Mediator-Based Immediate Attribute Revocation Mechanism for CP-ABE in Multicast Group Communications

Lyes Touati, Yacine Challal

Mediator-Based Immediate Attribute Revocation Mechanism for CP-ABE in Multicast Group Communications

Lyes Touati
Sorbonne universités, Université de Technologie de Compiègne,
CNRS, Heudiasyc UMR 7253, CS 60 319,
60 203 Compiègne cedex.
Email: lyes.touati@hds.utc.fr

Yacine Challal
Centre de Recherche sur l'Information
Scientifique et Technique CERIST,
05 Rue des Frères Aissou, Ben Aknoun
Algiers, Algeria
Email: ychallal@cerist.dz

***Abstract*—Attribute Based Encryption (ABE) scheme is a mechanism that allows implementing cryptographic fine grained access control to shared information. It achieves information sharing of type one-to-many users, without considering the number of users and their identities. However, original ABE systems presents some drawbacks, especially the non-efficiency of their attribute/key revocation mechanisms.**

Based on Ciphertext-Policy ABE (CP-ABE) scheme, we propose an efficient proxy-based immediate private key update for multicast group communications. Our solution does require neither re-encrypting cipher-texts, nor affecting other users (Updating secret keys).

The proxy that has been introduced plays the role of a necessary semi-trusted assistant during the decryption process without taking decisions about who is eligible or not to decrypt data.

Finally, we demonstrate that our scheme guarantees security requirements that we target and we also show through analysis that our scheme achieves effectively its goals.

***Index Terms*—CP-ABE; Access Control; Pairing Cryptography; Attribute revocation; Multicast group communications**

I. Introduction

Ciphertext-Policy Attribute Based Encryption [1] is a concept that allows to implement a fine grained cryptographic access control to shared data. The main idea is to associate a list of attributes to each user relative to her/his role in the application. A secret key is constructed by an Attribute Authority (AA) based on the user's attributes set. In ABE systems, data are encrypted upon a policy in a form of a logical expression using AND, OR, and threshold gates, and implying attributes from the universe of attributes. Users whose attributes set satisfies the access policy of a cipher-text are allowed to decrypt it using their secret keys.

The advantage of ABE schemes is the possibility to apply fine grained access control on data without relying a third party (server, cloud service provider, etc.). The access control is cryptographically ensured.

Y. Challal is associate professor at Ecole Nationale Supérieure d'Informatique (ESI, Algiers, Algeria). He is member of Systems Design Methods lab. (LMCS)

However, ABE schemes are criticized for some drawbacks bound to the complexity of their construction. The first drawback is the overhead in term of execution time and energy consumption, especially for encryption and decryption primitives. The second one, which is more challenging, is the key/attribute revocation problem. The latter is a very tricky problem as many users can share the same attribute, and when we want to remove an attribute from one user's secret key, it is very difficult to not affect other users sharing the same attribute. The objective of attribute/key revocation mechanism is to be able to efficiently update users' secret keys (adding and/or removing attributes) easily without affecting (or with a minimum effect) non-concerned users.

The rest of the paper is organized as follows. Section II discusses previous works on attribute/key revocation mechanism of ABE schemes. We review some necessary background notions in Section III. We introduce our solution in Section IV. Security and performance analysis are discussed in Section V and Section VI respectively. Finally, we conclude the paper in Section VII.

II. Related Works

The attribute revocation problem is a very tricky issue as attributes could be shared by many users, hence, revoking or adding one attribute from/to a user naively requires to renames that attribute and update all secret keys containing that attribute [1]. Therefore, scalability represents an important property to consider when designing such solutions.

There are mainly three approaches in the literature to resolve the attribute revocation issue for CP-ABE scheme.

The first category of solutions is the naive one, it is based on renaming attributes by adding the next expiration date at the end of the attribute [1]. This solution induces a huge overhead due to the attributes update and re-keying.

