BACHELOR'S DEGREE THESIS

Degree in Mathematics

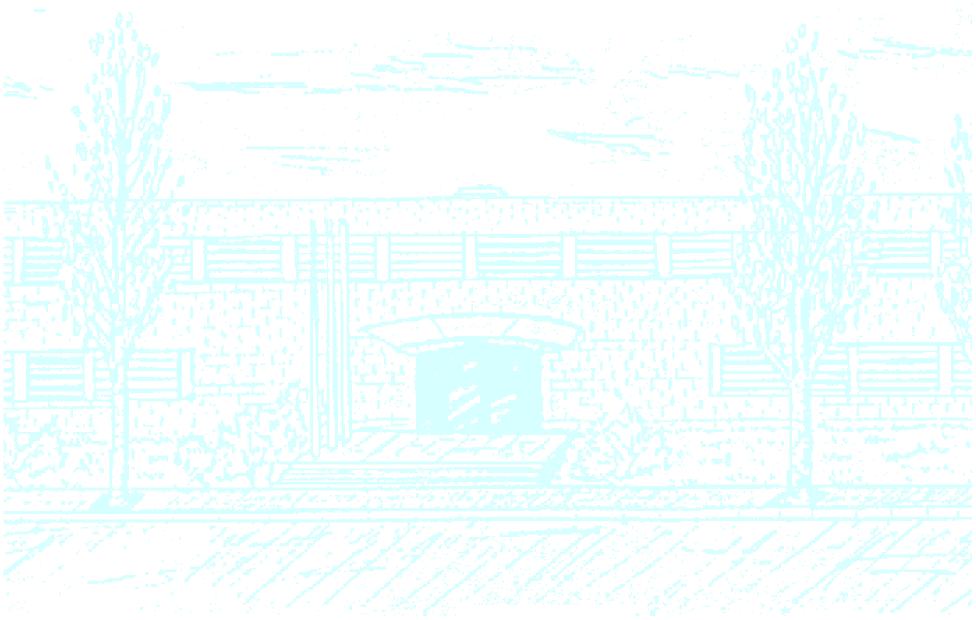
Title: Revocation of Users in RSA Attribute-Based Signatures

Author: María Rosa Fueyo Pestaña

Advisor: Javier Herranz Sotoca

Department: Matemàtica Aplicada IV

Academic year: 2013/2014



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Bachelor's Degree Thesis

Revocation of Users in RSA Attribute-Based Signatures

María Rosa Fueyo Pestaña

Advisor: Javier Herranz Sotoca

Matemàtica Aplicada IV

Abstract

Keywords: Attribute-based signatures, RSA, privacy, unforgeability, revocation, accumulators, Zero-Knowledge Proof.

MSC2000: 92A60, 92A62

An attribute-based signature with respect to a signing policy chosen by the signer, convinces the verifier that the signer sustains a subset of attributes satisfying that signing policy. The verifier must not obtain any other information about the identity of the signer or the attributes he holds. This type of signatures have a lot of applications in real life scenarios that demand both authentication and privacy properties. The ability of revoking users that have misbehaved or lost their attributes, so they can not compute more valid signatures, is very desirable for real life applications of attribute-based signatures.

In this dissertation, the main goal consists in incorporating revocation into an already existing RSA attribute-based signature that currently does not contain revocation. So as to achieve this objective a protocol has been developed. The basic notion this protocol relies on is the concept of dynamic universal accumulators, which is a scheme that would allow to commit a set of values into an accumulator, in this particular case the list of revoked users. Furthermore, this particular scheme admits to efficiently compute a non-membership witness for any value that has not been accumulated, so a user with a valid signature would be able to prove that he has not been revoked. It must be noted that this protocol is built in an anonymous way so as to the user does not need to reveal his identity to proof that he is not in the revocation list.

Finally, the previous protocol is incorporated into an existing RSA attribute-based signature, and the resulting signature is analyzed in terms of efficiency. The length of the information added to the signatures in order to admit the revocation property is independent of the signing policy, as opposed to the length of the original RSA attribute-based signature. Therefore, the conclusion of the work is that incorporating the revocation property in this particular case may be relatively efficient, specially when the considered signing policies contain many attributes.

Notation

$\mathbb{Z}$	Integer numbers
$N = PQ$	RSA modulus
$\phi(\cdot)$	Euler Function
$\mathbb{Z}_N*$	Set of integers less than N and relative prime to N
$neg(\cdot)$	Negligible function
S	Probability space
$x \leftarrow_R S$	Chosen at random according to S
$\mathsf{PK}\cdot$	Zero-Knowledge Proof of Knowledge
$\mathcal{P}$	Set of Attributes
$(\mathcal{P}, \Gamma)$	Signing Policy

Contents

Chapter 1. Introduction 1

Chapter 2. Preliminaries 3
1. Number-Theoretic assumptions 3
2. Proofs of Knowledge 4
3. Attribute-Based Signatures 6

Chapter 3. Universal Accumulators 9
1. Universal Accumulator 9
2. Dynamic Universal Accumulator 12

Chapter 4. Efficient Proof that a Committed Value was not Accumulated 15
1. Formulation of the protocol 15
2. Detailed analysis of the protocol 16

Chapter 5. Incorporating Revocation of Users into Attribute-Based Signatures 21
1. Existing Attribute-Based Scheme for a Threshold Signing Policy 21
2. Incorporating Revocation into the Signature Scheme 23
3. Efficiency 24

Chapter 6. Conclusions 27

References 29

Chapter 1
Introduction

Nowadays, privacy issues are a constant in our life. With the widespread of computer technologies, privacy problems arise everyday wherever personally identifiable information or other sensitive information, such as financial information or medical records, is used and stored. The work in cryptography is key to face these issues.

Attribute-based cryptography has come into sight in the last years as a very powerful paradigm [**18, 17, 19**]. An attribute-based signature can only be carried out by users who hold a subset of attributes that satisfy some policy. The main aspect of this type of signatures is that a successful execution should leak no information about the identity of the user or the attributes he holds, besides the fact that these attributes fulfill the given policy.

Attribute-based signatures were introduced explicitly in the first version of [**16**]. In an attribute-based signature scheme, each user receives from a master entity a secret key which depends on the attributes that he holds. Afterwards, a user can select a signing policy (a 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. As well as the general applications of any attribute-based cryptosystem such as private access control, this particular type of signature has many applications in specific scenarios where both authentication and privacy properties are desired, such as anonymous polls or the leakage of secrets.

Including revocation of users into the signature would be essential in order to apply it into real-life scenarios where users want to preserve a certain level of privacy. In situations where a user misbehaves or loses his attributes, it is necessary to have the option of revoking these users from the system. Implementing revocation into attribute-based signatures is not trivial at all because of the privacy property.

Incorporating revocation into an already existing RSA attribute-based signature is the main aim of this dissertation. Furthermore, the efficiency of the resulting signature scheme, admitting revocation, will be compared with the efficiency of the original signature. In order to do all that a number of papers on the topic have been read through to understand the concepts necessary to achieve this goal.

In Chapter 2, the underlying mathematics concepts and assumptions that intervene in the development of attribute based signatures will be explained. Furthermore, a detailed description of the algorithms that form an attribute-based signature are developed. Moreover, the main security properties and its relation to other properties of attribute-based signatures are laid out. Additionally, the concept of zero-knowledge proof of knowledge was introduced, which will be essential to understand the following chapters and how the privacy property is achieved.

In Chapter 3, the concept of universal accumulator is introduced. This concept will be instrumental to ensure privacy when the protocol for user revocation is designed. In our particular case, the definition of universal accumulator was amplified into the notion of dynamic universal accumulator, thus enabling adding and deleting users into the list of revoked users.

The protocol that will be used to implement revocation, and that was sketched originally in [**2**], is described and analyzed in detail in Chapter 4. Furthermore, the proofs of the soundness and zero-knowledge proof of knowledge properties are verified.

Finally, in Chapter 5, the protocol described in Chapter 4 for the revocation of users is incorporated into the RSA attribute-based signature scheme proposed in [**1**]. An efficiency analysis of the resulting attribute-based signature scheme with revocation is carried out, in order to know how expensive the addition of the revocation property will result.

Chapter 2
Preliminaries

1. Number-Theoretic assumptions

The strong RSA assumption was independently introduced by Barić and Pfitzmann [**13**] and by Fujisaki and Okamoto [**12**]. It strengthens the widely accepted RSA assumption that finding e^{th}-roots modulo N for any $e > 1$ is hard. The formal definition can be seen below:

Definition 1. (*Strong RSA Problem*) Given an RSA modulus $N = PQ$ and a random $x \leftarrow_R \mathbb{Z}_N^*$, the strong RSA problem consists of finding $e > 1$ and $y \in \mathbb{Z}_N^*$, such that $y^e = x \bmod N$

Assumption 1. (*The Strong RSA Assumption*) The Strong RSA Assumption states that the probability that any algorithm A solves the Strong RSA problem in polynomial time is negligible in λ, the length in bits of the RSA modulus. This means that the probability decreases, as λ increases, faster than the inverse of any polynomial. Formally, for any probabilistic polynomial time algorithm A,

$$\mathsf{Pr}\begin{bmatrix} N \leftarrow G(1^\lambda), x \leftarrow_R \mathbb{Z}_N, (y,e) \leftarrow A(N,x) : \\ y^e = x (\bmod N) \wedge 1 < e < N \end{bmatrix} = neg(\lambda)$$

where $G(1^\lambda)$ is an algorithm that generates a RSA modulus of size λ, and $neg(\lambda)$ is a negligible function.

Lemma 1. *For any integer N, given integers $u, v \in \mathbb{Z}_N^*$ and $a, b \in \mathbb{Z}$ such that $u^a = v^b \bmod N$ and $\gcd(a,b) = 1$, one can efficiently compute $x \in \mathbb{Z}_N^*$ such that $x^a = v \bmod N$.*

Proof. Since $\gcd(a,b) = 1$, one can find $c, d \in \mathbb{Z}$ using the extended Euclidean algorithm, such that $bd = 1 + ac$. Let $x = (u^d v^{-c} \bmod N)$, then

$$x^a = u^{ad} v^{-ac} = (u^a)^d v^{-ac} = (v^b)^d v^{-ac} = v (\bmod N)$$

□

Assumption 2. (*Small order assumption*) Let A be any probabilistic polynomial-time algorithm and $G(1^\lambda)$ an algorithm that generates a RSA modulus of size λ. Suppose A outputs $b \in \mathbb{Z}_N^*$ and a number σ. The probability that $b \neq 1$, $b^\sigma = 1$

and $b^2 \neq 1$ is negligible. This means that elements of relatively small known order should be hard to find, except possibly for order 2.

