Université Bordeaux 1
Master Cryptologie et Sécurité Informatique

Fuzzy identity based encryption

Florent LLOMBARD Lambert ROSIQUE

March 18, 2013

Supervisor : Gilles ZEMOR

Contents

1 Introduction

Cryptology is based on trapdoors problem. These are problems which are hard to solve, except if a certain information, a trap, is given. Suppose the trapdoor problem is to compute K a key. To cipher a message M, we use the key K as a mask : $C = MK$. Suppose the trap is given to an allowed user. He can compute K from the trap, and recover the message $M = CK^{-1}$. In a such cryptosystem, security is based on the hardness of solving the trapdoor problem without the trap.

A standard assumption is **Decisional Diffie-Hellman (DDH) assumption**. Suppose a challenger chooses $a, b, z \in \mathbb{Z}_p$ at random. The DDH assumption is that no polynomial-time adversary is able to distinguish the tuple (g^a, g^b, g^{ab}) from the tuple (g^a, g^b, g^z) with more than a negligible advantage. DDH is a trapdoor problem : (a, g^b) of (g^a, b) are obvious traps to compute g^{ab}. Cryptologists have two roles : define new trapdoors problems, and find new traps to these problems.

This document presents a type of attribute-based encryption, where a user who holds a set of attributes ω can decrypt ciphertext encrypted with attributes $\omega\prime$ *if and only if* they share at least k common elements, ie $|\omega \cap \omega\prime| \geq k$.
[2] presented such a scheme. To build their scheme, [2] combine two interesting technologies : a new trapdoor offered by a way to share a secret in exponent, and bilinear maps, which provide a new dimension in the DDH problem : the Bilinear Decision Diffie Hellman (BDH) problem. To combine these two ideas, a new assumption is defined, very close to BDH assumption : Modified Decision Diffie Hellman assumption.

We will see in a first part the trapdoor offered by sharing secret, and which scheme it is possible to build. In a second part, we will study this new BDH assumption, by exposing an example, and then a discussion on how to modify BDH assumption into MBDH assumption to reach the fuzzy attribute-based encryption secure. In a third part, we will present the [2] cryptosystem, and explain in detail its security proof.

2 Fuzzy attribute-based encryption

In this section, we are looking for builduing a fuzzy atttibute-based encryption scheme. Denote $\mathcal{U}$ the set of whole attributes. An attribute-based encryption scheme is a cryptosystem where encryption is made for some attributes $\omega \subset \mathcal{U}$. We will see identities as subsets of $\mathcal{U}$. A user who holds attributes $\omega\prime \supset \omega$ will be able to decrypt, since he holds every required attributes. Fuzziness in a such attribute-based scheme, is to make a user $\omega\prime$ able to decrypt if he holds at least k attributes in ω. Mathematically, decryption is possible only if $|\omega \cap \omega\prime| \geq k$.

In parallel, we are looking for building a key distribution scheme, where some users are revoked[1]. Suppose a provider broadcasts an encrypted channel. Some users are allowed to decrypt the broadcast, they all hold the same key. For some reasons, the group controller needs to revoke users. In order to do so, some informations are broadcasted to all allowed users, including faulty users to revoke. Users to revoke do not get enough information to compute the new key, when still allowed users do. The broadcast is then encrypted with the new key, and faulty users have been revoked, since they could not compute the new key.

These two cryptosystems can be made from the same tool : (k, n)-threshold scheme, these schemes used to share a secret. First is presented Shamir's secret sharing method. This very powerful tool will helps us to build a one-time revocation scheme, and a one-time fuzzy attribute based encryption. After one decryption, or key distribution, *material* to compute a new secret is exposed. We will see in a second part how to make use of this material possible, without revealing material itself. This is the secret sharing in exponent. Thanks to this new secret sharing method, it is possible to build a (many-times) fuzzy attribute based encryption, and also a (many-times) revocation scheme.

2.1 Threshold scheme

A threshold scheme is a kind of framework for secret sharing. The main idea of sharing secret is to divide a secret into a few shares, such that a defined number of shares put together permit us to compute efficiently the secret, but with less shares it is not possible to recover it.

Let us define formally the concept of sharing secret.

Denote y a secret to share. The aim is to divide y into n *shares*, noted $y_1, \ldots, y_n$. Let k be an integer such that $k \leq n$. The two properties below define a (k, n)-*threshold* scheme.

- The knowledge of any k shares is enough to compute efficiently the secret y.
- The knowledge of any $k-1$ shares does not give any information about y.

Below will be presented two threshold schemes, that fits this framework. Both are based on Lagrange's polynomial interpolation, a way to build a polynomial of degree k with $k+1$ given points.

2.2 Lagrange's interpolation

In this section, we review Lagrange's interpolation through its construction and basic properties. Given a set of $k+1$ points $((x_i, y_i))_{0\leq i\leq k}$, there is a unique polynomial, noted q, whose degree is at most k such that $q(x_i) = y_i$ for all i. Lagrange's interpolation is a method to build this polynomial. Less formally, a polynomial of degree k is uniquely defined by its values on $k+1$ points.

It is easily computable using Lagrange's polynomials, which is a family of $k+1$ polynomials $\lambda_i(x)$, defined as $\lambda_i(x) = \prod_{0\leq j\leq k,\ j\neq i} \frac{x-x_j}{x_i-x_j}$ for all $i \in \{0\dots k\}$.

The interpolating polynomial is $q(x) = \sum_{i=0}^{k} y_i\lambda_i(x)$.

Let's check that such a polynomial q **is indeed an interpolation of the family of points** $((x_i, y_i))_{0\leq i\leq k}$.

Let i_0 be in $\{0\dots k\}$. We want to show that $q(x_{i_0}) = y_{i_0}$.

For all $i \neq i_0$, we have $\lambda_i(x_{i_0}) = \prod_{0\leq j\leq k,\ j\neq i} \frac{x_{i_0}-x_j}{x_i-x_j}$. In particular , one of the factors is $\frac{x_{i_0}-x_{i_0}}{x_i-x_{i_0}} = 0$, which means $\lambda_i(x_{i_0}) = 0$ for all $i \neq i_0$.

On the other hand, $\lambda_{i_0}(x_{i_0}) = \prod_{j\neq i_0} \frac{x_{i_0}-x_j}{x_{i_0}-x_j} = 1$. Finally, we have $q(x_{i_0}) = \sum_{i=0}^{k} y_i\lambda_i(x_{i_0}) = y_{i_0}$, which proves that $q(x) = \sum_{i=0}^{k} y_i\lambda_i(x)$ is an interpolation of the family of points given above.

The degree of q is at most k. The degree of λ_i, which is a product of k polynomials of degree 1 is indeed at most k, and q is the sum of theses polynomials λ_i, so q is indeed of degree at most k.

Such an interpolating polynomial of degree at most k is unique. Let q_1 and q_2 be two polynomials such that for all i, $q_1(x_i) = q_2(x_i) = y_i$, and $\deg(q_1), \deg(q_2) \leq k$. Then the polynomial $q_1 - q_2$ admits $k+1$ roots, which are exactly the x_i, and $deg(q_1 - q_2) \leq k$. Since a polynomial of degree k has at most $k+1$ roots, we deduce

that $q_1 - q_2 = 0$ and the above interpolation formula provides the unique polynomial of degree at most k which interpolates the above family of $k+1$ points.