Always in the same category, L. Touati et al. proposed two efficient solutions [2] [3] for the attribute revocation issue of CP-ABE. Indeed, instead of renaming the attributes, they proposed to split the time axis into time slots, and they introduced a new hash function that takes two parameters: an

attribute and a time slot identifier. In the first solution [2], time slots are all with the same duration, this may generates a delay (lag) in the key delivery and revocation. The second one [3] is more flexible and eliminates the undesirable lag. Time slots are with variable durations and are constructed considering users' attribute validity periods. However, in this solution, the Attribute Authority must know beforehand all users' attribute validity periods to determine time slots durations. This seems to be more or less a hard assumption depending on the application we used it for.

The second category aims to integrate the revocation mechanism into the access policy [1]. This would come out with huge access trees as the key revocation condition is transformed into integer comparisons. This kind of approach supports only key revocation but no attribute revocation.

The third category of solutions based on the use of the Proxy Re-Encryption technique (PRE) [4]. [5], [6] are example of solutions of this category. As their name indicates it, these solutions introduce a proxy or powerful cloud server in order to absorb the overhead due to re-encryption of ciphertexts. In other words, for each decryption, a end user must request the ciphertext from a proxy, the later will re-encrypt it so as the user could decrypt it if and only if he is authorized.

In [7], Touati et al. proposed a mediator based attribute revocation solution for CP-ABE. The role of the mediator is to assist users during decryption processes. However, targeted security requirements are not adequate for group/multicast communication applications. In this paper, we propose an variant of [7] in order to implement a CP-ABE *immediate* key update mechanism (immediate attribute/key revocation mechanism) guaranteeing multicast group communication security requirements. In such applications, users still can access to old encrypted data when their secret keys are updated, and of course, their previous secret keys are not usable for future cipher-texts.

III. Background

In this section, we present some necessary notions to the well understanding of the rest of the paper. First, we recall the principle of ABE systems and list the primitives of CP-ABE scheme. After that, we present the notions of access tree and bilinear maps.

A. Bilinear Maps

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. the bilinear map e has the following properties:

1) Bilinearity: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e\left(u^a, v^b\right) = e\left(u, v\right)^{ab}$.
2) Non-degeneracy: $e\left(g, g\right) \neq 1$.

We say that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map e are both efficiently computable. Notice that the map e is symmetric since $e\left(g^a, g^b\right) = e\left(g, g\right)^{ab} = e\left(g^b, g^a\right)$.

B. CP-ABE scheme

Ciphertext-Policy Attribute-Based Encryption [1] is a powerful asymmetric encryption mechanism that allows implementing a fine-grained access control in cyber-physical systems. CP-ABE proposes to construct private keys based on a list of attribute that are assigned beforehand to an entity. Data are encrypted with an access policy in the form of an access tree. only those whose private key satisfies this access policy can decrypt the cipher-text.

It consists mainly of four primitives:

- **Setup**. The setup primitive is run by the Attribute Authority at the bootstrap phase. It takes no input other than the implicit security parameter. It outputs the public parameters PK which is shared with all the entities of the system, and a master key MK which is kept secret.
- **KeyGen**(MK, S). It is run by The Attribute Authority to generate secret keys to system's users. It takes the master key MK and list of attributes S. The keyGen primitive outputs a secret key SK corresponding to the list of attribute S.
- **Encrypt**(PK, M, γ). The encryption algorithm takes as input the public parameters PK, a message M, and an access structure γ over the universe of attributes. The primitive encrypt M and produce a cipher-text CT which could be decrypted only by a user that possesses a set of attribute satisfying the access structure γ.
- **Decrypt**(PK, CT, SK). It takes as parameters the public parameters PK, a cipher-text CT and a secret key SK which is a secret key for a set S of attributes. If the list of attributes S verifies the access policy defined in CT, the primitive decrypt the ciphertext and outputs the original message M.

C. Access tree

The access trees (Figure 1) are used in CP-ABE to describe the access policy with which a message is encrypted. Only user with an attributes sets satisfying the access tree could decrypt the cipher-text.

Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

Some functions are defined to facilitate working with access trees:

- *parent(x)*: denotes the parent of the node x in the tree.
- *att(x)*: is defined only if x is a leaf node, and denotes the attribute associated with the leaf node x in the tree.
- *index(x)*: denotes the order of the node x between its brothers. The nodes are numbered from 1 to num.

Satisfying an access tree. Let T be an access tree with root r. Denote by T_x the sub-tree of T rooted at the node x. Hence T is the same as T_r. If a set of attributes γ satisfies the access tree T_x, we denote it as $T_x\left(\gamma\right) = 1$. We compute $T_x\left(\gamma\right)$ recursively as follows. If x is a non-leaf node, evaluate

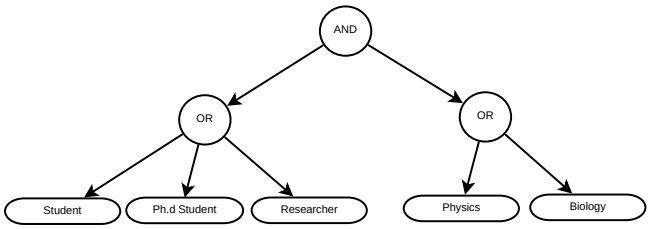

Figure 1: Example of an access tree

$T_{x'}(\gamma)$ for all children x' of node x. $T_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $T_x(\gamma)$ returns 1 if and only if $att(x) \in \gamma$.

IV. Our Solution

In this section, we present our solution. First, we motivate our work and give some application examples for our solution. After that, we itemize the security requirements and present the network model. Then, we detail our solution implementing an attribute revocation mechanism for CP-ABE scheme.

A. Motivations and Application Cases

In this paper, we tackle the attribute/user revocation issue of CP-ABE in application cases having specific properties and security requirements. In these applications, users gaining new attributes, hence new secret key, must be prevented from using the new secret key to access old encrypted data, his new secret key is valid only for future encrypted data. Likewise, a user losing some attributes, hence getting new secret key, could not use old secret keys to decrypt future cipher-texts requiring some of the lost attributes.

In other words, the validity period of a user's key is bounded by its delivery (beginning of the validity) and its revocation (end of the validity). A user's key can only decrypt cipher-texts which are encrypted during its validity periods.

We can cite "MultiCast Group Communication" as a kind of applications that require these security properties. Many applications are part of MultiCast Group Communications. For example: *Online Network Games*, *Video on Demand*, *Chat rooms*, *TVoD (Encrypted TV)*, etc.

- Chat rooms: Chat rooms applications require a confidentiality of communications. Users joining chat rooms gain attributes that allow them to decrypt exchanged messages. The chat room managing system must prevent new users from accessing old messages sent before their membership. Further, users quitting a chat room lose all access rights to future communications between the members.

- TVoD (Encrypted TV): One of the applications that we target is Encrypted TV. Users subscribe to channels and programs. Users gain the access to the channels and programs only from the moment they subscribe (Backward secrecy). Likewise, when the validity of the subscription expires or the service provider decides to stop the subscription, the users lose the access to future programs (Forward secrecy).

B. Security Requirements

In this section, we present the security requirements that our mechanism must verify in order to be validated. The applications we target in this paper require specific security properties which are summarized below.

- **Backward Secrecy.** A user receiving new secret key with new attributes is not able to use it to decrypt old messages.
- **Forward Secrecy.** A user receiving a new secret key after losing some attributes has no access to futures cipher-text requiring the lost attributes even if these cipher-texts' policies are satisfied by a previous secret key.
- **Collusion resistant.** This is a very important property of Attribute Based Encryption. It means that the conspiracy of many non-authorized users for decrypting a cipher-text is useless, even if the union of their attributes sets satisfies the encryption policy defined for the cipher-text.
- **Immediate Key Update.** When a user's attributes set is updated (adding and/or removing attributes), a new secret key must be constructed based on the new user's attributes set and must be delivered for that user immediately.
- **User Privacy.** The attribute management mechanism must preserve user privacy. Information concerning a user like its attributes list must be kept secret from a third party. The data accessed by a user also must be hidden from other users.