Definition 2. (*Decisional Diffie-Hellman Problem in QR_N, with known factorization*)

An algorithm A solves the Decisional Diffie-Hellman problem in QR_N, with known factorization, if it is able to distinguish between the two probability distributions $(N, P, Q, g, g^x \bmod N, g^y \bmod N, g^{xy} \bmod N)$ and $(N, P, Q, g, g^x \bmod N, g^y \bmod N, g^z \bmod N)$, where $(P, Q, N, g) \leftarrow G(1^\lambda)$ is an algorithm that generates a RSA modulus and $x, y, z \leftarrow_R \mathbb{Z}_{pq}$.

Assumption 3. (*Decisional Diffie-Hellman Assumption*) The DDH Assumption in QR_N, with known factorization, states that the success probability of any such algorithm A is negligible in λ. Formally, for any algorithm A running in polynomial time, we have that the *advantage*

$$\left| \Pr\begin{bmatrix} 1 \leftarrow A(N, P, Q, g, g^x \bmod N, g^y \bmod N, g^{xy} \bmod N); \\ (P, Q, N, g) \leftarrow G(1^\lambda); x, y \leftarrow_R \mathbb{Z}_{pq}. \end{bmatrix}(\lambda) - \Pr\begin{bmatrix} 1 \leftarrow A(N, P, Q, g, g^x \bmod N, g^y \bmod N, g^z \bmod N); \\ (P, Q, N, g) \leftarrow G(1^\lambda); x, y, z \leftarrow_R \mathbb{Z}_{pq}. \end{bmatrix}(\lambda) \right|$$

is negligible in the security parameter λ.

This is proved in the full version of [**14**]: the Decisional Diffie-Hellman problem in QR_N, with known factorization, is equivalent to the Decisional Diffie-Hellman problem in a cyclic subgroup of QR_N of either prime order p or prime order q. The Decisional Diffie-Hellman problem in a cyclic group of big prime order is considered to be computationally hard, and therefore the Decisional Diffie-Hellman Assumption in QR_N, with known factorization, makes perfect sense.

2. Proofs of Knowledge

In cryptography, a proof of knowledge is an interactive proof in which the *prover* succeeds 'convincing' a *verifier* that it knows something, for instance, a solution of an equation. The trivial solution consists in the *prover* sending to the *verifier* what he knows, and the *verifier* authenticates it. But in some cases, the *prover* wants to keep his information private, and just wants to prove the fact that he knows it; this method receives the name of zero-knowledge proof of knowledge.
A so called zero-knowledge proof of knowledge allows a prover to demonstrate the knowledge of a secret with respect to some public information such that no other information is revealed in the process.
Some examples of zero-knowledge proofs of knowledge are:

- Given two graphs G, H, the *prover* wants to prove to the verifier that G is isomorphic to H, without revealing the isomorphism.
- Given a cyclic group $\mathbb{G} = \langle g \rangle$ and $y \in \mathbb{G}$, the *prover* wants to prove that he knows $x \in \mathbb{Z}$ such that $g^x = y$, which is the discrete logarithm of y in basis g, without revealing x.

- Given a public key pk, the *prover* wants to prove that he knows the matching secret key sk, without revealing it. This is known as identification protocol.

2.1. Properties of a Zero-Knowledge Proof of Knowledge.

For any given zero-knowledge proof of knowledge, it satisfies the following properties. Let L be a language in nondeterministic polynomial-time, and given $\alpha \in L$, let $W(\alpha)$ be the set of witnesses of the fact that $\alpha \in L$. The relation R can be defined as $R = \{(\alpha, \omega) \; : \; \alpha \in L, \omega \in W(\alpha)\}$.
Given L and α which are public. The secret input for the *prover* may be the witness ω. The properties are:

- **Completeness:** if the *prover* knows ω such that $(\alpha, \omega) \in R$, then the *verifier* always accepts the *prover*'s proof.
- **Proof of Knowledge:** if the *prover*'s proofs for α are accepted with probability ϵ, then it is possible to extract a witness ω such that $(\alpha, \omega) \in R$ with probability $\geq \epsilon$, given oracle access to the *prover.*
- **Zero-Knowledge:** if the *prover* knows ω such that $(\alpha, \omega) \in R$, then even a malicious *verifier* V' obtains no new information on ω from the execution of the protocol. This means that for any such malicious *verifier*, there exists a Simulator algorithm $\mathcal{S}$ such that:
$$\mathcal{S}(\alpha) \;\approx\; \text{Outputs}[P(\alpha, \omega) \leftrightarrow V'(\alpha)].$$

The description of a zero-knowledge proof of knowledge for the first two examples can be seen below:

- **A Zero-Knowledge Proof of Knowledge Protocol for Graph Isomorphism**
L_{G_0} are the graphs isomorphic to G_0. A graph $G_1 \in L_{G_0}$ if and only if there exists an isomorphism π such that $\pi(G_1) = G_0$. Therefore, $R = \{(G_1, \pi) \; : \; G_1 \in L_{G_0}, \pi(G_1) = G_0\}$. Suppose the *prover* knows π such that $\pi(G_1) = G_0$.
  (1) The *prover* chooses isomorphism ρ at random and sends $H = \rho(G_0)$ to V.
  (2) The *verifier* chooses random bit $b \in \{0, 1\}$ and sends b to the *prover.*
  (3) If $b = 0$, P replies $\psi = \rho$. If $b = 1$, the *prover* replies $\psi = \rho \circ \pi$.
  (4) The *verifier* outputs 1 if and only if $\psi(G_b) = H$.

  If this process is repeated n times in parallel, the cheating probability of a dishonest *prover* is 2^{-n}. This process fulfills the subsequent properties:
  - **Completeness:** trivial.
  - **Proof of Knowledge:** extractor runs the *prover* until step 3, with $b = 0$, and obtains ψ_0. Then rewinds back to step 2, chooses $b = 1$ and lets the *prover* output ψ_1. The extracted witness $\pi = \psi_0^{-1} \circ \psi_1$ satisfies $\pi(G_1) = G_0$.
  - **Zero-Knowledge:** a transcript (H, b, ψ) of the protocol between the *prover* and a malicious *verifier* can be simulated by $\mathcal{S}$ as follows:
    (1) choose a random permutation ψ,

(2) choose bit $b \in \{0,1\}$ with the same distribution as the malicious *verifier* does,
(3) compute $H = \psi(G_b)$.

- **A Zero Knowledge Proof of Knowledge Protocol for Discrete Logarithm**
  $L_{(\mathbb{G},g)}$ are the elements in the cyclic group $\mathbb{G} = \langle g \rangle$. A witness for $y \in G$ is $x \in \mathbb{Z}$ such that $g^x = y$. Therefore, $R = \{(y,x) \ : \ y \in \mathbb{G}, g^x = y\}$. $\mathbb{G}$ has public prime order p. Suppose the *prover* knows $x \in \mathbb{Z}_p$ such that $g^x = y$.

  (1) The *prover* chooses $r \in_R \mathbb{Z}_p^*$ at random and sends $R = g^r$ to the *verifier.*
  (2) The *verifier* chooses $h \in_R \mathbb{Z}_p$ at random and sends h to the *prover.*
  (3) The *prover* computes $s = r + x \cdot h \bmod p$ and sends s to the *verifier.*
  (4) The *verifier* outputs 1 if and only if $g^s = R \cdot y^h$.

  The previous protocol satisfies this properties:

  – **Completeness:** trivial.
  – **Proof of Knowledge:** the extractor runs the *prover* until step 3, with random h, and obtains s. Then rewinds back to step 2, chooses a different $h' \neq h$ and lets the *prover* output s'. The extracted witness $x = \frac{s-s'}{h-h'} \bmod p$ satisfies $g^x = y$.
  – **Zero-Knowledge:** a transcript (R, h, s) of the protocol between the *prover* and a malicious *verifier* can be simulated by $\mathcal{S}$ as follows:
    (1) choose at random $s \in \mathbb{Z}_p$,
    (2) choose $h \in \mathbb{Z}_p$ with the same distribution as the malicious *verifier* does,
    (3) compute $R = g^s \cdot y^{-h}$.

3. Attribute-Based Signatures

This section focuses on describing the concept and protocols of attribute-based signatures. These particular protocols were developed in [**1**] over the protocols of [**15**] in order to deal explicitly with the identity of users. 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 implies 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 $S_1 \subset S_2, S_1 \in \Gamma \ \Rightarrow S_2 \in \Gamma$. A simple example of such a monotone increasing family of subsets is the threshold case. Given a (ℓ, n)-threshold signing policy, with a set $\mathcal{P}$ which contains n attributes, and $\Gamma = \{S \subset \mathcal{P} \ : \ |S| \geq \ell\}$, a verifier authenticates a threshold attribute-based signature if he 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.

An 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 the initial public parameters pms and the master secret key msk for the master entity. Within the public parameters appear the possible universe of attributes $\tilde{\mathcal{P}} = \{\mathsf{at}_1, \dots, \mathsf{at}_n\}$.
- $\mathsf{KeyGen}(\mathsf{id}, S, \mathsf{msk}, \mathsf{pms})$. The key generation algorithm takes as input the master secret key msk, the public parameters pms, furthermore, an identity id that satisfies a set of attributes $S \subset \tilde{\mathcal{P}}$ is required. The output is a private key $\mathsf{sk}_{\mathsf{id},S}$.
- $\mathsf{Sign}(m, \mathcal{P}, \Gamma, \mathsf{sk}_{\mathsf{id},S}, \mathsf{pms})$. 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 $\mathsf{sk}_{\mathsf{id},S}$ and the public parameters pms, and outputs a signature σ.
- $\mathsf{Verify}(\sigma, m, \mathcal{P}, \Gamma, \mathsf{pms})$. The verification algorithm takes as input the signature σ, the message m, the signing policy $(\mathcal{P}, \Gamma)$ and the public parameters pms. The outputs are 1 if the signature is accepted or 0 if it is rejected. signature.

Such a scheme fulfills the correctness' property, if for a signature and for a signing policy $(\mathcal{P}, \Gamma)$ that is computed by using $\mathsf{sk}_{\mathsf{id},S}$ such that $S \in \Gamma$, the signature is always accepted by the verification protocol.