2.3 Secret sharing methods

We will present Shamir's sharing secret method, that uses Lagrange's polynomial interpolation. This first method to share a secret is enough to build a one-time fuzzy attribute-based encryption and a one-time revocation scheme. Then we will see an other threshold scheme, that is derived from Shamir's method. Thanks this new tool, we will build our two cryptosystems.

2.3.1 Shamir's secret method

In 1979, Shamir presented [3] a way to share a secret. Remember that we want to create a (k, n)-threshold scheme. Denote y a data. We want to divide y into n shares such that k of them is enough to compute y but $k-1$ shares does not provide any information about the secret data y. Suppose that y is a number.

Let p be a prime number bigger than n and y. We see y as an element of the finite field $\mathbb{Z}_p$. Let $q(x) = \sum_{i=0}^{k-1} q_i x^i \in \mathbb{Z}_p[X]$. We choose q_i uniformly at random in $\mathbb{Z}_p$, and we impose $q(0) = y$.

A share of the secret y is a couple $(x_i, y_i) = (x_i, q(x_i))$, with $x_i \neq 0$. We can build $n \leq p-1$ different shares. This sharing secret scheme is a (k, n)-threshold scheme.

Given k shares of y, it is easy to compute y. Let $(x_i, y_i)_{i\in\{0\dots k-1\}}$ be k different shares of the secret y. Using Lagrange's interpolation, it is possible to compute efficiently a polynomial P of degree at most $k-1$ such that $P(x_i) = y_i$. This polynomial is unique, and $\deg(q) = k-1$, so $P = q$. q is the interpolating polynomial built by Lagrange's method. Finally, y is computed with the formula $y = \sum_{i=0}^{k-1} y_i \lambda_i(0)$, where $\lambda_i(0) = \prod_{0\leq j\leq k j\neq i} \frac{x_j}{x_j - x_i}$.

We dont have any information about y with knowledge of $k-1$ shares. Suppose given $k-1$ shares $(x_i, y_i)_{i\in\{0\dots k-2\}}$. There is a unique polynomial P of degree at most $k-2$ which interpolates these points. The aim is to compute a polynomial whose degree is $k-1$ which interpolates these shares. There are p such polynomials, these are the $(X-a)P(X)$ for $a \in \mathbb{Z}_p$. The good polynomial q is one of them, but it was randomly generated, so, given a a, $Pr\left[q(X) = (X-a)P(X)\right] = \frac{1}{p}$. This proves that $k-1$ shares do not provide any relevant information about the polynomial $q(X)$ and the secret y.

Application

One-time revocation scheme Consider now the pay-per-view context. Suppose that a provider broadcasts an encrypted content, which should be only decrypted by allowed users and assume only one key is able to decrypt the broadcast. The group controller first broadcasts this key. For some reasons, the group controller might want to revoke one or many users. Revocation is simplily a broadcast of a new key for the encryption, such that revoked users can not compute the message anymore.

Naor and Pinkas [1] presented a way to revoke k users. The scheme operates over a field $\mathcal{F}$. Before the revocation, y, the future key, is generated, with a polynomial q of degree k, which will be used in a $(k+1, n)$-threshold scheme, taken at random. Each user u receives an unique identifier $I_u \in \mathcal{F}$, and a share $K_u = (I_u, q(I_u))$ of the next key y.

Suppose the group controller knows the identity of the k users $I_{u_1}, \ldots, I_{u_k}$ whose keys should be revoked. The group controller broadcasts the shares K_{u_i} to all users. Then each revoked user knows k shares of the new key y, which do not give any information about y. However all others users who still are allowed to decrypt the channel know $k+1$ shares, so they can compute y. The provider just has to encrypt the broadcast with this new key y.

Such a scheme is a good way to proceed one revocation of at most k users, and never a coalition of all the k revoked users can get any information about the new key. The only drawback, is that it can work only one time, because the polynomial is found. Else, we have to take a new polynomial and construct new keys.

[1] gives two ways to tranform this one-time revocation scheme in a many-times revocation scheme.

The first one is to generate a new instance of the scheme after each revocation, during a maintenance phase. A new polynomial corresponding to the new key will be used, as above, in a (k, n)-threshold scheme.

Another method is to keep the same polynomial but to share the secret in the exponent. We will describe this method below, that is used by [2].

A one-time attribute based encryption With sharing secret scheme presented above, we are already able to make an attribute-based encryption scheme.

Let $\mathcal{U}$ be a set of attributes, called the universe. Let p be a prime such that $p \geq |\mathcal{U}|$. First define the universe $\mathcal{U}$. For simplicity, we associate the elements of the universe (the attributes) with the first elements of ${\mathbb{Z}_p}^*$, namely the integers $1, \ldots, |\mathcal{U}|$.

Each user *holds* some attributes. A controller wishes to encrypt a message for a given set ω of attributes, in such a way that only users who holds all the attributes in ω could decrypt the message.

Denote y the key used to encrypt messages by the controller. We will use a $(2 \cdot |\mathcal{U}|, n)$-threshold scheme to share this key. Let q be a random polynomial of degree $2 \cdot |\mathcal{U}| - 1$,

such that $q(0) = y$.

At the initialisation phase, each user is given $|\mathcal{U}|$ shares of this secret. Suppose a user holds l attributes $\omega\prime = \{t_1, \ldots, t_l\} \subset \mathcal{U}$. This user is given $|\mathcal{U}|$ shares, which are

- $|\mathcal{U}| - l$ shares $(t, q(t))$, for all $t \in \mathcal{U} \backslash \omega\prime$,
- l other shares $(x, q(x))$ for random values x.

To cipher a message for the attributes $\omega \subset \mathcal{U}$, the controller will send the encrypted message with the key k, using a standard encryption scheme, and he will provide $|\mathcal{U}|$ shares of the secret y which are

- $|\omega|$ shares $(t, q(t))$ for all $t \in \omega$,
- $|\mathcal{U}| - |\omega|$ other random shares $(x, q(x))$.

Therefore, a user who holds the attributes $\omega\prime \supset \omega$ will have $2 \cdot |\mathcal{U}|$ shares of the secret :

- $|\mathcal{U}|$ shares corresponding to each attributes : the $|\mathcal{U}|-l$ given during initialization, and the l attributes given with the ciphertext,
- The random shares given during initialization (l) and the one given with the ciphertext ($|\mathcal{U}| - |\omega|$), which makes an amount of $|\mathcal{U}| - |\omega\prime| + |\omega|$ shares. We know that $\omega\prime \supset \omega$, as a consequence $|\omega\prime| = l \geq |\omega|$ and the number of random shares $|\mathcal{U}| - |\omega| + l$ is at least $|\mathcal{U}|$.

These $2 \cdot |\mathcal{U}|$ shares are enough to interpolate $q(X)$, so the user can compute y the key to decrypt.

By contrast, if a user is missing an attribute, he will not have the $|\mathcal{U}|$ shares corresponding to each attributes. Since he already knew the share of his missing attribute, the share given with the ciphertext does not provide a new share... Thereby, such a user will know at mot $2|\mathcal{U}| - 1$ shares of the key, and the $(2|\mathcal{U}|, n)$-threshold scheme used ensures that he does not have any information about the key.

Finally we can add fuzzy property to this scheme. Suppose the controller wishes to send messages that can be decrypted by users who hold at least $k \leq |\omega|$ attributes. Instead of giving $|\mathcal{U}| - |\omega|$ random shares in the ciphertext, he just has to give $|\mathcal{U}| - k$ random shares.