Notation	Description
PK	Public Key generated by the Attribute Authority
MK	Master Key generated by the Attribute Authority
SK	User's Secret Key generated by the Attribute Authority based on user's set of attributes
$SK^{(1)}$	Part of SK sent to the corresponding user
$SK^{(2)}$	Part of SK sent to the proxy
$PrSK$	Proxy Secret key
$PrPK$	Proxy Public Key
T_{enc}	date of encryption
DD_i	Delivery Date (timestamp) of a secret key part D_i
$\{.\}_{-k}$	The content between accolades is encrypted with the key k
M	Message to be encrypted
γ	access structure defining the access policy to the message M
CT	The cipher-text got after encryption of M with the policy γ
$K_{p,i}$	Symmetric key shared between the proxy and an entity or a user i

Table I: Notation Table

C. Network model

We assume the existence of a semi-trusted powerful proxy that assists entities in the decryption process. We mean by semi-trusted that the proxy is curious but honest: it honestly executes the tasks assigned to it, however, it could try to learn any possible information about encrypted data using the elements given to it. The proxy receives parts of entities' secret keys during the key generation phase. A special entity called *Attribute Authority* manages users' attributes and creates users' secret keys. This entity has also the role of holding the public parameters and it is responsible for the revocation mechanism. All other entities are considered as users of the system.

We assume that after the initialization phase, each user U_i in the system shares a symmetric key $K_{p,i}$ with the proxy. We assume also that the proxy has received a couple of keys (secret key $PrSK$ and public key $PrPK$) constructed using a public-key cryptosystem like RSA [8]. The proxy public key is shared with all users in the system.

The Proxy must keep information about users. The data structure in Table II shows which information the proxy keeps about a given user U_i.

Secret key part	Delivery date
$D^{(1)}$	DD_1
$\vdots$	$\vdots$
$D^{(i)}$	DD_i
$D^{(i+1)}$	DD_{i+1}
$\vdots$	$\vdots$
$D^{(n)}$	DD_n

Table II: Proxy data structure

In the first column we have the secret key part D_i and the corresponding delivery date DD_i is in the second column.

Figure 2 shows the global architecture of our solution illustrating the different involved parties. It shows also the exchanges between these entities. The *Attribute Authority* is responsible for generating the *Public Key* (PK) and sharing it with the *Data Owners (DO)* (**Step 1**). It also generates to each user a *Secret Key* (SK) based on the user's attributes set (**Step 2**). The secret key SK is divided into two parts $SK^{(1)}$ and $SK^{(2)}$. The first part ($SK^{(1)}$) is sent to the concerned user, and the other part ($SK^{(2)}$) to the proxy. The latter insert $SK^{(2)}$ in its data structure (See Table II) along with the timestamp of receipt.

Meanwhile, the data owner is able to encrypt his sensitive data and stores them in a remote server (**Step 3**). We recall that, the storage server is not charged of ensuring access control to data; but rather, it is cryptographically implemented by CP-ABE.

The user gets the cipher-text from the storage server (**Step 4**), he solicits the assistance of the proxy to decrypt it (**Step 5**).

The key update process is similar to the key generation (**Step 6**). For more information (see Section IV-G).

D. Assumptions

Let e be a non-degenerate bilinear pairing. Two assumptions related to bilinear maps are listed below and used to construct our solution.

- The Fixed Argument Pairing Inversion 1 (FAPI-1) [9]: Given $D_1 \in \mathbb{G}_1$ and $z \in \mathbb{G}_T$, compute $D_2 \in \mathbb{G}_2$ such that $e(D_1, D_2) = z$.
- The Fixed Argument Pairing Inversion 2 (FAPI-2) [9]: Given $D_2 \in \mathbb{G}_2$ and $z \in \mathbb{G}_T$, compute $D_1 \in \mathbb{G}_1$ such that $e(D_1, D_2) = z$.