3.2. Security Definitions.

Privacy. The privacy property entails that given a valid signature, nobody can obtain any information about the real author of the signature. That is to say, given two pairs (id_0, S_0) and (id_1, S_1), with $S_0, S_1 \subset \mathcal{P}^*$, and a valid signature $\sigma \leftarrow \mathsf{Sign}(m, \mathcal{P}, \Gamma, \mathsf{sk}_{\mathsf{id}_b,S_b}, \mathsf{pms})$ for a signing policy Γ such that $S_0, S_1 \in \Gamma$, nobody would be able to guess the bit b with probability significantly bigger than 1/2. The privacy property is formally defined via the following experiments $\mathbf{Exp}^{\mathsf{priv}}_{b,\mathcal{B}}(\lambda)$, for $b = 0, 1$, involving an adversary $\mathcal{B}$.

```
$\underline{\mathbf{Exp}^{\mathsf{priv}}_{b,\mathcal{B}}(\lambda)}$
    $(\mathsf{pms}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(m, \mathcal{P}, \Gamma, \mathsf{id}_0, S_0, \mathsf{sk}_{\mathsf{id}_0,S_0}, \mathsf{id}_1, S_1, \mathsf{sk}_{\mathsf{id}_1,S_1}, st_1) \leftarrow \mathcal{B}(\mathsf{pms}, \mathsf{msk})$
    Verify that $\mathsf{sk}_{id_i,S_i}$ is a valid secret key for $S_i$, for $i = 0, 1$
    Verify that $S_0 \cap \mathcal{P} \in \Gamma$ and $S_1 \cap \mathcal{P} \in \Gamma$
    $\sigma^* \leftarrow \mathsf{Sign}(m, \mathcal{P}, \Gamma, \mathsf{sk}_{\mathsf{id}_b,S_b}, \mathsf{pms})$
    $b' \leftarrow \mathcal{B}(\sigma^*, \mathsf{pms}, \mathsf{msk}, st_1)$
    Output $b'$
```

The advantage of $\mathcal{B}$ in breaking the privacy property is defined as

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{B}}(\lambda) = \left| \mathsf{Pr}[\mathbf{Exp}^{\mathsf{priv}}_{0,\mathcal{B}}(\lambda) = 1] - \mathsf{Pr}[\mathbf{Exp}^{\mathsf{priv}}_{1,\mathcal{B}}(\lambda) = 1] \right| .$$

Definition 3. An attribute-based signature scheme is private if, for any adversary $\mathcal{B}$ that runs in polynomial time, the advantage $\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{B}}(\lambda)$ is negligible in the security parameter λ.

Seeing that the adversary $\mathcal{B}$ can obtain the master secret key, other properties of signatures are implied by the privacy property, such as anonymity and unlinkability. The anonymity property means that given a valid signature, identifying the actual signer is computationally hard and the unlinkability property implies that deciding whether two different valid signatures were computed by the same user is computationally hard.

Unforgeability. An attribute-based signature scheme must fulfill the property of existential unforgeability against chosen message and signing policy attacks. Such property is defined by the following experiment $\mathbf{Exp}_{\mathcal{F}}^{\mathsf{unf}}(\lambda)$ involving an adversary $\mathcal{F}$.

```
$\underline{\mathbf{Exp}_{\mathcal{F}}^{\mathsf{unf}}(\lambda)}$
    $(\mathsf{pms}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\sigma^*, m^*, \mathcal{P}^*, \Gamma^*) \leftarrow \mathcal{F}^{\mathsf{KeyGen}(\cdot,\mathsf{msk},\mathsf{pms}),\ \mathsf{Sign}(\cdot,\mathsf{pms})}(\mathsf{pms})$
    Output 1 if the three following statements are true:
        (i) $\mathsf{Verify}(\sigma^*, m^*, \mathcal{P}^*, \Gamma^*, \mathsf{pms})$ returns 1;
        (ii) $\mathcal{F}$ has not made any secret key query $(\mathsf{id}, S)$ such that $S \cap \mathcal{P}^* \in \Gamma^*$;
        (iii) $(m^*, \mathcal{P}^*, \Gamma^*, \sigma^*)$ is not the result of any signature query from $\mathcal{F}$.
    Otherwise, output 0
```

The advantage of $\mathcal{F}$ in breaking the unforgeability of the scheme is defined as $\mathsf{Adv}_{\mathcal{F}}^{\mathsf{unf}}(\lambda) = \Pr[\mathbf{Exp}_{\mathcal{F}}^{\mathsf{unf}}(\lambda) = 1]$. We stress that $\mathcal{F}$ is allowed to make adaptive queries for secret keys of pairs (id, S) of his choice, and adaptive signing queries for tuples $(m, \mathcal{P}, \Gamma)$ of his choice, where $\Gamma \subset 2^{\mathcal{P}}$. The last kind of queries are answered by choosing a random subset $S \subset \mathcal{P}$ with $S \in \Gamma$, and then by running $\mathsf{sk}_{\mathsf{id},S} \leftarrow \mathsf{KeyGen}(\mathsf{id}, S, \mathsf{msk}, \mathsf{pms})$ and $\sigma \leftarrow \mathsf{Sign}(m, \mathcal{P}, \Gamma, \mathsf{sk}_{\mathsf{id},S}, \mathsf{pms})$.

Definition 4. An attribute-based signature scheme is unforgeable if, for any adversary $\mathcal{F}$ that runs in polynomial time, the advantage $\mathsf{Adv}_{\mathcal{F}}^{\mathsf{unf}}(\lambda)$ is negligible in the security parameter λ.

As well as in the privacy definition, the unforgeability definition implies another signature's property such as the collusion resistance property. A group of colluding users (even if it is comprised of all the users) that pool together their secret keys, will not be able to sign messages for a signing policy that none of the attribute sets of these users satisfies.

Chapter 3
Universal Accumulators

The strategy employed to incorporate revocation of users in the signature scheme employs a cryptographic scheme called universal dynamic accumulator. The construction used corresponds to the one defined by Li, Li and Xue [**2**] which is build upon the one of Camenisch and Lysyanskaya [**4**]. Subsequently, the definition of dynamic universal accumulators and the construction used are explained. In the first place, the definition of universal accumulators will be introduced, as the dynamic universal accumulator is developed from this notion.

1. Universal Accumulator

1.1. Definition.

Let λ be a security parameter, for a family of input $\{\mathcal{X}_\lambda\}$ a secure universal accumulator is a family of functions $\{\mathcal{F}_\lambda\}$ that satisfy the following properties:

- **Efficient Generation:** A random function f of $\mathcal{F}_\lambda$ is produced on input 1^λ by an efficient probabilistic polynomial time algorithm called G. Furthermore, auxiliary information about f represented as aux_f is obtained with G.
- **Efficient Evaluation**: $\forall f \in \mathcal{F}_\lambda$, f is a polynomial time function. The inputs are $(g, x) \in \mathcal{G}_f \times \mathcal{X}_\lambda$, where $\mathcal{X}_\lambda$ is the input domain for the elements that will be accumulated and $\mathcal{G}_f$ is the input domain for f. The output value is $h \in \mathcal{G}_f$.
- **Quasi-Commutative:** $\forall f \in \mathcal{F}_\lambda$, $\forall g \in \mathcal{G}_f$, and $\forall x_1, x_2 \in \mathcal{X}_\lambda$, $f(f(g, x_1), x_2) = f(f(g, x_2), x_1)$. If $X = \{x_1, ..., x_m\} \subset \mathcal{X}_\lambda$, $f(f(...(g, x_1), ...), x_m)$ is denoted as $f(g, X)$.
- **Membership Witness:** $\forall f \in \mathcal{F}_\lambda$, a membership verification function called ρ_1 exists. w_1 is called a membership witness if $\rho_1(c, x, w_1) = 1$, where $c \in \mathcal{G}_f$ and $x \in \mathcal{X}_\lambda$.
- **Nonmembership Witness:** $\forall f \in \mathcal{F}_\lambda$, a nonmembership verification function ρ_2 exists. w_2 is called a nonmembership witness if $\rho_2(c, x, w_2) = 1$, where

$c \in \mathcal{G}_f$ and $x \in \mathcal{X}_\lambda$.

- **Security:** A dynamic universal accumulator scheme is secure if, for all probabilistic polynomial-time adversary A_λ,

$$Pr\begin{bmatrix} f \leftarrow G(1^\lambda); g \leftarrow_R \mathcal{G}_f; \\ (x, w_1, w_2, X) \leftarrow A_\lambda(f, \mathcal{G}_f, g); \\ x \in \mathcal{X}_\lambda; X \subset \mathcal{X}_\lambda; \\ \rho_1(f(g,X), x, w_1) = 1; \rho_2(f(g,X), x, w_2) = 1 \end{bmatrix} = neg(\lambda)$$

  That is, given any set $X \in \mathcal{X}_\lambda$, it is infeasible to find $x \in X$ with a valid nonmembership witness or in the other case, find $x \in \mathcal{X}_\lambda \backslash X$ with a valid membership witness.

1.2. Construction.

Let λ be a security parameter. For two integers γ_1, γ_2 which depend on λ and which will be specified later, Δ is the set of integers in the interval $[2^{\gamma_1} - 2^{\gamma_2} + 1, 2^{\gamma_1} + 2^{\gamma_2} - 1]$. Let $\mathcal{X}_\lambda$ denote the set of all primes in Δ. The input domain for the elements accumulated is $\mathcal{X}_\lambda$. The construction built has the following steps:

- The generation algorithm G takes 1^λ as input, the output is a random modulus N of length λ that is a safe prime, $N = PQ$, where $P = 2p+1, Q = 2q+1$, each one of the same length and are also prime numbers.
- f_N is the respective function for modulus N. The factorization of N is the auxiliary information aux_f for f_N. The input domain $\mathcal{G}_f$ for f_N is defined as $\mathcal{G}_f = \{g \in QR_N : g \neq 1\}$ where QR_N is the set of quadratic residues module N.
- For $f = f_N$, $f(g,x) = g^x \bmod N$.
- For $f = f_N$, the membership verification function ρ_1 is defined as $\rho_1(c,x,w) = 1 \Leftrightarrow w^x = c$, where $w \in \mathcal{G}_f$ is the original membership witness for x.
- For $f = f_N$, the nonmembership verification function ρ_2 is defined as $\rho_2(c,x,a,d) = 1 \Leftrightarrow c^a = d^x g \bmod N$, where $(a,d) \in \Delta \times \mathcal{G}_f$ is the nonmembership witness for x.

1.2.1. *Computing witness without aux_f.*

Suppose $X = \{x_1, \cdots, x_m\}$ is a subset of $\mathcal{X}_\lambda$ and g is a random value in QR_N. u is defined as $\prod_{i=1}^m x_i$ and $f(g,X) = g^u \bmod N$. The membership witness for any $x \in X$ is $c_x = g^{u/x} \bmod N$ and to verify the witness, one checks that $x \in \mathcal{X}_\lambda$ and $(c_x)^x = c \bmod N$.
In the case that $x \in \mathcal{X}_\lambda \backslash X$, because $x, x_1, \cdots, x_m$ are distinct prime numbers, $\gcd(x,u) = 1$ and the values $a \in \Delta$ and $b \in \mathbb{Z}$ satisfy that $au + bx = 1$. Using the Euclid algorithm a' and b' are find such that $a'u + b'x = 1$. Since x is a positive integer in Δ, an integer k that satisfies $a' + kx \in \Delta$ can be found. It can be noted that $(a'+kx)u + (b'-ku)x = 1$, thus $a = a' + kx$ and $b = b' - ku$. The

nonmembership witness for x is (a, d) where $d = g^{-b} \bmod N$. In order to verify the witness, it is checked that $x \in \mathcal{X}_\lambda$, $a \in \Delta$, and $c^a = d^x g \bmod N$.