This attribute based encryption scheme is really not perfect. In the first place it is a one-time scheme : once an user has decrypted a ciphertext, he knows the secret key y, so he could decrypt any ciphertext. In the second place, if some users collude they can learn the whole attributes' shares, and then decrypt any ciphertext, even if none of the colluders could decrypt it alone...

We will see that the one-time drawback is omitted by the same method as the revocation scheme : sharing secret in exponent. However, the collision issue is tied to the fact that user's keys are not independent : we give directly shares corresponding to missing attributes. It would be nice if two users who collude do not get any more information. This problem will be solved later, by making an identity based scheme.

2.3.2 Sharing secret in exponent

Naor and Pinkas [1] present a way to share a secret in the exponent, inspired by Feldman. Let $\mathcal{F}$ be a field, and g be a generator of this field. Denote, as above, q a random polynomial of degree $k-1$. We suppose $q(0) = y$.

Given k shares $(x_i, y_i)_{i\in\{0\dots k-1\}}$ such that $y_i = q(x_i)$, it is easy to compute $g^y = g^{\sum_{i=0}^{k-1} y_i\lambda_i}$, with $\lambda_i = \prod_{0\leq j\leq k j\neq i} \frac{x_j}{x_j - x_i}$.

Remark that $g^y = g^{\sum_{i=0}^{k-1} y_i\lambda_i} = \prod_{i=0}^{k-1} g^{y_i\lambda_i}$ which is really essential in the following : it shows that given k shares (x_i, g^{y_i}), g^y can be computed easily, while $k-1$ of these shares are still not giving any information about g^y or y. We note that (x_i, y_i, g) is enough to recover a share (x_i, g^{y_i}).

Finally, a share $(x_i, q(i))$ in Shamir's sharing can be made $(x_i, g^{q(i)})$, a share of g^y in sharing secret in exponent scheme. Given g^r, a Shamir's share $(x_i, q(i))$ can also be made $(x_i, g^{rq(i)})$, that is a share of the secret g^{yr}. In other words, we can go to the exponent without any loss in our scheme. Even better : doing so provides us more security and freedom.

Recall the **Decisional Diffie-Hellman (DDH) assumption**, introduced before : suppose a challenger chooses $a, b, z \in \mathbb{Z}_p$ at random. The Decisional DH assumption is that no polynomial-time adversary is able to distinguish the tuple $(g^a, g^b, e(g,g)^{ab})$ from the tuple $(g^a, g^b, e(g,g)^z)$ with more than a negligible advantage. Thanks to sharing secret's method, we just got a new trap. Indeed k shares $(x_i, g^{rq(i)})$ are a trap to compute $g^{rq(0)}$.

Application Sharing secret in exponent will be useful to get a many-times revocation scheme, or a many-times attribute based encryption.

Many-times revocation scheme Recall that a provider broadcasts an encrypted channel. Allowed users know the key to decrypt the channel. The group controller can revoke at most k users by proceeding as follow :

Suppose that a new key y will be used after revocation by still allowed users. Let q denote a random polynomial of degree k such that $q(0) = y$.

Each user, identified by I_u is given a share of the secret $(I_u, q(I_u))$.

To proceed revocation of k users $u_1 \dots u_k$, the group controller broadcasts their share $(I_{u_i}, q(I_{u_i}))$. Thus revoked users will know k shares of the new key y, and non revoked users will know $k+1$ shares.

This describes a $(k+1, n)$-threshold scheme and proves that revocation will be effective.

Such a system is a one-time scheme, since nothing ensures that another revocation of k users will be effective. Suppose now that we work in a field $\mathcal{F}$ whose g is a generator, and for which the Decisional Diffie Hellman assumption holds. This means that no efficient algorithm can distinguish between the two distributions (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) where a, b, c are randomly chosen in $\mathcal{F}$. See [5] for further information about this assumption.

We construct a revocation scheme where $g^{rq(0)}$ is the new key. As above, let q denote a random polynomial of degree k. A share of $q(0)$ is given to each user u : $K_u = (I_u, q(I_u)) := (I_u, y_u)$. When the group controller wishes to revoke k users $u_1, \dots, u_k$, it picks uniformly a random value $r \in \mathcal{F}^*$ and sets the new key as $g^{rq(0)}$. The effective revocation is the broadcast of $(g^r, (I_{u_1}, g^{rq(I_{u_i})}), \dots, (I_{u_k}, g^{rq(I_{u_k})}))$.

A non revoked user u will know $k+1$ shares of the secret, because he can compute $(I_u, g^{rq(I_u)})$, using its own share that was given at initialisation. Then, using sharing secret in exponent, it is easy to compute $g^{rq(0)} = g^{r\sum_{i=0}^{k-1} y_i \lambda_i} = \prod_{i=0}^{k-1} g^{r y_i \lambda_i}$. However, a revoked user will only possess k shares of the new secret, and so he doesn't have any information about the new key. Moreover, a coalition of the k users gives only k shares.

If the revocation message $(g^r, (I_{u_1}, g^{rq(I_{u_i})}), \dots, (I_{u_k}, g^{rq(I_{u_k})}))$ is sent encrypted, the above system describes a many-times revocation of up to k users at each revocation. The group controller will just have to pick a new alea r to reuse the scheme.

Many-times attribute-based encryption Recall that we identify the universe as $\mathcal{U} = \{1, \dots |\mathcal{U}| mod p\}$. Let y be a private value. Let q be a random polynomial of degree $2 \cdot |\mathcal{U}| - 1$ such that $q(0) = y$. As earlier, q will be used to share y in a $(2 \cdot |\mathcal{U}|, n)$-threshold scheme.

An user who holds the attributes $\omega \subset \mathcal{U}$ is given during initialisation $|\mathcal{U}|$ shares of the value y.

- $|\mathcal{U}| - |\omega|$ shares $(t, q(t))$, for all $t \in \mathcal{U} \backslash \omega$,
- $|\omega|$ other shares $(x, q(x))$ for random values x.

In the one-time attribute based encryption, the controller encrypted a message with the key y which was decryptable only by users who holds all attributes in $\omega\prime \subset \mathcal{U}$:

- $|\omega\prime|$ shares $(t, q(t))$ for all $t \in T$,
- $|\mathcal{U}| - |\omega\prime|$ other random shares $(x, q(x))$.

Let g be a generator of the field used in this scheme, such that the DDH assumption holds for g and this field.

To transform this scheme in a many-times attributes based encryption, the controller picks a random value r and sends some shares of the secret y in exponent :

- g^r
- $|\omega\prime|$ shares $(t, g^{rq(t)})$ for all $t \in \omega\prime$,
- $|\mathcal{U}| - |\omega\prime|$ other random shares $(x, g^{rq(x)})$.

Remember with secret sharing in exponent, a share can be either (x_i, y_i) or (x_i, g^{y_i}) indifferently. Considering this definition of a share, a user who holds a set ω of attributes, can decrypt any ciphertext that the controller encrypted for a subset $\omega\prime \subset \omega$. The discussion 2.3.1 shows exactly that if $\omega\prime \subset \omega$, the $\omega\prime$'s user will get $2|\mathcal{U}|$ shares, and then will be able to compute g^{ry}. In the same way, if an user is missing an attribute, then he will now less than $2|\mathcal{U}|$ shares. Sharing secret in exponent is a $(2|\mathcal{U}|, n)$-threshold scheme, and the one-time scheme security was based on a *threshold scheme*, so the security here is the same.

Setup The scheme operates over a group $\mathbb{Z}_p$ of prime order. Let g be a generator of $\mathbb{Z}_q$, such that Decision Diffie Hellman assumption holds for $(g, \mathbb{Z}_q)$.