We have also set up some assumptions related to the network model:

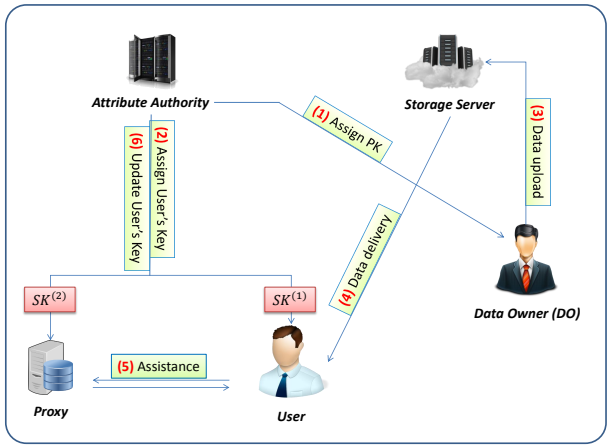

Figure 2: Architecture of the solution

1) Each entity of the system shares a symmetric key with the proxy.
2) The proxy possesses a unique Proxy Secret Key $PrSK$ and its corresponding Proxy Public Key $PrPK$ is published and is known by all system entities.
3) The proxy is semi-trusted (honest but curious): It executes its function honestly without disclose information to other parties. However, it could be curious to get as much as possible of information about data.

E. Basic idea

We split the secret key into two parts, the first one, $SK^{(1)}$ which contains all elements related to the attributes is sent to the user. The second part $SK^{(2)}$ which represents the element D, is sent to the proxy along with the time of key update (Delivery date) DD. The proxy maintains all the history of users' keys.

We bind the h^s with the time of encryption T_{enc} in the ciphertext CT by encrypting them with the proxy public key $PrPK$. We finally get $C = \{h^s, T_{enc}\}_{\_PrPK}$.

During the decryption, the user has to solicit the proxy to completely recover the plain-text. She/he sends C to the proxy. The latter decrypt is using $PrSK$ and then, extract the T_{enc}. Using T_{enc} the proxy finds the element D_i to use. Then, it computes $e(h^s, D_i)$ and sends it back to the user. To find the D_i, the proxy can do it by dichotomy. The i of D_i must satisfy this property:

$$DD_i \leq T_{enc} < DD_{i+1} \tag{1}$$

F. Scheme

In this section, we describe in detail the different primitives and how our solution achieves attribute revocation.

- Setup. The setup primitive generates the public key PK and the master key MK of the system. It is run by the Attribute Authority during the bootstrap phase. The primitive chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g.

Then it chooses two random exponents $\alpha, \beta \in \mathbb{Z}_p$. The public key is published as:

$$PK = G_0, g, h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha}. \quad (2)$$

and the master key is:

$$MK = (\beta, g^{\alpha}). \quad (3)$$

- KeyGen(MK,S).

This primitive is run by the Attribute Authority to generate users' secret keys. It takes as input the master key MK and user's attributes set S. It outputs two parts $SK^{(1)}$ and $SK^{(2)}$. The first one is given to the corresponding user, and the other one is sent to the proxy.

The algorithm first chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. It computes two parts as:

$$SK^{(1)} = \left(\forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}\right) \quad (4)$$

and

$$SK^{(2)} = D = g^{(\alpha+r)/\beta} \quad (5)$$

- Encrypt(PK,γ,M).

The encryption primitive encrypts a message M under the tree access γ. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the access tree γ. These polynomials are chosen in the following way in a top-down manner, starting from the root R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$.

Starting with the root node R the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x.

Let Y be the set of leaf nodes in γ. The cipher-text is then constructed by giving the tree access tree γ and computing:

$$CT = \Big(\gamma, \tilde{C} = Me(g,g)^{\alpha s}, C = \{h^s, T_{enc}\}\_{PrPK}, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\Big) \quad (6)$$

- Decrypt(CT,SK).