1.2.2. *Computing witness with aux_f.*

The membership witness and nonmembership witness can be calculated efficiently given the auxiliary information aux_f. An existing trusted master entity who knows aux_f, keeps the set X, and has already computed the accumulator $c = f(g, X)$, this master entity is able to compute the nonmembership witness for any $x \in \mathcal{X}_\lambda$ with one short modular exponentiation.
For $x \in X$, the master entity first checks whether $x \in X$, then computes $a = x^{-1} \bmod \phi(N)$, and finally computes $c_x = c^a \bmod N$. The membership witness for x is c_x, it can be observed that $(c_x)^x = (c^a)^x = c^{x^{-1} x \bmod \phi(N)} = c (\bmod N)$.
For $x \in \mathcal{X}_\lambda \backslash X$, let $u' = u \bmod \phi N$, the master entity first checks if $\gcd(x, u') = 1$.

- If $\gcd(x, u') = 1$, the master entity finds a and b such that $au' + bx = 1$, and sets the nonmembership witness for x as $(a, g^{-b} \bmod N)$. The nonmembership witness is legitimate because $c^a = (g^u)^a = (g^{u'})^a = g^{u'a} = g^{1-bx} = g^{-bx} g = d^x g (\bmod N)$.
- If $\gcd(x, u') \neq 1$, the master entity finds a and b such that $au + bx = 1$, then computes $b' = b \bmod \phi(N)$, and sets the nonmembership witness for x as $(a, g^{-b'} \bmod N)$. The nonmembership witness is legitimate because $c^a = g^{ua} = g^{1-bx} = g^{-bx} g = (g^{-b'})^x g = d^x g (\bmod N)$.

It must be reminded that in some scenarios, computing witness using auxiliary information may not be recommended, since the auxiliary information can be used to prove arbitrary statements. Only in the case the party that computes the accumulator is trusted, it is adequate to give up the auxiliary information.

Theorem 2. *Under the strong RSA assumption, the aforementioned construction is a secure universal accumulator.*

Proof. The strong RSA assumption says that given a RSA modulus N and a random value $g \leftarrow_R QR_N$, it is computationally infeasible to find x and y such that $x > 1$ and $y^x = g$.
Daresay there is a polynomial time adversary $\mathcal{A}$, which on input N and $g \in QR_N$, outputs $c_x \in \mathcal{G}_f$, $d \in \mathcal{G}_f$, $x \in \mathcal{X}_\lambda$, $a \in \Delta$ and $X = \{x_1, \cdots, x_m\} \subset \mathcal{X}_\lambda$, where $c = g^{x_1, \cdots, x_m} \bmod N$, $(c_x)^x = c \bmod N$ and $c^a = d^x g \bmod N$. From here, an algorithm $\mathcal{B}$ can break the strong RSA assumption invoking $\mathcal{A}$. There are two cases:

- In the first case, assume $x \in X$ and u represents $\prod_{i=1}^m x_i$. The adversary can compute u, a, d and x, such that $c = g^u \bmod N$ and $c^a = d^x g \bmod N$. Because $x \in X$, $x | u$ and $\gcd(au - 1, x) = 1$. By Lemma 1, y can be found such that $y^x = g$.
- In the second case, assume $x \notin X$ and u represents $\prod_{i=1}^m x_i$. The adversary can compute u, c_x and x, such such $c = g^u \bmod N$ and $(c_x)^x = c \bmod N$ which implicates that $(c_x)^x = g^u \bmod N$. Since $x_1, \cdots, x_m$ are all prime and $x \notin X$, $\gcd(x, u) = 1$ and by Lemma 1 y can be efficiently found such that $y^x = g$.

An efficient algorithm $\mathcal{B}$ that breaks the strong RSA assumption can be constructed as follows. Given a RSA modulus N and $g \leftarrow_R QR_N$, $\mathcal{B}$ invokes $\mathcal{A}$ with inputs N and g which obtains the outputs c_x, d, x, a, X from $\mathcal{A}$. $\mathcal{B}$ can efficiently compute y from c_x, d, x, a, X such that $y^x = d$, which contradicts the strong RSA assumption.

□

Corollary 3. *In the construction mentioned above, for any $f \in \mathcal{F}_\lambda$ and any given set $X \subset \mathcal{X}_\lambda$, it is computationally infeasible to find $x \in X$ with a valid nonmembership witness.*

2. Dynamic Universal Accumulator

2.1. Definition.

Dynamic universal accumulators were proposed so as to dynamically add and delete elements. The construction is based in the one of universal accumulators with some modifications that can be seen below:

- **Efficient Update of Accumulator:** There is an efficient algorithm D that given $c = f(g, X)$, if $\hat{x} \notin X$, thus $D(c, \hat{x}) = \hat{c}$ such that $\hat{c} = f(g, X \cup \{\hat{x}\})$, in the other case, $\hat{x} \in X$, then $D(aux_f, c, \hat{x}) = \hat{c}$ such that $\hat{c} = f(g, X \backslash \{\hat{x}\})$.

- **Efficient Update of Membership Witness:** c and $\hat{c}$ are the original and updated accumulators, and $\hat{x}$ is the new updated element. An efficient algorithm called W_1 satisfies that, if $x \neq \hat{x}, x \in X$, and $rho_1(c, x, w) = 1$, then $W_1(w, c, \hat{c}, x, \hat{x}) = \hat{w}$ such $\rho_1(\hat{c}, x, \hat{w}) = 1$.

- **Efficient Update of Nonmembership Witness:**c and $\hat{c}$ are the original and updated accumulators, and $\hat{x}$ is the new updated element. An efficient algorithm called W_2 satisfies that, if $x \neq \hat{x}$,$x \notin X$, and $\rho_2(c, x, w) = 1$, then $W_2(w, c, \hat{c}, x, \hat{x}) = \hat{w}$ such $\rho_2(\hat{c}, x, \hat{w}) = 1$.

- **Security:** A dynamic universal accumulator scheme is secure if, for all probabilistic polynomial-time adversary A_λ,

$$\Pr\left[\begin{array}{c} f \leftarrow G(1^\lambda); g \leftarrow_R \mathcal{G}_f; \\ (x, w_1, w_2, X) \leftarrow A_\lambda(f, \mathcal{G}_f, g) \leftrightarrow M(f, aux_f, g) \rightarrow (X, c); \\ x \in \mathcal{X}_\lambda; X \subset \mathcal{X}_\lambda; \rho_1(c, x, w_1) = 1; \rho_2(c, x, w_2) = 1 \end{array}\right] = neg(\lambda)$$

  where M is an interactive Turing machine whose inputs are (f, aux_f, g). M keeps a list of values X which is initially empty. g is set to be the initial accumulator c. M answers two types of messages:
  - (add, x): Adds $x \in \mathcal{X}_\lambda$ to the set X, runs D which modifies c and then updates$c$.
  - $(delete, x)$: Checks that $x \in \mathcal{X}_\lambda$, then deletes it from X, updates c by running D.

  The output of M consists of the current values of X and c.

2.2. Construction.

In the case of the dynamic universal accumulators some additional functionalities have been built on the existing universal accumulator construction (see Section 1.2).

- Update of the accumulator: If a value $\hat{x}$ wants to be added to the accumulator, it can be computed as $\hat{c} = c^x \bmod N$. On the other hand, if a value $\hat{x}$ wants to be deleted from the accumulator,it will be computed as $\hat{c} = D(\phi(N), c, \hat{x}) = c^{\hat{x}^{-1} \bmod \phi(N)} \bmod N$, where $\phi(N)$ is the auxiliary information.
- Update of membership witness: w is considered the original membership witness of x. c and $\hat{c}$ are the original and new accumulators. The two operations consists in:
  - **Addition:** Once $\hat{x}$ has been added, the new membership is computed as $\hat{w} = f(w, \hat{x}) = w^{\hat{x}} \bmod N$.
  - **Deletion:** If $\hat{x} \neq x$ has been deleted, the new membership witness $\hat{w}$ is computed choosing an algorithm W_1 who chooses two integer values a and b that satisfy $ax + b\hat{x} = 1$ and so $\hat{w}$ is computed as $\hat{w} = w^b \hat{c}^a \bmod N$.

$$\hat{w}^x = (w^b \hat{c}^a)^x = ((w^b \hat{c}^a)^{x\hat{x}})^{1/\hat{x}} = (c^{b\hat{x}} c^{ax})^{1/\hat{x}} = \hat{c} \bmod N$$

- Update of Nonmembership Witness: Let (a, d) be the original nonmembership witness of x.
  - **Addition:** If $\hat{x} \neq x$ has been added, given $x, \hat{x}, c, \hat{c}$ that verify $\hat{c} = c^{\hat{x}} \bmod N$. The algorithm W_2 is used to compute the new nonmembership witness $(\hat{a}, \hat{d})$. In first place, two integers $\hat{a}_0$ and r_0 that verify $\hat{a}_0\hat{x}+r_0x = 1$ are chosen. Then, the former equation is multiplied by a in both sides, and $\hat{a}_0 a\hat{x} + r_0 ax = a$ is obtained. After that, W_2 computes $\hat{a} = \hat{a}_0 a \bmod x$, and finds $r \in \mathbb{Z}$ such that $\hat{a}\hat{x} = a + rx$ where $\hat{a} \in \Delta$. Finally W_2 computes $\hat{d} = dc^r \bmod N$. It can be verified that $\rho_2(\hat{c}, x, \hat{a}, \hat{d}) = 1$ or $\hat{c}^{\hat{a}} = \hat{d}^x g$ holds.

$$\hat{c}^{\hat{a}} = c^{\hat{a}\hat{x}} = c^{a+rx} = c^{rx} c^a = c^{rx} d^x g = (dc^r)^x g = \hat{d}^x g \bmod N$$

  - **Deletion:** If $\hat{x}$ has been deleted, given $x, \hat{x}, c, \hat{c}$ that verify $c = \hat{c}^{\hat{x}} \bmod N$. The new nonmembership witness $(\hat{a}, \hat{d})$ is obtained through the algorithm W_2. W_2 chooses an integer r that satisfies $a\hat{x} - rx \in \Delta$. If $\hat{a}, \hat{d}$ are $\hat{a} = a\hat{x} - rx$ and $\hat{d} = d\hat{c}^{-r} \bmod N$. It can be verified that $\rho_2(\hat{c}, x, \hat{a}, \hat{d}) = 1$ or $\hat{c}^{\hat{a}} = \hat{d}^x g$ holds.

$$\hat{c}^{\hat{a}} = c^{a\hat{x}-rx} = c^a \hat{c}^{-rx} = d^x g \hat{c}^{-rx} = (d(\hat{c})^{-r})^x g = \hat{d}^x g \bmod N$$

  Several values can be added or deleted at the same time with this algorithm, $\hat{x}$ only needs to be the product of the values that want to be added or deleted.