Let q be a random polynomial of degree $2|\mathcal{U}|$ over $\mathbb{Z}_q$. Denote $y := q(0)$.

Public values are q, g, g^y.

Extract Suppose a user holds $\omega \subset \mathcal{U}$ a set of attributes. Its private key is made of $|\mathcal{U}|$ components $(S_i)_i$. S_i is a share of secret y. These components are divided into two groups :

- $|\mathcal{U}| - |\omega|$ shares $(i, q(i))$, for all $i \notin \omega$.
- $|\omega|$ other shares $(x, q(x))$ for random values x.

Encrypt Suppose we want to encrypt a plaintext $M \in \mathbb{Z}_p$ for a set of attributes $\omega\prime \subset \mathcal{U}$. Pick randomly $r \in \mathbb{Z}_p$. Set $C = (g^r, M \cdot g^{ry}, S\prime_i)$, with $S\prime_i$ shares of the secret g^{ry} divided into two groups :

- $|\omega\prime|$ shares $(i, g^{rq(i)})$ for all $i \in \omega\prime$,
- $|\mathcal{U}| - |\omega\prime|$ other random shares $(x, g^{rq(x)})$.

Decrypt Let $C = (C_0, C_1, (S_i)_i)$ be a ciphertext encrypted with the set of attributes $\omega\prime$. Suppose we know the private of a user who holds ω attributes, with $|\omega \cap \omega\prime| \geq d$

Using its components $S_i = (i, q(i))$, compute $g^{rq(i)}$ gives $|\mathcal{U}|$ private shares of g^{ry}. By grouping shares from private key and shares from ciphertext, we get at least $2|\mathcal{U}|$ different shares of g^{ry}.

This scheme is secure assuming the DDH assumption. However some users can collude to be able to decrypt any ciphertext. The matter is in users' keys. $|\mathcal{U}|$ shares of

y are given to each user $(x_i, q(x_i))$. Let K and $K\prime$ be two keys, for attributes ω and $\omega\prime$ respectively. Colluding K and $K\prime$ gives a key which opens attributes $\omega \cup \omega\prime$. In order to avoid collusion, we have to make K and $K\prime$ uniformly independent. There should be one polynomial q_u per user, so two users who collude would get independent values $q_u(x_i), q_{u\prime}(x_j)$ that are uniformly independents.

What we need is to give different keys, even to users who holds the same attributes. This is an identity-based encryption scheme. Actually, this scheme is very close to an identity-based encryption scheme, defined below.

3 Identity-based encryption

Identity-based encryption (IBE) was asked by Shamir in 1984. It is a public key encryption scheme in which the public key can be an arbitrary string. Usually, private key is picked randomly and public key is built using the private key. In an IBE scheme, private key is any abritrary string.

The difficulty is to compute a private key from this arbitrary public key string, in such a way that the private key seems uniformly random. Such a scheme is defined with four algorithms :

setup that generates system parameters and a master-key,

extract to generate a private key given an arbitrary public key string and the master-key,

encrypt and **decrypt**, to encrypt messages using an arbitrary public key string and decrypt them with the corresponding private key, provided by extract.

The main drawback is that Key escrow is inherent to this model. An entity, called private key generator (PKG), is able to compute every private keys, using the master-key. Thus, PKG can decrypt any ciphertext.

Suppose that Alice wants to send a message to Bob. However, such an IBE scheme permits to avoid public key certificates management. In a classical public key cryptosytem, suppose Alice wants to send Bob an encrypted message. She first needs to obtain Bob's public key certificate to be able to properly encrypt the message. In an IBE scheme, to send Bob a message, Alice uses "bob@mail.com" as public key. Bob will have to contact the private key generator (**PKG**), the entity that knows the master-key, to obtain his private key, related to his public key "bob@mail.com".

Boneh and Franklin [4] presented an efficient IBE encryption scheme, based on bilinear maps. We will see first a little background on bilinear maps, and the assumptions related to them, then we will present the first IBE scheme [4], based on BDH assumption.

This first scheme is good to avoid public key certficates managements.

Next will be presented a scheme based on MBDH assumption. To use this scheme, it is necessary to manage public key certificates, but it appears kind of triggers. Finally, we will show to use these triggers, combined with sharing secret, to build a fuzzy IBE scheme, where identities are sets of attributes.

3.1 Bilinear maps

In this section we will define an admissible bilinear map over a group, and see two hypothesis Bilinear Diffie-Hellman (BDH) and Modified Bilinear Diffie-Hellman

(MBDH).

3.1.1 Definition

Let G_1 and G_2 be two groups of order prime p. Throughout this section, we see G_1 and G_2 as multiplicative groups. Let $e : G_1 \times G_1 \to G_2$ be a map between these two groups. We say that e is an admissible bilinear map if these three properties are satisfied :

- e **is bilinear**, which means that each partial application is linear. In particular, for all $g, h \in G_1$ an all $a, b \in \mathbb{Z}_p$, we have $e(g^a, h^b) = e(g, h)^{ab}$.
- e **is non-degenerate** : $\forall g \in G$, $e(g, x) = 0 \iff g = 0$ and $e(g, x) = 1 \iff g = 1$ (where 1 is the identity)
- e **is efficiently computable** : there exists a polynomial (in time) algorithm that computes $e(g, h)$ for any $g, h \in G_1$.

To apply these admissible bilnear maps, pairings over elliptic curves are generally chosen.

3.1.2 Some properties

As a consequence of e's bilinearity, and since G_1 is a cyclic group, e **is symmetric**. Indeed, if t is a generator of G_1, every element can be expressed as a power of t. By bilinearity from e, we thus have : $\forall a, b \in \mathbb{N}, e(t^a, t^b) = e(t, t)^{ab} = e(t^b, t^a)$.
Hence, $e(g, h) = e(h, g)$ for every $g, h \in G_1$.

The non-degenerescence of an admissible bilinear map e provides a way to compute a generator in G_2 given one in G_1. G_1 and G_2 are groups of prime order, so if $g \in G_1$ is a generator, then $e(g, g)$ is a generator of G_2.
As a matter of fact, if $e(g, g)^k = 1$ for some k, we have for all x $e(g^k, g^x) = e(g, g)^{kx} = 1$ so $g^k = 1$ by non-degenerescence. Thus, $k = p - 1$ because g is a generator.

The existence of such an admissible bilinear map makes the **Decision Diffie-Hellman (DDH) problem easy in** G_1.
As seen in previous section, the DDH problem is to distinguish between the two distributions (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where a, b, c are chosen randomly and $g \in G_1$. Given (g^a, g^a, g^c), $e(g^a, g^b) = e(g, g)^{ab}$, so if $g^c = g^{ab}$, then $e(g, g^c) = e(g, g)^{ab} = e(g^a, g^b)$.
We have indeed the following equivalence : $c = ab \bmod q \Leftrightarrow e(g, g^c) = e(g^a, g^b)$, that makes the DDH problem easy. However, it exists groups where Computational Diffie-Hellman problem, that is the problem to compute g^{ab} given randoms (g, g^a, g^b), is still hard in despite of DDH is hard in these groups. Some examples are presented by Joux and Nguyen [6].

3.1.3 Assumptions

Bilinear maps offer a main assumption, called **Decisional Bilinear Diffie-Hellman (BDH) assumption**. Suppose a challenger chooses $a, b, c, z \in \mathbb{Z}_p$ at random. The Decisional BDH assumption is that no polynomial-time adversary is able to distinguish the tuple $(g^a, g^b, g^c, e(g,g)^{abc})$ from the tuple $(g^a, g^b, g^c, e(g,g)^z)$ with more than a negligible advantage.