During the decryption process, an decryptor uses the *DecryptNode* function defined in [1] to compute $A = DecryptNode(CT, SK, r) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$. Where r represents the root node of the access tree γ defined for CT.

The decryptor requires $e(h^s, D)$ value to proceed the decryption, he sends $C = \{h^s, T_{enc}\}\_{PrPK}$ to the assisting proxy. The latter uses proxy secret key $PrSK$ for decryption et gets h^s and T_{enc}. Then it choose the adequate D to use according to T_{enc} (Formula 1). The proxy computes $R = e(h^s, D)$ and sends it back to the decryptor after encryption using the shared key between them.

Now, the decryptor can retrieve the original message this way:

$$\begin{aligned}\tilde{C}/(e(R,D)/A) &= \tilde{C}/(e(h^s, D)/A) \\ &= \tilde{C}/\left(e\left(h^s, g^{(\alpha+r)/\beta}\right)/e(g,g)^{rs}\right) \\ &= M \end{aligned} \quad (7)$$

The decryption primitive succeeds if and only if the set of attributes associated with SK satisfies the access policy γ defined for CT. Otherwise, the primitive fails and returns nothing.

G. Revocation Operation

In order to revoke a user or some attributes to a user, the Attribute Authority has only to geenrate a new secret key corresponding to the new attributes set of the use. Then, it sends $SK^{(1)}$ to the user and $SK^{(2)}$ to the proxy. The later will add an entry to the table associated to that user in a form of ($D^{(n+1)} = SK^{(2)}$, DD_{n+1}).

From this moment, the user could not use an old secret key to decrypt future ciphertexts. Indeed, future ciphertexts include a time of encryption greater than the last delivery date $DD_{n+1} \leq T_{enc}$, so the proxy will use the corresponding $D^{(n+1)}$ which is compatible only with the new secret key $SK^{(1)}$ recently generated.

V. Security Analysis

In this section, we give proofs that our revocation mechanism meets the security requirements defined in Section IV-B

Proposition 1. *Our scheme guarantees Backward secrecy.*

Proof:

Once the Attributes Authority updates the user's secret key, the new secret key is usable only for ciphertexts with a timestamps greater than its delivery date.

We may believe that the user could cheat the proxy by constructing a fake C element. He could choose a timestamps in the interval of an old key, but the value of h^s is hidden to user. Hence, he cannot forge a fake C element of the cipher-text.

Proposition 2. *Our scheme guarantees Forward secrecy.*

Proof:

Once the user's secret key is updated: the user receives elements related to the new attributes set, and the proxy receives from the Attribute Authority the D_{i+1} element at DD_{i+1}. The previous user's key cannot be used to decrypt messages encrypted after DD_{i+1}.

Th user cannot forge a fake C element with an old time of encryption, as the latter is composed of h^s and the time of encryption T_{enc}, all encrypted with the proxy public key $PrPK$. Indeed, the user is not aware of the h^s as it is hidden by encryption.

Proposition 3. *Our scheme prevents users from collusion.*

Proof:

The collusion resistance is a property ensured the ABE scheme construction. Thanks to the random elements r, r_j,

secret keys cannot be used together as they are randomized. For more information about collusion ressistance property, we invite the reader to take a look at the original paper [1].

As our solution is based on CP-ABE scheme, this property is also verified.

Proposition 4. *Our scheme ensures immediate users' keys update.*

Proof:

Once the Attribute Authority decides to update a user's secret key, it just had to generate a new CP-ABE secret key and as detailed in Section IV-F and securely send the two parts to the concerned user and the proxy. Since then, the user's secret key is effectively updated and his previous secret key is no longer usable for future cipher-texts.