Theorem 4. *Under the strong RSA assumption, the dynamic universal accumulator construction is secure.*

Proof. A dynamic universal accumulator is a secure construction against an adaptive adversary if the underlying universal accumulator is secure. This can be demonstrated using the reduction argument. If an adversary $\mathcal{A}$ breaks the security property of the dynamic universal accumulator, another adversary $\mathcal{B}$ can be built that breaks the security property of the universal accumulator by invoking $\mathcal{A}$. On input $(f, \mathcal{G}_f, g)$, $\mathcal{B}$ gives these values to $\mathcal{A}$. $\mathcal{B}$ is chosen to act as the manager M, hence $\mathcal{A}$ can interact with it to update the elements:

- If $\mathcal{A}$ sends an (add,x) query, $\mathcal{B}$ inserts x into X and computes $c = f(g, X)$.
- If $\mathcal{A}$ sends a (delete,x) query, $\mathcal{B}$ removes x from X and computes $c = f(g, X)$.

Finally $\mathcal{A}$ outputs an element $x \in \mathcal{X}_\lambda$with a valid membership witness w_1 and a valid nonmembership witness w_2, and $\mathcal{B}$ outputs (x, X, w_1, w_2), seemingly breaking the security property of the universal accumulator. $\square$

Chapter 4
Efficient Proof that a Committed Value was not Accumulated

In Chapter 3, the main concepts of universal accumulators were explained, in this section a protocol using accumulators will be build in order to certificate revocation of users in an anonymous setting. Since the anonymity of the users is required, one can not just simply show his serial number to verify that he is not in the revocation list. The basic notion is that the user commits his value or serial number in his signature or certificate (in a private way) and proves that this value has not been accumulated in the revocation list. The protocol used has been proposed in Section 5 of [**2**].

1. Formulation of the protocol

The commitment scheme is originally from [**12**], but in this case the main goal is to implement it in the attribute-based signature scheme of [**1**]. The main parameters of the scheme are N, g, h where N is a RSA modulus of lenght λ, $g \in QR_N$, $h \in QR_N$ and $r \leftarrow_R \mathbb{Z}_N$. To commit a value x belonging to $\Delta = [2^{\gamma_1} - 2^{\gamma_2} + 1, 2^{\gamma_1} + 2^{\gamma_2} - 1]$ for certain parameters γ_1, γ_2 which depend on λ, the commitment is computed as $commitment(x, r) = g^x h^r \bmod N$

Aside from the parameters of commitment scheme, the protocol requires an element $\tilde{h}$ in QR_N such that $log_{\tilde{g}}\tilde{h}$ is unknown to the prover, where $\tilde{g}$ and N are the parameters of the universal accumulators in the previous Chapter 3. The basic inputs of the protocol are $\tilde{B}$, N, g, h, $\tilde{g}$, $\tilde{h}$, and c. The additional inputs of the prover are x, r, a, d such that $B = g^x h^r \bmod N$ and $c^a = d^x g \bmod N$, where the first equation depicts that x is the committed value of B and the second equation shows that x is not accumulated in c.

The protocol to prove knowledge of a nonmembership witness has six steps, which are shown below.

Protocol $\mathsf{PK}\{(x, r, a, d) : B = g^x h^r \wedge c^a = d^x g \wedge x \in \Delta \wedge a \in \Delta\}$

- *Step 1:* The prover selects uniformly at random the values w, r_x, r_a, r_w, r_z and r_f of length λ. Afterwards, the prover computes the following values $\bmod N$: $c_x = \tilde{g}^x \tilde{h}^{r_x}$, $c_a = \tilde{g}^a \tilde{h}^{r_a}$, $c_d = d\tilde{g}^w$, $c_w = \tilde{g}^w \tilde{h}^{r_w}$, $z = xw$, $c_z = \tilde{g}^z \tilde{h}^{r_z}$, $c_f = (c_d)^x \tilde{h}^{r_f}$ and sends $(c_x, c_a, c_d, c_w, c_z, c_e)$ to the verifier and carry out the zero-knowledge proofs.

- *Step 2:* The prover proves to the verifier that the value committed in B in bases (g, h) is the same as the value committed in c_x in bases $(\tilde{g}, \tilde{h})$:
$$\mathsf{PK}\{(x, r, r_x) : B = g^x h^r \wedge c_x = \tilde{g}^x \tilde{h}^{r_x}\}$$

- Step 3: The prover proves to the verifier that the value committed in c_f in bases $(c_d, \tilde{h})$ is the same as the value committed in c_x in bases $(\tilde{g}, \tilde{h})$:
$$\mathsf{PK}\{(x, r_f, r_x) : c_f = (c_d)^x \tilde{h}^{r_f} \wedge c_x = \tilde{g}^x \tilde{h}^{r_x}\}$$

- Step 4: The prover proves to the verifier that $c_f\tilde{g}$ is also a commitment in bases $((c, \tilde{g}), \tilde{h})$ and the values committed in $c_f\tilde{g}$ are the same as the values committed in c_a, c_z and the power of $\tilde{h}$ in $c_f\tilde{g}$ is the same as in c_f in bases $(c_d, \tilde{h})$:
$$\mathsf{PK}\{(a, z, x, r_a, r_z, r_x) : c_f\tilde{g} = c^a \tilde{g}^z \tilde{h}^{r_f} \wedge c_f = (c_d)^x \tilde{h}^{r_f} \wedge c_a = \tilde{g}^a \tilde{h}^{r_a} \wedge c_z = \tilde{g}^z \tilde{h}^{r_z}\}$$

- Step 5: The prover proves to the verifier that c_z is a commitment to the product of values committed in c_x and c_w:
$$\mathsf{PK}\{(x, w, z, r_z, r_w, r_x, r) : c_z = (c_w)^x \tilde{h}^r \wedge c_x = \tilde{g}^x \tilde{h}^{r_x} \wedge c_w = \tilde{g}^w \tilde{h}^{r_w} \wedge c_z = \tilde{g}^z \tilde{h}^{r_z}\}$$

- Step 6: The prover proves to the verifier that c_x is a commitment to an integer belonging in Δ, and that c_a is a commitment to an integer belonging in Δ:
$$\mathsf{PK}\{(x, r, a, d) : B = g^x h^r \wedge c^a = d^x g \wedge x \in \Delta \wedge a \in \Delta\}$$

2. Detailed analysis of the protocol

The details of the protocol were omitted in [**2**], but similar protocols were described in other articles, for example, step 2,3,6 can be found in [**8**], the protocol for zero-knowledge proof that a committed value is the product of two other in step 5 can be found in [**7**]. However, step 4 has been completely developed here since a description of the protocol could not be found in the literature as it can be seen below:

Step 2: The prover proves to the verifier that the value committed in B in bases (g, h) is the same as the value committed in c_x in bases $(\tilde{g}, \tilde{h})$:
$$\mathsf{PK}\{(x, r, r_x) : B = g^x h^r \wedge c_x = \tilde{g}^x \tilde{h}^{r_x}\}$$

(1) The prover chooses $y \in [0, 2^{(\gamma_2+\kappa)}]$, $\rho_1, \rho_2 \in [0, 2^{(2\lambda+\kappa)}]$ at random and sends $Y_1 = g^y h^{\rho_1}$ and $Y_2 = \tilde{g}^y \tilde{h}^{\rho_2}$ to the verifier.
(2) The verifier chooses random $s \in [0, 2^\kappa]$ and sends it to the prover.
(3) The prover sends $u = y + sx$, $v_1 = \rho_1 + sr$ and $v_2 = \rho_2 + sr_x$.
(4) The verifier checks that $g^u h^{v_1} = Y_1(B)^s$ and $\tilde{g}^u \tilde{h}^{v_2} = Y_2(c_x)^s$.

Step 3: The prover proves to the verifier that the value committed in c_f in bases $(c_d, \tilde{h})$ is the same as the value committed in c_x in bases $(\tilde{g}, \tilde{h})$:

$$\mathsf{PK}\{(x, r_f, r_x) : c_f = (c_d)^x \tilde{h}^{r_f} \wedge c_x = \tilde{g}^x \tilde{h}^{r_x}\}$$

(1) The prover chooses $y \in [0, 2^{(\gamma_2+\kappa)}]$, $\rho_1, \rho_2 \in [0, 2^{(2\lambda+\kappa)}]$ at random and sends $Y_1 = (c_d)^y \tilde{h}^{\rho_1}$ and $Y_2 = \tilde{g}^y \tilde{h}^{\rho_2}$ to the verifier.
(2) The verifier chooses random $s \in [0, 2^\kappa]$ and sends it to the prover.
(3) The prover sends $u = y + sx$, $v_1 = \rho_1 + sr_f$ and $v_2 = \rho_2 + sr_x$.
(4) The verifier checks that $(c_d)^u h^{v_1} = Y_1(c_f)^s$ and $\tilde{g}^u \tilde{h}^{v_2} = Y_2(c_x)^s$.

Step 4: The prover proves to the verifier that $c_f\tilde{g}$ is also a commitment in bases $((c, \tilde{g}), \tilde{h})$ and the values committed in $c_f\tilde{g}$ are the same as the values committed in c_a, c_z and the power of $\tilde{h}$ in $c_f\tilde{g}$ is the same as in c_f in bases $(c_d, \tilde{h})$:

$$\mathsf{PK}\{(a, z, x, r_a, r_z, r_x) : c_f\tilde{g} = c^a \tilde{g}^z \tilde{h}^{r_f} \wedge c_f = (c_d)^x \tilde{h}^{r_f} \wedge c_a = \tilde{g}^a \tilde{h}^{r_a} \wedge c_z = \tilde{g}^z \tilde{h}^{r_z}\}$$

(1) The prover chooses $y_1, y_2, y_3 \in [0, 2^{(\gamma_2+\kappa)}]$, $\rho_1, \rho_2, \rho_3 \in [0, 2^{(2\lambda+\kappa)}]$ at random and sends $Y_1 = c^{y_1} \tilde{g}^{y_2} \tilde{h}^{\rho_1}$, $Y_2 = \tilde{g}^{y_1} \tilde{h}^{\rho_2}$ $Y_3 = \tilde{g}^{y_2} \tilde{h}^{\rho_3}$ and $Y_4 = (c_d)^{y_3} \tilde{h}^{\rho_1}$ to the verifier.
(2) The verifier chooses random $s \in [0, 2^\kappa]$ and sends it to the prover.
(3) The prover sends $u_1 = y_1 + sa$, $u_2 = y_2 + sz$, $u_3 = y_3 + sx$ $v_1 = \rho_1 + sr_f$, $v_2 = \rho_2 + sr_a$ and $v_3 = \rho_3 + sr_z$.
(4) The verifier checks that $c^{u_1} \tilde{g}^{u_2} \tilde{h}^{v_1} = Y_1(c_f\tilde{g})^s$, $\tilde{g}^{u_1} \tilde{h}^{v_2} = Y_2(c_a)^s$, $\tilde{g}^{u_2} \tilde{h}^{v_3} = Y_3(c_z)^s$ and $(c_d)^{u_3} h^{v_1} = Y_4(c_f)^s$.