Recall the **Decisional Diffie-Hellman (DDH) assumption**. Suppose a challenger chooses $a, b, z \in \mathbb{Z}_p$ at random. The Decisional DH assumption is that no polynomial-time adversary is able to distinguish the tuple (g^a, g^b, g^{ab}) from the tuple (g^a, g^b, g^z) with more than a negligible advantage.

The BDH assumptions acts as DDH with one more level, one more degree. We can actually formalize these types of assumptions. Consider the following assumption, that we call ${prodDH^n}_{G_1,g,G_2,h}$ assumption.

Let G_1, G_2 be two groups of prime order p, whose generators are respectively g and h. Suppose a challenger chooses $(a_i)_{1 \leq i \leq n}, z \in \mathbb{Z}_p$ at random. The ${prodDH^n}_{G_1,g,G_2,h}$ assumption is that no polynomial-time adversary is able to distinguish the tuple $(g^{a_1}, \ldots, g^{a_n}, h^{\prod_{i \leq 1 \leq n}^{a_i}})$ from the tuple (g^a, g^b, h^z) with more than a negligible advantage

Both DDH and BDH assumptions are particular cases of the latter assumption. DDH assumption is ${prodDH^2}_{G,g,G,g}$ and BDH assumption is ${prodDH^3}_{G_1,g,G_2,e(g,g)}$.

We will see that ElGamal's cryptosystem is based on DDH assumption, and in the same way, the first IBE scheme, based naturally on BDH assumption. Then we will present another IBE scheme based on the called **Decisional Modified Bilinear Diffie-Hellman (MBDH) assumption**.

Suppose a challenger chooses $a, b, c, z \in \mathbb{Z}_p$ at random. The Decisional BDH assumption is that no polynomial-time adversary is able to distinguish the tuple $(g^a, g^b, g^c, e(g,g)^{\frac{ab}{c}})$ from the tuple $(g^a, g^b, g^c, e(g,g)^z)$ with more than a negligible advantage.

3.2 From ElGamal to IBE

We review a natural way to cipher a plaintext M : one-time pad. Let K be chosen randomly. To encrypt a plaintext M, K is used as a mask : $C = M \oplus K$. Knowledge of K is enough to decrypt : $M = C \oplus K$. In such a scheme, both encryptor and decryptor have to know K. This is a symmetric encryption scheme. Note that the operator is simplily the sum bit to bit, or equivalently, xor.

To build an encryption scheme over an assumption, we review what is hard to compute, and which mathematicals pieces are missing to compute it easily. For one-time pad, K is random, so it is hard to compute, but knowing K, it is easy to compute

K.

We want to use this mask technique with something else than a pure alea : a mathematical object that is hard to compute for an attacker, but easy for encryptor and decryptor, while both encryptor and decryptor's knowledges are differents. Based on DDH assumption, given g^r,g^s, it is hard to distinguish g^{rs} from random. However, there are easy ways to compute g^{rs}. Given r, g^s or s, g^r, it is easy to compute g^{rs}. In this case, mathematical pieces missing DDH to compute g^{rs} are s or r.

ElGamal cryptosystem uses g^{rs} as a mask. s is the private key, r is a random used during the encryption, and g^s, g^r are publicly known. An encryptor chooses r itself, so he knows g^s the public key, and r. He thus can compute g^{rs} and send a ciphertext $M = (g^r, M \cdot K)$, where $K = g^{rs}$. In the other part, knowledge of private key s is enough to compute K : g^r is indeed public.

This assymetric cryptosystem uses two dimensions of DDH assumption. One-time pad was in one dimension : K was used as a mask, and it is needed to know K to encrypt and decrypt. With another dimension, it is possible to separate encryptor and decryptor's knowledge : one dimension for the decryptor, one for the encryptor, who both publish g^r, g^s.

Recall that DDH assumption is the same as $prodDH^2{}_{G,g,G,g}$. We saw that BDH assumption provides a new dimension in the problem. Given g^r, g^s, g^t, it is hard to compute h^{rst}, with $h = e(g, g)$. BDH assumption is $prodDH^3{}_{G_1,g,G_2,e(g,g)}$. Maybe we can use the three dimensions offered to separate decryptor, encryptor and PKG. PKG's part of K, the mask, would be the master-key.

We will use $K = h^{rst}$ as the new mask.

If an encryptor is able to build h^{rst} then it will cipher like before : $C = (g^r, Me \cdot h^{rst})$.

Thanks to e's bilinearity, given g^{st}, g^r it is easy to compute $e(g^{rt}, g^r) = e(g, g)^{urs}$. Moreover, given g^s, g^t, r, it is easy to compute $e(g^s, g^t)^r = e(g, g)^{rst}$. The first IBE scheme [4] is based on these two ways to compute g^{trs}.

Let $ID \in \{0, 1\}^*$ be an arbitrary string, the name of a user. Let (G_1, G_2, e) be two groups of order p with an admissible bilinear map $e : G_1 \times G_1 \to G_2$. Choose a random generator $g \in G_1$.

We suppose there is a hash function $H_1 : \{0, 1\}^* \to G_1$. We will identify the user ID with $I_{ID} = H_1(ID) \in G_1$. Let $s \in \mathbb{Z}_q$ be the master-key, picked uniformly. $\mathcal{P} = g^s$ is publicly known.

The **extract** algorithm takes as inputs ID and s, and gives $d_{ID} = I_{ID}^s$. d_{ID} is ID's private key.

To **encrypt** a text M, let $r \in \mathbb{Z}_p$ be an alea. Encryption is computed as $C = (g^r, M \cdot e(I_{ID}, g^s)^r)$. Note that $K = e(I_{ID}, g^s)^r = e(I_{ID}, g)^{rs}$.

To **decrypt** a ciphertext $C = (C_1, C_2)$, the aim is to compute $K = e(I_{ID}, g)^{rs}$. The private key $d_{ID} = {I_{ID}}^s$ and $C_1 = g^r$ value are enough to compute $K = e(d_{ID}, C_1) = e(I_{ID}, g)^{rs}$.

The security is based on BDH in (G_1, G_2, e). Recall that g is generator of G_1, so it exists t_{ID} such that $I_{ID} = g^{t_{ID}}$. The scheme is secure until it is hard to compute $e(g, g)^{t_{ID} sr}$. In Boneh and Franklin's scheme, t_{ID} is anything : it depends on the hash function H_1, and its value is absolutely secret. We will have to suppose that H_1 is a random oracle, to be sure that I_{ID} does not leak any information about t_{ID}.

This cryptosystem is ideal to avoid public key certificates. If the hash function H_1 is public and secure, it is possible to cipher with any arbitrary string as public key. Now suppose that t_{ID} is chosen by the PKG. We loose the first IBE advantage, but we will gain a new degree of liberty.

3.3 An IBE cryptosystem based on MBDH

In this section, we will present a scheme based on MBDH, where public key and private key are chosen by the PKG.

Let G_1 and G_2 be two groups of prime order p, with $e : G_1 \times G_1 \to G_2$ an admissible bilinear map. Suppose that users are identified to elements of $\mathbb{Z}_p$. For each user i, PKG picks $t_i \in \mathbb{Z}_p$ randomly. This value is part of the master-key, when $T_i = g^{t_i}$ is published. PKG chooses also a global $y \in \mathbb{Z}_p$ randomly, and publishes $Y = e(g, g)^y$. Finally, master-key is $(t_0, \ldots, t_{p-1}, y)$, if there are p users in this system. Public global parameters are $(e, G_1, G_2, g, T_0, \ldots, T_{p-1}, Y)$.

