_C are independent. We note that $F(\rho_1, \rho_2) = ((1-\rho_2)(1-\rho_1)^{q_E} + \rho_2)(1-\rho_2)(1-\rho_1)^{q_E}(1-1/p)(1-(1-\rho_1)^{q_E})\rho_2$ is greater than 0 except when $\rho_1 = 0, \rho_1 = 1, \rho_2 = 0$ or $\rho_2 = 1$. Therefore, by choosing appropriate values for ρ_1 and ρ_2, we obtain

$$
\mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{CDH}} \geq \mathrm{Adv}_{\mathcal{F}_2}^{\mathrm{ABS}} \cdot \Omega(1)
$$

$\square$