Step 5: The prover proves to the verifier that c_z is a commitment to the product of values committed in c_x and c_w:

$$\mathsf{PK}\{(x, w, z, r_z, r_w, r_x, r) : c_z = (c_w)^x \tilde{h}^r \wedge c_x = \tilde{g}^x \tilde{h}^{r_x} \wedge c_w = \tilde{g}^w \tilde{h}^{r_w} \wedge c_z = \tilde{g}^z \tilde{h}^{r_z}\}$$

It is important to note that $c_z = (c_w)^x \tilde{h}^{(r_z - r_w x)}$, as it will be used in the proof.

(1) The prover chooses $y \in [0, 2^{(\gamma_2+\kappa)}]$, $\rho_x, \rho_z \in [0, 2^{(2\lambda+\kappa)}]$ at random and sends $Y_x = \tilde{g}^y \tilde{h}^{\rho_x}$ and $Y_z = \tilde{g}^y \tilde{h}^{\rho_z}$ to the verifier.
(2) The verifier chooses random $s \in [0, 2^\kappa]$ and sends it to the prover.
(3) The prover sends $u = y + sx$, $v_x = \rho_x + sr_x$ and $v_z = \rho_z + s(r_z - r_w x)$.
(4) The verifier checks that $\tilde{g}^u \tilde{h}^{v_x} = Y_x(c_x)^s$ and $\tilde{g}^u \tilde{h}^{v_z} = Y_z(c_z)^s$.

Step 6: The prover proves to the verifier that c_x is a commitment to an integer belonging in Δ, and that c_a is a commitment to an integer belonging in Δ:

$$\mathsf{PK}\{(x, r, a, d) : B = g^x h^r \wedge c^a = d^x g \wedge x \in \Delta \wedge a \in \Delta\}$$

As it can be seen in the construction of the dynamic universal accumulator, the updated value of a is calculated $\bmod x$, thus proving that x is an integer belonging to the interval Δ will be enough. The complete description and proof of this step 6 can be found in the Section 3 of [**8**].

Theorem 5. *The previous protocol is a zero-knowledge proof of knowledge of the values (x, r, a, d) such that $B = g^x h^r \bmod N$ and (a, d) is a valid nonmembership witness of x for the accumulator c.*

Proof. Completeness of the protocol is clear. The commitments c_x, c_a, c_f, c_z, c_d and c_w are computed at random, since the commitments do not reveal anything statistically, in order to prove the zero-knowledge of the protocol, it is sufficient to verify the zero-knowledge of each step. In this case, since it would be very similar for each step, only the zero-knowledge proof of knowledge of step 2 would be demonstrated. It will be shown that even a malicious verifier V', would not obtain new information on x from the execution of the protocol.
Step 2:

$$\mathsf{PK}\{(x, r, r_x) : B = g^x h^r \wedge c_x = \tilde{g}^x \tilde{h}^{r_x}\}$$

A transcript $(Y_1, Y_2, s, u, v_1, v_2)$ of the protocol between the prover and V' can be simulated as follows:

(1) Choose at random $u \in [0, 2^{(\gamma_2+\kappa)}]$, $v_1, v_2 \in [0, 2^{(2\lambda+\kappa)}]$
(2) Choose at random $s \in [0, 2^\kappa]$ with the same distribution as V' does and,
(3) compute $Y_1 = g^u h^{v_1} \cdot B^{-s}$ and $Y_2 = \tilde{g}^u \tilde{h}^{v_2} \cdot (c_x)^{-s}$

In order to show soundness, just as in the zero-knowledge proof, it is enough with the verification of each step of the protocol. Such proofs of soundness are very similar for instance, the ones in [**7**]. Only the proof for Step 4 will be detailed, since this is the step that was nos explicitly available in the existing literature.
To demonstrate soundness, it can be assumed that some prover P^* can execute the protocol with a non-negligible success probability. An algorithm that uses P^* as a subroutine and computes a way to open the commitment, except with negligible probability, is shown below.
By assumption on P^*, using standard rewinding techniques, there is a situation, for a given PK, P^* could answer for two different challenges s and s' the values $(u_1, u_2, u_3, v_1, v_2, v_3)$ and $(u'_1, u'_2, u'_3, v'_1, v'_2, v'_3)$. Let $\hat{a} = u_1 - u'_1$, $\hat{z} = u_2 - u'_2$, $\hat{x} = u_3 - u'_3$, $\widehat{r_f} = v_1 - v'_1$, $\widehat{r_a} = v_2 - v'_2$, $\widehat{r_z} = v_3 - v'_3$ and $\hat{s} = s - s'$. Then we have:

(1) $c_a^{\hat{s}} = \tilde{g}^{\hat{a}} \tilde{h}^{\widehat{r_a}}$
(2) $c_z^{\hat{s}} = \tilde{g}^{\hat{z}} \tilde{h}^{\widehat{r_z}}$
(3) $c_f^{\hat{s}} \tilde{g} = c^{\hat{a}} \tilde{g}^{\hat{z}} \tilde{h}^{\widehat{r_f}}$
(4) $c_f^{\hat{s}} = (c_d)^{\hat{x}} \tilde{h}^{\widehat{r_f}}$

In this case, only the value c_a will be developed, as the other three can be computed in the same way.
Straightaway, consider that $\hat{s}$ divides both $\hat{a}$ and $\widehat{r_a}$. Let $a' = \hat{a}/\hat{s}$ and $r'_a = \widehat{r_a}/\hat{s}$, then the element $b = c_a^{-1} \tilde{g}^{a'} \tilde{h}^{r'_a}$ satisfies that $b^{\hat{s}} = 1$. But under the small order assumption stated in the number theoretic assumptions (see Section 1 of preliminaries), the probability that $b \neq 1$ is negligible and so c_a can be open correctly by

sending $\hat{a}/\hat{s}$, $\widehat{r_a}/\hat{s}$, b. Now, the case where $\hat{s}$ does not divide both $\hat{a}$ and $\widehat{r_a}$ happens with negligible probability, as we argue below.
Now suppose the case where $\hat{s}$ does not divide both $\hat{a}$ and $\widehat{r_a}$ happens with non-negligible probability. It will be demonstrated that an algorithm can be constructed that breaks the strong RSA assumption.
Let h be an input chosen at random. Then, g is set as $g = h^{\alpha}$ for a random α. Then g, h are sent to the adversary. Now, suppose the case $c_a^{\hat{s}} = \tilde{g}^{\hat{a}}\tilde{h}^{\widehat{r_a}}$ and $\hat{s}$ does not divide both $\hat{a}$ and $\widehat{r_a}$, which by assumption happens with non-negligible probability. If the equation $g = h^{\alpha}$ is inserted in the other equation it is obtained:

$$c_a^{\hat{s}} = h^{\alpha(\hat{a})+\widehat{r_a}}$$

Suppose $s > s'$ that is $\hat{s} > 0$. The analysis is divided into two cases:

- $\hat{s}$ *does not divide* $\alpha(\hat{a}) + \widehat{r_a}$. In this case, let $m = \gcd(\hat{s}, \alpha(\hat{a}) + \widehat{r_a})$. η, β are chosen that

  $$\eta(\hat{s}) + \beta(\alpha(\hat{a}) + \widehat{r_a}) = m$$

  and

  $$h^m = h^{\eta(\hat{s})+\beta(\alpha(\hat{a})+\widehat{r_a})} = (h^{\eta}c_a^{\beta})^{\hat{s}}$$

  $\tilde{b}$ is set as $\tilde{b} = (h^{\eta}c_a^{\beta})^{\hat{s}/m}h^{-1}$, where $\tilde{b}^m = 1$ and $h\tilde{b} = (h^{\eta}c_a^{\beta})^{\hat{s}/m}$.

  If $\tilde{b} = 1$, then the strong RSA assumption is broken. Otherwise, the only remaining non-negligible case is $\tilde{b}^2 = 1$ under the small order assumption. In this case, two options can be considered:

  – if $(\hat{s})/d$ is odd, then $\tilde{b}^{\hat{s}/d} = \tilde{b}$, but this breaks again the strong RSA assumption.
  – if $(\hat{s})/d$ is even, then $(h^{\eta}c_a^{\beta})^{\hat{s}}$ has odd order, which contradicts the fact that $ord(h\tilde{b}) = 2ord(h)$.

  In both cases, the assumptions of the group are broken if $\hat{s}$ does not divide $\alpha(\hat{a}) + \widehat{r_a}$.

- $\hat{s}$ *divides* $\alpha(\hat{a}) + \widehat{r_a}$. Since the adversary does not know the full information of the choice of α, this case happens with a probability at most of $1/2$ in the situation that $\hat{s}$ does not divide both $\hat{a}$ and $\widehat{r_a}$. Let q be some prime factor in $\hat{s}$ such that q^j is the maximal q-power dividing $\hat{s}$, and at least one of $\hat{a}$, $\widehat{r_a}$ are non-zero module q^j (such q must exist since $\hat{s}$ does not divide both $\hat{a}$ and $\widehat{r_a}$). Mind that if q^j divides $\hat{a}$, it would have to divide $\widehat{r_a}$ as well, which is a contradiction. Therefore $\hat{a} \neq 0 \bmod q^j$. α can be written as $\alpha = y + z.ord(h)$, where $y = \alpha(\bmod ord(h))$. It must be mentioned that g represents all the information the adversary has about α, and y is uniquely determined from g and z is completely unknown. Seeing that q^j actually divides $\alpha(\hat{a}) + \widehat{r_a}$, it can be observed:

  $$\alpha(\hat{a}) + \widehat{r_a} = z(\hat{a})ord(h) + y(\hat{a}) + \widehat{r_a} = 0(\bmod q_j)$$

  z must satisfy the previous equations in order for the bad case to occur. The number of solutions modulo q^j of this equation is at most $\gcd((\hat{a})ord(h), q^j)$. This number is a power of q, but is at most q^{j-1}. The probability that z satisfies the equation is statistically close to $1/q \leq 1/2$.

Now, that soundness for c_a has been shown, the same protocol can be followed for the other values c_z, c_f, $c_f\tilde{g}$ and the soundness of the whole step 4 could be demonstrated.