The **extract** algorithm provides a private key $D_i = g^{\frac{y}{t_i}}$ to user i. To **encrypt** a text M, pick $r \in \mathbb{Z}_p$ randomly and computes $K = Y^r = e(g, g)^{yr}$. An encryption is of the form $C = (g^{t_i r}, M \cdot K)$.

To **decrypt** a ciphertext $C = (C_1, C_2)$, any user knows $g^{t_i r}, g^{t_i}, e(g, g)^y$. The aim is to compute $K = e(g, g)^{ry}$. With the private key D_i, it is possible to compute $e(D_i, C_1) = e(g, g)^{\frac{yrt_i}{t_i}} = K$, and so decryption is reached.

The security is obviously based on MBDH assumption. Given g^y, g^{rt_i}, g^{t_i}, it is hard to compute $e(g, g)^{\frac{t \cdot rt_i}{t_i}} = e(g, g)^{yr}$. How could a decryptor compute $e(g, g)^{ry}$ from g^{rt_i} and g_i^t ? Given g^x, it is possible to compute $e(g^x, grt_i) = e(g, g)^{rt_i x}$. We do want $rt_i x = yr$, so $x = \frac{y}{t_i}$.

We remark the private key $g^{\frac{y}{t_i}}$ is made of two components. y that was used to encrypt, and $\frac{1}{t_i}$, which acts as a trigger. A private key $D_j = g^{\frac{y}{t_j}}$ cannot be used to decrypt a ciphertext for i. Indeed, r is hidden by t_i, so g^{rt_i} can be exploited only with a trigger $\frac{1}{t_i}$. With D_j private key, ry will be hidden by a random value $\frac{t_i}{t_j}$.

This system is not very efficient, since we lost IBE's main advantage, that was avoid-

ing public key certificates management. However, this scheme can be made realistic if there are not many users. We can see these users as attributes, and find an application of this scheme.

Suppose a provide broadcasts an encrypted channel. Users are divided into two groups : AM and PM. The AM's group is allowed to decrypt the channel in the morning, and PM's groups in the afternoon. At each group is given one attribute.

All AM's member have got the same key : $g^{\frac{y}{t_{AM}}}$. In the morning, the provider has to encrypt its channel by providing $g^{rt_{AM}}$, and in the afternoon with $g^{rt_{PM}}$. It is a one-attribute based encryption.

Recall threshold schemes presented in first section. A (k, n)-threshold scheme is a scheme where a secret y is divided into n shares, such that k of them are enough to compute y efficiently, but $k-1$ of them do not provide any information about y. We want to turn the scheme based on MBDH into an attribute-based encryption, in such a way that a ciphertext is encrypted for some attributes, but can be decrypted by users who hold at least k of these attributes.

The actual private key for an attribute i is $g^{\frac{y}{t_i}}$. Suppose an identity is a set ω of attributes. Its private key would be $g^{\frac{y_i}{t_i}}$, where y_i is a share of the secret y. Each share of the secret would be triggered as above, and a ciphertext encrypted over a set of attributes $\omega\prime$ could be decrypted if and only if k shares have been triggered, wich means $|\omega \cap \omega\prime| \geq k$.

We are on the good way : these shares can be triggered, but how to compute the secret y effectively ?

3.4 A linear operation

We want to use sharing secret in exponent method. Let denote q a random polynomial of degree $k-1$ such that $q(0) = y$. We are looking for a way to compute $e(g,g)^{yr}$, given (x_i, g^{y_i}) a share of the secret, with $y_i = q(x_i)$. We define $\lambda_i = \prod_{j \neq i} \frac{x_j}{x_j - x_i}$ Lagrange's coefficients, for all $i \in \mathcal{U}$.

$$\begin{aligned} e(g,g)^y &= e(g,g)^{\sum_{i=0}^{k-1} y_i \lambda_i} \\ &= \prod_{i=0}^{k-1} e(g,g)^{y_i \lambda_i} \\ &= \prod_{i=0}^{k-1} e(g^{y_i}, g)^{\lambda_i} \end{aligned}$$

Our aim is not to compute $e(g,g)^y$ but $e(g,g)^{ry}$. As above, based on MBDH, we use a value t_i as a trigger :

$$e(g,g)^{ry} = \prod_{i=0}^{k-1} e(g^{\frac{y_i}{t_i}}, g^{rt_i})^{\lambda_i}$$

Finally, we can combine triggers method offered by bilinear maps, and secret sharing method, to make a fuzzy ibe scheme.

4 Fuzzy IBE scheme

4.1 Algorithm

In this section, we will give details of the fuzzy IBE scheme given in [2]. In this scheme, identites are sets of attributes. Each identity holds at least k attributes.

Let $\mathcal{U}$ be the universe, the set of all attributes. An identity is seen as a set of attributes $\omega \subset \mathcal{U}$. We wish to create a system in which a ciphertext encrypted over identity $\omega\prime$ can be decrypted by identity ω if and only if they have k common attributes, which means $|\omega \cap \omega\prime| \geq k$.

Setup Let G_1 be a group of prime oder p, and let g be a generator og G_1. Let G_2 be another group of order p and $e : G_1 \times G_1 \to G_2$ be an admissible bilinear map.

Let's define $\mathcal{U}$ the universe. There are $|\mathcal{U}|$ different attributes, and the universe is associated to the first $|\mathcal{U}|$ elements of ${\mathbb{Z}_p}^*$. Namely, the integers $1, \ldots, |\mathcal{U}| \mod p$.

Choose $t_1, \ldots t_{|\mathcal{U}|}, y$ uniformly at random from $\mathbb{Z}_p$. The master key is $(t_1, \ldots, t_{|\mathcal{U}|}, y$, and published parameters are $(e, G_1, G_2, T_1 = g^{t_1}, \ldots, T_{|\mathcal{U}|} = g^{t_{|\mathcal{U}|}}, Y = e(g,g)^y)$.

The t_i will serve to link polynomials to identities (t_i will be given for i identity owned by the reader).

Extract This algorithm is run by the PKG, who knows the master-key. Let $\omega \subset \mathcal{U}$ be an identity. First a $k-1$ degree polynomial q is randomly chosen in $\mathbb{Z}_p[X]$, such that $q(0) = y$. The private key is made of components $(D_i)_{i \in \omega}$, where $D_i = g^{\frac{q(i}{t_i}}$.

Encrypt To encrypt a message $M \in G_2$ over a public key $\omega\prime \subset \mathcal{U}$, first pick a random value $r \in \mathbb{Z}_p$. The ciphertext is then published as $E = (\omega\prime, E\prime = M \cdot Y^r, (E_i = {T_i}^r)_{i \in \omega\prime})$.

Decrypt Suppose a ciphertext E encrypted for identity $\omega\prime$, and suppose we have a private key for identity ω, the components $(D_i)_{i \in \omega}$. If $|\omega \cap \omega\prime| < k$, then decryption is not possible, by natur of our scheme. Now suppose $|\omega \cap \omega\prime| \geq k$. Let $S \subset \omega \cap \omega\prime$ be an arbitrary subset of d elements.