Proposition 5. *Our scheme ensures users' privacy.*

Proof:

The user's privacy in question here is about his attributes set and the cipher-texts he intend to decrypt. Indeed, the proxy has no idea about the list of attributes of any user, as he possesses only the element $SK^{(2)} = D$ related to the secret key which gives no information about the attributes set.

Likewise, during the decryption process, the proxy receives only the two elements $\{h^s, T_{enc}\}$ encrypted with his public key $PrPK$. These two elements don't say much about the cipher-text itself except the time of encryption.

VI. Performance Analysis

In this section, we analyze the performances of our solution in terms of computation and storage costs. Then, we give a highlight on how we can speed up the search at the side of the proxy.

A. Computation performance

Setup and Key Generation primitives: Our solution executes the *setup* and *keygen* primitives exactly as the original CP-ABE [1] except that it splits the secret key into two parts and sends them to the user and the proxy. It does not require more computation overhead.

Encrypt primitive: The encryption primitive of our solution is pretty similar to the original one, it just adds a timestamps T_{enc} to the cipher-text and replaces the element C by the concatenation of h^s and T_{enc} all encrypted with the proxy public key $PrPK$. Our solution does not generate much overhead for the data owner.

Decrypt primitive: The Decrypt primitive of our solution requires an exchange between the user and the proxy. The user sends the element C whose size equals $|\mathbb{G}_1| + |T_{enc}|$ and receives and element from $\mathbb{G}_T$.

The proxy has to look for the adequate D_i in the proxy table (Formula 1), it can use the dichotomy method to speed up the search. The user also has to find the adequate $SK^{(1)}$ to use depending on T_{enc}.

It is important to notice that our scheme allows to distribute the proxy overhead (both computation and storage) on several proxies without loosing in security level. In this case, all the proxies must possess $PrSK$, and each proxy is assigned to a group of users to assist them.

B. Storage performance

The storage overhead is an inherent drawback of our solution, as the proxy and the user have to store all old secret key parts. For the user, he can keep only the most recent key part $SK^{(1)}$ if he is sure that he does not need to access old messages.

VII. Conclusion

In this paper we presented a new attribute revocation mechanism for ABE schemes that verifies security requirements of some applications like multi-cast group communication. Our solution introduces a proxy in the system that assists the user during the decryption process, and hence, the attribute revocation process does affect only concerned users.

In order to achieve backward and forward secrecy, we impose to both users and proxy to store different versions of the secret keys parts. Indeed, users may access to old encrypted messages if they are illegible.

For our best of knowledge, our solution is the first solution that tackles the attribute revocation with group communication backward and forward secrecy assumptions.

VIII. Acknowledgment

This work was carried out and funded in the framework of the Labex MS2T. It was supported by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02).

References

[1] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2007, pp. 321–334.

[2] L. Touati and Y. Challal, "Batch-Based CP.-ABE with attribute revocation mechanism for the internet of things," in *2015 International Conference on Computing, Networking and Communications, Wireless Networks Symposium (ICNC'15 WN)*, Anaheim, USA, Feb. 2015.

[3] ——, "Efficient cp-abe attribute/key management for iot applications," in *IEEE International Conference on Computer and Information Technology*, Liverpool, United Kingdom, Oct. 2015.

[4] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *In EUROCRYPT*. Springer-Verlag, 1998, pp. 127–144.

[5] Z. Xu and K. Martin, "Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, June 2012, pp. 844–849.

[6] S. Jahid and N. Borisov, "Piratte: Proxy-based immediate revocation of attribute-based encryption," *arXiv preprint arXiv:1208.4877*, 2012.

[7] L. Touati and Y. Challal, "Instantaneous Proxy-Based key update for CP-ABE," in *41st Annual IEEE Conference on Local Computer Networks (LCN 2016)*, Dubai, United Arab Emirates (UAE), Nov. 2016.

[8] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[9] S. Galbraith, F. Hess, and F. Vercauteren, "Aspects of pairing inversion," *Information Theory, IEEE Transactions on*, vol. 54, no. 12, pp. 5719–5728, 2008.