□

Combining the proofs of soundness of the six steps, there is a knowledge extractor that outputs a valid committed value x and its nonmembership witness. In case the knowledge extractor fails, the strong RSA assumption can be broken. If it can be assumed that the extractor succeeds and computes $(x, a, w, z, r, r_x, r_a, r_w, r_z, r_f)$ such that:

(1) $B = g^x h^r$
(2) $c_x = \tilde{g}^x \tilde{h}^{r_x}$
(3) $c_a = \tilde{g}^a \tilde{h}^{r_a}$
(4) $c_z = \tilde{g}^z \tilde{h}^{r_z}$
(5) $c_z = \tilde{g}^{xw} \tilde{h}^{r_z}$
(6) $c_f = c_d \tilde{h}^{r_f}$
(7) $c_f\tilde{g} = c^a \tilde{g}^z \tilde{h}^{r_f}$
(8) $c_d = d\tilde{g}^w$

where the equations (1) and (2) are obtained in step 1, the (2) and (6) equations are procured in step 3 of the protocol, in step 4 the equations (3), (4) and (7) are find, and finally the equations (4) and (5) come from the step 5 of the protocol. From equations (4) and (5), it can be derived that $z = xw$. $(c_d)^x\tilde{g} = c^a\tilde{g}^z$ is deduce by equation (6) and (7). From equation (8), d can be defined as $d = c_d/\tilde{g}^w$, by elevating this to x, it is obtained $d^x = (c_d/\tilde{g}^w)^x = (c_d)^x/\tilde{g}^z = c^a\tilde{g}^{-1}$, so $c^a = d^x\tilde{g}$. It is known from step 6 that x and a are integers in the interval Δ, therefore (x, r, a, d) are the outputs of the protocol, where B is a commitment of x, and (a, d) are the nonmembership witness for x.

Chapter 5

Incorporating Revocation of Users into Attribute-Based Signatures

In this section, it is formulated how to incorporate revocation of users into an attribute-based signature scheme. In this particular case, the original scheme is the one in [**1**]. The security of this scheme (privacy and unforgeability of the signatures) is proven under the Strong RSA Assumption and the Decisional Diffie-Hellman Assumption in QR_N with known factorization (Assumptions 1 and 3 in Chapter 2). Firstly, it will be explained the main aspects of this signature scheme and then how to add revocation into the scheme. Finally, efficiency will be assessed as it is important to evaluate how more costly our resulting scheme would be.

1. Existing Attribute-Based Scheme for a Threshold Signing Policy

The original attribute-based signature scheme was developed in [**1**]. For simplicity of explanation, the signature scheme is developed in the case of threshold signing policies, a pair $(\mathcal{P}, \Gamma)$ will be represented as $(\mathcal{P}, \ell)$, where $1 \leq \ell \leq |\mathcal{P}|$. The four existing algorithms (Setup, KeyGen, Sign, Verify), will be explained below.

$\mathsf{Setup}(1^\lambda)$. The setup algorithm starts by running $(P, Q, N, g) \leftarrow \mathsf{RSA.Inst}(1^\lambda)$, where $N = PQ$, $P = 2p+1$ and $Q = 2q+1$. Choose security parameters $\kappa, \gamma_1, \gamma_2 \in \mathbb{N}$ and $\epsilon \in \mathbb{R}$, $\epsilon > 1$, such that $\gamma_1 - 2 > \epsilon(\gamma_2 + \kappa) > \lambda$. We denote as Δ the set of integers in the interval $[2^{\gamma_1} - 2^{\gamma_2} + 1, 2^{\gamma_1} + 2^{\gamma_2} - 1]$.

A prime number q' in the interval $[2^{\kappa-1}, 2^\kappa]$ is chosen. Two cryptographic hash functions $H_0 : \{0,1\}^* \to QR(N)$ and $H_1 : \{0,1\}^* \to \mathbb{Z}^*_{q'}$ are also chosen. Finally, the global set of attributes $\tilde{\mathcal{P}}$ has to be chosen.

The public parameters are $\mathsf{pms} = (\kappa, \gamma_1, \gamma_2, \epsilon, \Delta, \lambda, N, g, H_0, H_1, q', \tilde{\mathcal{P}})$, whereas the master secret key is $\mathsf{msk} = (P, Q)$.

$\mathsf{KeyGen}(\mathsf{id}, S, \mathsf{msk}, \mathsf{pms})$. The key generation algorithm takes as input an identity id, a subset of attributes $S \subset \tilde{\mathcal{P}}$ satisfied by id, the master secret key msk and the public parameters pms. The master entity chooses at random a prime number $e \stackrel{R}{\leftarrow} \Delta$ such that $\gcd(e, pq) = 1$ and, for each $\mathsf{at}_i \in S$, computes the value $\mathsf{sk}_i = H_0(\mathsf{at}_i)^{1/e} \bmod N$ (using the knowledge of the prime numbers P, Q). The global secret key is $\mathsf{sk}_{\mathsf{id},S} = (e, \{\mathsf{sk}_i\}_{\mathsf{at}_i \in S})$.

$\mathsf{Sign}(m, \mathcal{P}, \ell, \mathsf{sk}_{\mathsf{id},S}, \mathsf{pms})$. 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 $\mathsf{sk}_{\mathsf{id},S} = (e, \{\mathsf{sk}_i\}_{\mathsf{at}_i \in S})$ and the public parameters pms. The algorithm selects a minimally authorized set S', this is, a subset of $S \cap \mathcal{P}$ of cardinality exactly ℓ. Without loss of generality, let us assume $\mathcal{P} = \{\mathsf{at}_1, \ldots, \mathsf{at}_n\}$ and $S' = \{\mathsf{at}_1, \ldots, \mathsf{at}_\ell\}$. To generate the signature, it proceeds as follows:

(1) Choose $h \stackrel{R}{\leftarrow} QR(N)$ and $r \stackrel{R}{\leftarrow} \mathbb{Z}_N$. Compute $A = g^r \bmod N$, $B = g^e \cdot h^r \bmod N$.

(2) For $j = \ell + 1, \ldots, n$, choose $c_j \stackrel{R}{\leftarrow} \mathbb{Z}_{q'}$, $C_j, Z_j \stackrel{R}{\leftarrow} QR(N)$, $u_j \stackrel{R}{\leftarrow} \pm\{0,1\}^{\epsilon(\gamma_2+\kappa)}$, $v_j \stackrel{R}{\leftarrow} \pm\{0,1\}^{\epsilon(\lambda+\kappa)}$, $w_j \stackrel{R}{\leftarrow} \pm\{0,1\}^{\epsilon(\gamma_1+\lambda+\kappa+1)}$, and compute the values $D_j = \frac{A^{u_j - c_j 2^{\gamma_1}}}{g^{w_j}} \bmod N$, $E_j = \frac{g^{v_j}}{A^{c_j}} \bmod N$, $F_j = g^{u_j - c_j 2^{\gamma_1}} \cdot h^{v_j} \cdot B^{c_j} \bmod N$ and $G_j = \frac{C_j^{u_j - c_j 2^{\gamma_1}} \cdot H_0(\mathsf{at}_j)^{c_j}}{Z_j^{w_j}} \bmod N$.

(3) For $i = 1, \ldots, \ell$, choose $Z_i \stackrel{R}{\leftarrow} QR(N)$, $\alpha_i \stackrel{R}{\leftarrow} \pm\{0,1\}^{\epsilon(\gamma_2+\kappa)}$, $\beta_i \stackrel{R}{\leftarrow} \pm\{0,1\}^{\epsilon(2\lambda+\kappa)}$, $\delta_i \stackrel{R}{\leftarrow} \pm\{0,1\}^{\epsilon(\gamma_1+2\lambda+\kappa+1)}$, and compute the values $C_i = \mathsf{sk}_i \cdot Z_i^r \bmod N$, $D_i = \frac{A^{\alpha_i}}{g^{\delta_i}} \bmod N$, $E_i = g^{\beta_i} \bmod N$, $F_i = g^{\alpha_i} \cdot h^{\beta_i} \bmod N$ and $G_i = \frac{C_i^{\alpha_i}}{Z_i^{\delta_i}} \bmod N$.

(4) Compute the hash value $c = H_1\left(m, \mathcal{P}, \ell, h, A, B, \{C_i, D_i, E_i, F_i, G_i, Z_i\}_{\mathsf{at}_i \in \mathcal{P}}\right)$.

(5) Find the (only) polynomial $f(x) \in \mathbb{Z}_{q'}[X]$ with degree at most $n - \ell$ such that $f(0) = c \bmod q'$ and $f(j) = c_j \bmod q'$ for all $j = \ell + 1, \ldots, n$.

(6) For $i = 1, \ldots, \ell$, compute $c_i = f(i) \bmod q'$ and then compute the values $u_i = \alpha_i - c_i \cdot (e - 2^{\gamma_1})$, $v_i = \beta_i - c_i \cdot r$ and $w_i = \delta_i - c_i \cdot e \cdot r$, over the integers.

The resulting signature is $\sigma = (f(x), h, A, B, \{(C_i, u_i, v_i, w_i, Z_i)\}_{\mathsf{at}_i \in \mathcal{P}})$.

$\mathsf{Verify}(\sigma, m, \mathcal{P}, \ell, \mathsf{pms})$. The verification algorithm takes as input a message m, the signature σ of m, the threshold signing policy $(\mathcal{P}, \ell)$, with $n = |\mathcal{P}|$, and the public parameters pms. It proceeds as follows:

(1) Verify that the degree of $f(x)$ is at most $n - \ell$. For all $\mathsf{at}_i \in \mathcal{P}$, verify that $u_i \in \pm\{0,1\}^{\epsilon(\gamma_2+\kappa)}$, $v_i \in \pm\{0,1\}^{\epsilon(\lambda+\kappa)}$ and $w_i \in \pm\{0,1\}^{\epsilon(\gamma_1+\lambda+\kappa+1)}$. Return 0 if this is not the case.

(2) For all $\mathsf{at}_i \in \mathcal{P}$, compute $c_i = f(i)$ and then compute the values $D_i = \frac{A^{u_i - c_i 2^{\gamma_1}}}{g^{w_i}} \bmod N$, $E_i = \frac{g^{v_i}}{A^{c_i}} \bmod N$, $F_i = g^{u_i - c_i 2^{\gamma_1}} \cdot h^{v_i} \cdot B^{c_i} \bmod N$ and $G_i = \frac{C_i^{u_i - c_i 2^{\gamma_1}} \cdot H_0(\mathsf{at}_i)^{c_i}}{Z_i^{w_i}} \bmod N$.

(3) Return 1 if $f(0) = H_1\left(m, \mathcal{P}, \ell, h, A, B, \{C_i, D_i, E_i, F_i, G_i, Z_i\}_{\mathsf{at}_i \in \mathcal{P}}\right)$, and return 0 otherwise.