The k attributes in subset S will be used to trigger k shares of secret y, hidden in private key for identity ω. Let's define $\lambda_{i,S} = \prod_{j \in S j \neq i} \frac{j}{j-i}$ Lagrange's coefficients,

depending on the set S. Then we can compute

$$\begin{aligned} K &= \prod_{i \in S} e(D_i, E_i)^{\lambda_{i,S}} \\ &= \prod_{i \in S} e(g^{\frac{q(i)}{t_i}}, g^{rt_i})^{\lambda_{i,S}} \\ &= \prod_{i \in S} e(g,g)^{rq(i)\lambda_{i,S}} \\ &= e(g,g)^{r \sum_{i \in S} q(i)\lambda_{i,S}} \\ &= e(g,g)^{ry} \end{aligned}$$

The mask being computed, it is easy to get $M = E\prime \cdot K$.

4.2 Security's proof

4.2.1 Preliminaries

As we have seen above, Fuzzy IBE scheme is really powerful thanks to its flexibility. Given that we share at least k attributes in common with some identity, we will be able to decypher its messages. This point finds all its interest in biometrics, for instance, where the identity's acquisition (like an iris scan) is noisy but we want the person to be recognize eventhough. This is called "error-tolerance"

On the other hand, we have to ensure that this strength is not a weakness. For, people may try to exploit this adaptability to gain access to messages they should not be able to decypher.

One can think of a "collusion attack", which is an action carried out by a set of malicious users, in possession of protected content, joining together in order to obtain an unprotected asset.

This idea is essentially motivated by the key generation algorithm : for each of the attributes associated to a user's identity ω, and its polynomial q of finite degree $k-1$ (such that $q(0) = y$), it will provide D_i the secret key components via Lagrange's Interpolation. These components are the ones to match in order to decypher the message, but even with "enough" information, malicious users can not conjecture the polynomial of the private key identity.

This stands from the fact that D_i are tied to a fixed random polynomial so they can not combine their D_i to attack a single polynomial, i.e. it is resistant to collusion attacks.

What's more, we will see that the security of this model relies on Decisional MBDH assumption, and more generally on the difficulty of computing the discrete logarithm.

4.2.2 Fuzzy Selective-ID Model

Let us define the **Fuzzy Selective-ID Model** in which we are going to do the proof. Note that the Adversary will not be allowed to ask for private keys sharing more than k attributes with the identity ω, essentially because from a practical point of view, both are the same or k is not high enough to prevent the scanner to distinguish the two persons associated.

Consider the following protocol :

Init : Adversary wants to challenge the identity ω

Setup : Challenger generates public parameters

Phase 1 : Adversary is allowed to query for private keys from many identities $\gamma_1, ..., \gamma_n$ such that $\forall i \in \{1, ..., n\}, |\gamma_i \cap \omega| < k$ (i.e. private keys issued from sufficiently "far away" identities)

Challenge : Adversary submits messages M_0, M_1 of equal length.
Challenger takes at random $b \in \{0, 1\}$, encrypts M_b with ω, sends it to Adversary

Phase 2 : Phase 1 is repeated

Guess : Adversary outputs a guess b' of b.

We then define adversary $\mathcal{A}$'s advantage as $Adv(\mathcal{A}) = Pr(b' = b) - \frac{1}{2}$

A scheme is secure in the "Fuzzy Selective-ID model of security" if any polynomial-time adversary does not have more than a negligible advantage to guess correctly.

This describe the idea that the encrypted messages are well spread in the space, preventing people to classify it and, then, having informations on M (like its length, its Hamming's weigth,...)

4.2.3 Security's proof

Now we can prove our statement of security by contradiction.

Assume $\mathcal{A}$ is a polynomial-time adversary which can break through "Fuzzy Selective-ID Model" such that $Adv(\mathcal{A}) = \alpha$. Let us build a simulator $\mathcal{B}$ that can exploit $\mathcal{A}$'s properties to play the Decisional MBDH with advantage $\frac{\alpha}{2}$, which should be impossible by assumption.

The main idea behind is to do a reduction between the two problems (Selective-ID model and Decisional MBDH assumption), or in other words, between distinguishing two ciphertext and a MBDH-tuple from a random one. As a result, we are going to

take an adversary $\mathcal{A}$ that can do the first part with a non-negligible advantage and construct some algorithm $\mathcal{B}$ that can exploit $\mathcal{A}$ to gain an advantage in the MBDH problem.

On the first part of the proof, we set up the background of the Fuzzy-IBE Scheme (groups, universe, map) and the public parameters. Last ones are random but depends on the challenged identity ω.

In the second part, the adversary issues private key extraction from identities "far enough" from ω, and the challenger has to construct the private keys corresponding (with a random polynomial q, defined by its components D_i). $\mathcal{A}$ then provides two chosen plaintext (of same length), but only one of them is encrypted (chosen at random) by $\mathcal{B}$. The encryption is designed to prevent $\mathcal{A}$ to retreive the original message it comes from. After this, $\mathcal{A}$ goes back to Phase 1 and can issue private keys again.

Third part is the guess from $\mathcal{A}$ on the message encrypted, which is explained later on.

Selection $\mathcal{A}$ sets the groups G_1, G_2 in an universe U, e an admissible bilinear map, and takes at random $\epsilon_0 \in \{0,1\}$, a, b and c. Denote by (A, B, C, Z) the quadruplet (g^a, g^b, g^c, Z) where, if $\epsilon_0 = 0$, $Z = e(g,g)^{\frac{ab}{c}}$ and else, $Z = e(g,g)^z$. This $epsilon_0$ is unknown from $\mathcal{B}$, and serve in the Decisional MBDH assumption. Indeed, if $\epsilon_0 = 0$, in the challenge part the adversary will have an advantage (because it will not be just some random encryption of the messages) and will output a correct guess more frequently.

Init ω is the identity targeted by $\mathcal{A}$, given to $\mathcal{B}$

Setup Public parameters generated by $\mathcal{B}$ given to $\mathcal{A}$:
For all $i \in \omega$, $\beta_i \in \mathbb{Z}_p$ is taken at random. We set $T_i := C^{\beta_i} = g^{c\beta_i}$
For all $i \in \mathcal{U} - \omega$, $\omega_i \in \mathbb{Z}_p$ is taken at random. We set $T_i := g^{\omega_i}$
Set $Y := e(g, A) = e(g,g)^a$, denote by t_i the power of g in T_i.

Phase 1 As before, $\mathcal{A}$ asks for private keys γ sharing at most $d-1$ attributes with identity ω. Let $\mathcal{B}$ generate a private key for this identity, with a random polynomial q which matches :

Take a random set Γ of size $d-1$ of γ and containing the common attributes between γ and ω, a set denoted as Ω.

On Ω, $D_i := g^{s_i} = g^{\frac{q(i)}{t_i}} = g^{\frac{q(i)}{c\beta_i}}$ for randomly chosen $s_i \in \mathbb{Z}_p$, so $q(i) = c\beta_i s_i$
On $\Gamma - \Omega$, $D_i := g^{\frac{\lambda_i}{\omega_i}}$ for randomly chose $\lambda_i \in \mathbb{Z}_p$, so $q(i) = \lambda_i$
Lastly, we impose that $q(0) = a$.

To end the private key generation, $\mathcal{B}$ has to calculate D_i for all $i \in \gamma$, so we lack the ones for $i \notin \Gamma$. Since $\mathcal{B}$ knows the t_i and has defined q, he can easily interpolate his results, cutting the sum between "in Ω" part and "in $\Gamma - \Omega$" part (because t_i depends on i from ω or not).