2. Incorporating Revocation into the Signature Scheme

The revocation of users method, based on the dynamic universal accumulator protocol defined in the previous Chapter 4, is developed in this section. Our main goal is to design a protocol to remove users that have misbehaved or that have lost their attributes. The major problem consists in maintaining privacy, a user who wants to prove that he is not in the revocation list, can not just show his serial number and its corresponding nonmembership witness, because the user will be revealing his serial number. With the aid of this protocol, there is the possibility to revoke a user without revealing his identity to a master entity, which is essential in the attribute-based signature previously formulated.

Setup: In the RSA attribute-based signature, the master entity chooses $N = PQ$, $P = 2p+1$ and $Q = 2q+1$. The public parameters are $\mathsf{pms} = (\kappa, \gamma_1, \gamma_2, \epsilon, \Delta, \lambda, N, g, H_0, H_1, q', \tilde{\mathcal{P}})$ and the master secret key is $\mathsf{msk} = (P, Q)$. Additionally, the master creates the universal accumulator of Chapter 3, and chooses a random $\tilde{g} \in QR_N$ and adds $\tilde{g}$ to the public parameters.

KeyGen: In our signature, we take as input an identity id, a subset of attributes $S \subset \tilde{\mathcal{P}}$ satisfied by id, the master secret key msk and the public parameters pms. The master entity chooses at random a prime number e, and the global secret key $(\mathsf{sk}_{\mathsf{id},S} = (e, \{\mathsf{sk}_i\}_{\mathsf{at}_i \in S}))$ is computed. Moreover, the master entity secretly stores the relation between id and e in a table st, and given the current revocation list c, the master computes the nonmembership witness (a, d) for the non-accumulated value e and sends it to the user. The new global secret key $(\mathsf{sk}_{\mathsf{id},S} = (e, \{\mathsf{sk}_i\}_{\mathsf{at}_i \in S}), a, d)$

Sign and Verify: $(m, \mathcal{P}, \ell, \mathsf{sk}_{\mathsf{id},S}, \mathsf{pms})$.The signature consists of a zero-knowledge non-interactive proof of knowledge of an integer e and at least ℓ e-th roots modulo N of the values $H(\mathsf{at}_1), \ldots, H(\mathsf{at}_n)$. In addition, the prover proves to the verifier that e is not in the revocation list. This can be achieved with the zero-knowledge proof protocol in Chapter 4. The prover first commits his value e, and then proves that it is the same as the value committed in the element B of the signature, and that this value has not been accumulated in c.

Revocation: If we want to revoke a user, the master entity must add his value to the current accumulator. Let c be the current accumulator, the new accumulator results in $\tilde{c} = c^e \bmod N$.

Membership Update: Let $\tilde{c}$ be the current accumulator and c be the previous accumulator stored. Each new user updates his nonmembership witness using the method described in Chapter 4.

3. Efficiency

At last, it is essential to assess the efficiency of the resulting signature scheme. In order to do that, with the choice of the parameters given in [**1**], the growth in the length of the signature will be determined. For a security level of $\lambda = 1024$, the parameters used are $\gamma_1 = 1080$, $\gamma_2 = 800$ and $\kappa = 160$. It must be noted that, although in the preceding Chapter 4 the protocol used to prove that a committed value was not accumulated was interactive, when executing the protocol now, as part of the attribute-based signature scheme, a non-interactive method must be enlisted, thus a hash function will be used instead. For example, the resulting protocol for Step 2 of the protocol developed in Chapter 4 will be:

Step 2: The prover proves to the verifier that the value committed in B in bases (g, h) is the same as the value committed in c_x in bases $(\tilde{g}, \tilde{h})$:

$$\mathsf{PK}\{(x, r, r_x) : B = g^x h^r \wedge c_x = \tilde{g}^x \tilde{h}^{r_x}\}$$

(1) The prover chooses $y \in [0, 2^{(\gamma_2+\kappa)}]$, $\rho_1, \rho_2 \in [0, 2^{(2\lambda+\kappa)}]$ and computes $Y_1 = g^y h^{\rho_1}$ and $Y_2 = \tilde{g}^y \tilde{h}^{\rho_2}$.

(2) The prover computes $s = H(Y_1 \parallel Y_2)$.

(3) The prover computes $u = y + sx$, $v_1 = \rho_1 + sr$, $v_2 = \rho_2 + sr_x$ and sends (s, u, v_1, v_2) to the verifier.

(4) The verifier checks whether:
$$s = H(g^u h^{v_1} B^{-s} \bmod N \parallel \tilde{g}^u \tilde{h}^{v_2} (c_x)^{-s} \bmod N)$$

The length of this particular step with the parameters given is 5536 bits and four additional exponentiations will be needed for the prover to perform this step. In particular, for the complete protocol described in Chapter 4 for proving that the value e has not been accumulated, the length of the proof would be around 50000 bits and the cost of the protocol is dominated by 44 exponentiations modulo N.

The original attribute-based signature scheme for a (ℓ, n)-threshold signing policy with $n = |\mathcal{P}|$ attributes has an approximate length of $6800n + 3200 - 160\ell$ bits. The cost of each execution of the signing protocol is dominated by $10n$ exponentiations modulo N. Since the length of the signature depends on the number of attributes and revocation of users is independent of it, implementing revocation of users might not be efficient for a small number of attributes, but it could be efficient for larger signing policies, with a larger number of attributes. In Table 1 below, the length of the attribute-based signatures is included, with and without the revocation extension, for different values of the number n of attributes of the signing policy, as well as the percentage increase of length of the signatures. The results in Table 1 give an idea about how much does it cost, at least in terms of communication cost, to add revocation to this particular attribute-based signature. This was the main goal of this work.

n	Approx. Kbit of ABS	Approx. Kbit of ABS + Revocation	Increasing %
5	34	84	147%
10	68	118	74%
15	102	152	49%
20	136	186	37%
30	204	254	25%
50	340	390	15%
75	510	560	10%
100	680	730	7%

Table 1. Cost of ABS Signatures with and without revocation.

Chapter 6
Conclusions

In this dissertation, the main goal was to incorporate revocation of users into an existing RSA attribute-based signature. In the first place, I already had some basic notions about cryptography prior to my involvement in this project. However, despite my knowledge of this field, attribute-based cryptography was an area into which I had not delved before. Therefore, in order to get acquainted with the concepts of attribute-based cryptography I undertook an extensive research on the subject reading a number of papers on the topic. Some of these concepts appear in the preliminaries which are used later in the development of the work.

Afterwards, in order to incorporate revocation into the attribute-based signature, a protocol was chosen from the existing literature. This protocol proposed in [**2**] in a sketched way and described in detail here in Chapter 4, was based on the concept of universal accumulators. The sketched description of the protocol in [**2**] referred to some other papers: [**8**], [**7**]; and even in some cases such as step 4, the description could not be found in the literature and was developed from scratch. For this step 4, the soundness property was demonstrated using similar arguments as the ones that could be found in [**7**]. The development of Chapter 4 is maybe the main technical contribution of our work.

Furthermore, once the protocol was completely developed, it was incorporated into the RSA attribute-based signature. The value e of the secret key of the user is the value that would be used by the protocol to prove that this user has not been revoked. At last, once the new scheme that incorporates revocation was designed, the efficiency of the new protocol was analyzed, in particular for the case of the communication cost, or in other words the length of the signature. From the table at the end of Chapter 5, it can be observed that this protocol for incorporating revocation should work better in policies with a large number of attributes as the total length of the new signature would not be much bigger than the original signature. An analysis of the computational complexity, that is, the time required to verify a signature, could be done essentially in the same way, and the conclusions would be almost identical.

Finally, I would like to say that this has been an excellent learning experience and I am satisfied with the results from applying and adapting the revocation protocol into the existing attribute-based signature scheme. In addition, thanks to my acquired knowledge on this field after the research and application of this protocol, I could further develop this paper in the future by studying different types of revocation protocols and testing them for efficiency in the same given signature.

References

[1] Javier Herranz. Attribute-Based Signatures from RSA. *Theoretical Computer Science*, Volume **527**, pag. 73-82 (2014).

[2] J. Li, N. Li and R. Xue. Universal accumulators with efficient nonmberbership proofs. *Proc. of ACNS'07*, LNCS **4521**, Springer-Verlag, pg. 253-269 (2007)

[3] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. *Proceedings of the 3rd Conference on Security in Communication Networks*, pg. 268-289 (2002)

[4] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. *Advances of Criptology - CRYPTO '02*, pg. 61-76 (2002)

[5] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. *Advances of Criptology - EUROCRYPT '99*, pg. 106-121 (1999)

[6] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. *Advances of Criptology - CRYPTO '99*, pg. 413-430 (1999)

[7] I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. *Advances of Criptology - ASIACRYPT '02*, pg. 125-142 (2002)

[8] F.Boudot. Efficient proofs that a committed number lies in an interval. *Advances of Criptology - EUROCRYPT '00*, pg. 431-444, (2000)

[9] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. *Advances of Criptology - CRYPTO '00*, pg. 255-270, (2000)

[10] G. Ateniese, D. Song and G. Tsudik. Quasi-efficient revocation of group signatures. *Proceedings of Financial Cryptography*, pg. 183-197 (2001)

[11] J. Camenisch and A. Lysynskaya. Efficient Revocation of Anonymous Group Membership Certificates and Anonymous Credentials (2001)

[12] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. *Advances of Criptology - CRYPTO '97*, pg. 16-30, (1997)

[13] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. *Advances of Criptology - EUROCRYPTO '97*, pg. 480-494, (1997)

[14] A. Kiayias and M. Yung. Efficient secure group signatures with dynamic joins and keeping anonymity against group managers. *Proc. of Mycrypt '05*, LNCS **3715**, Springer-Verlag, pg. 151-170 (2005)

[15] H.K. Maji, M. Prabhakaran and M. Rosulek. Attribute-based signatures. *Proc. of CT-RSA'11*, LNCS **6558**, Springer Verlag, pg. 376-392 (2011)

[16] H.K. Maji, M. Prabhakaran and M. Rosulek. Attribute-based signatures. *Proc. of CT-RSA'11*, LNCS **6558**, Springer-Verlag, pg. 376-392 (2011)

[17] J. Bethencourt, A. Sahai and B. Waters. Ciphertext-policy attribute-based encryption. *Proc. of IEEE Symposium on Security and Privacy*,IEEE Society Press, pg. 321-334 (2007)

[18] V. Goyal, O. Pandey, A. Sahai and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. *Proc. of Computer and Communications Security, CCS'06*, ACM Press, pg. 89-98 (2006)

[19] A. Lewko, T. Okamoto, A. Sahai, K. Takashima and B. Waters. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. *Proc. of Eurocrypt'10*, LNCS **6110**, Springer-Verlag, pg. 62-91 (2010)