If we write $\Gamma_0 = \Gamma \cup \{0\}$,
$\forall i \notin \Gamma$, $D_i = (\prod_{j\in\Omega} C^{\frac{\beta_j s_j \Delta_{j,\Gamma_0}(i)}{\omega_i}})(\prod_{j\in\Gamma-\Omega} g^{\frac{\lambda_i \Delta_{j,\Gamma_0}(i)}{\omega_i}})(Y^{\frac{\Delta_{0,\Gamma_0}(i)}{\omega_i}})$

Hence, $\mathcal{B}$ has constructed a private key for the identity γ i.e. its distribution is identical to that of the original scheme, because we have to make sure $\mathcal{B}$ respects the Fuzzy IBE scheme we want to prove the security.

Challenge $\mathcal{A}$ sends messages M_0, M_1 to $\mathcal{B}$
$\mathcal{B}$ takes at random ϵ_1 $in\{0,1\}$ and returns Encrypt(M_{ϵ_1})$= E := (\omega, E' := M_{\epsilon_1} Z, \{E_i := B^{\beta_i}\}_{i\in\omega})$
By definition of Z, we can distinguish two cases.

If $\epsilon_0 = 0$, let $r = \frac{b}{c}$. Then
$E' = M_{\epsilon_1} Z = M_{\epsilon_1} e(g,g)^{\frac{ab}{c}} = M_{\epsilon_1} e(g,g)^{ar} = M_{\epsilon_1} Y^r$
$\forall i, E_i = B^{\beta_i} = g^{\frac{b}{c} c\beta_i} = T_i^r$
Therefore, E is a random encryption, but, by hypothesis, $\mathcal{A}$ has an advantage in distinguishing ciphertexts coming from M_0, from M_1.

If $\epsilon_0 = 1$, since z is random,
$E' = M_{\epsilon_1} e(g,g)^z$ is totally random in G_2
As a result, no information is provided on the message from $\mathcal{B}$ (under the public key ω) and $\mathcal{A}$ has no clue on M_{ϵ_1}, so he will guess totaly at random, with probability $\frac{1}{2}$ (he has no advantage because E' does not contain Y and E_i, T_i).

Phase 2 same as Phase 1.

Guess
Like before, $\mathcal{A}$ has to output his guess ϵ_1' of ϵ_1.
The case of "correct guess" happens more often when $\epsilon_0 = 0$ because in this case, $\mathcal{A}$ outputs on something he has an advantage in. The simulator $\mathcal{B}$ will, hence, output $\epsilon_1' = 0$ to "say" that he thinks he was given a MBDH-tuple, and otherwise $\epsilon_1' = 1$ if he was given a random quadruple.

As a result, if $\epsilon_0 = 0$, $\mathcal{A}$ sees an encryption of M_{ϵ_1} and he has an advantage α in this situation. Hence, $Pr_{\epsilon_0=0}(\epsilon_1' = \epsilon_1) = \frac{1}{2} + \alpha$. The probability that $\mathcal{B}$ is correct in this case is not $\frac{1}{2}$ because it only depends on the probability $\mathcal{A}$ is correct, so $Pr_{\epsilon_0=0}(\epsilon_0' = \epsilon_0) = \frac{1}{2} + \alpha$

On the other hand, if $\epsilon_0 = 1$, $\mathcal{A}$ does not have any information on ϵ_1 because he has no advantage. Since he outputs at random, $P_{\epsilon_0=1}(\epsilon_1' \neq \epsilon_1) = \frac{1}{2} = P_{\epsilon_0=1}(\epsilon_0' = \epsilon_0)$ (for the same reason as above, said $\mathcal{B}$ guesses $\epsilon_0' = 1$ for the inequality).

We can now cross the results and evaluate the overall advantage of $\mathcal{B}$ in the Decisional MBDH game : $Adv(\mathcal{B}) = \frac{1}{2} Pr_{\epsilon_0=0}(\epsilon_0' = \epsilon_0) + \frac{1}{2} Pr_{\epsilon_0=1}(\epsilon_0' = \epsilon_0) - \frac{1}{2} =$

$\frac{1}{2}(\frac{1}{2}+\alpha)+\frac{1}{2}\frac{1}{2}-\frac{1}{2}=\frac{\alpha}{2}$.

Hence, if α is non-negligible, it goes all the same for the advantage of $\mathcal{B}$ in the Decisional MBDH game. This contradicts our assumption.

We have made a reduction from our initial problem to the Decisional MBDH game, so all the security of the scheme relies on this last game's security (which in return depends on discrete logarithm problem).

Let us note that all of this is true for the Fuzzy Selective-ID Model which is a chosen-plaintext model. Actually, some definitions and proofs are adapted to the case of a chosen-ciphertext model by Amit Sahai in "Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security". Moreover, no random oracles have been used in this proof, but one can thinks of using them with more standard techniques such as Fujisaki-Okamoto transformation.

5 Conclusion

In this paper we studied trapdoors and how they may be combined to create a new scheme which share some properties from its predecessors while providing us more security and freedom.

From the threshold scheme, based on the attributes, we saw a secret sharing method relying on the Lagrange's interpolation. This method introduced the possibility of user's revocation, since without enough shares, it is not possible to decipher, unless users collude.

Coming out of this problem, caused by user's keys, we presented an identity-based encryption, making the polynomials intrinsequely linked to the users. To do so, using bilinear maps has proven to be a good way combined with ElGamal's ideas. Secret was no more discovered (the scheme was no more a one-time scheme), revocation was possible, the protocol was finally secure (no need to access to a public key certificate anymore).

This point on, the Fuzzy IBE Scheme appears to be a new type of IBE, because it accounts a set overlap distance metric and finds applications in biometrics or sharing a document on an unsecure server.

In the particular case of the Fuzzy Selective-ID Model, we have made a reduction between the Fuzzy IBE scheme and a slightly different version of the Decisional BDH assumption. Doing so, we have seen that it is secure to collusion attack, i.e. malicious users can not cross their keys to decipher.

On the other hand, the scheme allows error-tolerance, which may be really usefull in practice : identity's acquisition is often noised and differ at each measure. Thus, we can set a degree of sharing properties to allow decryption or not.

However, from the definition of the algorithm, it goes out that the public parameters' size may be of a problem. As a matter of fact, it is linearly dependent in the number of attributes. [2] presented a "Large Universe Construction" in order to embed it rather to the parameter n (in practical, this will be the largest size identity encryptable), and be able to use collision-resistant hash functions or arbitrary strings instead of attributes.

References

[1] M. Naor, B. Pinkas, "Efficient Trace and Revoke Schemes", *Financial Cryptography*, pp. 1-20, 2001.

[2] A. Sahai, B. Waters, "Fuzzy Identity-Based Encryption", *EUROCRYPT*, pp. 457-473, 2005.

[3] A. Shamir, "How to share a secret", *Communications* ACM, Vol. 22, No. 11, 1979.

[4] D. Boneh, M. Franklin, "Identity-based encryption from the Weil pairing", in *Advances in Cryptology - Crypto 2001*, Lecture Notes in Computer Science, Vol. 2139, Springer-Verlag, pp. 514-532, 2001.

[5] D. Boneh, "The Decision Diffie-Hellman Problem", in *Proc. Third Algorithmic Number Theory Symposium - AsiaCrypt 2001*, Lecture Notes in Computer Science, Vol. 1423, Springer-Verlag, pp. 48-63, 1998.

[6] A. Joux and K. Nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups", J. Cryptology 16(4), pp. 239-247, 2003.