Instantaneous Proxy-Based Key Update for CP-ABE

Lyes Touati, Yacine Challal

Instantaneous Proxy-Based Key Update for CP-ABE

Lyes Touati
Sorbonne universités,
Université de Technologie de Compiègne,
CNRS, Heudiasyc UMR 7253,
CS 60 319, 60 203 Compiègne cedex. France
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 has been proposed to implement cryptographic fine grained access control to shared information. It allows to share information of type *one-to-many* users, without considering the number of users and their identities. However, original ABE systems suffer from the non-efficiency of their attribute revocation mechanisms.**

Based on Ciphertext-Policy ABE (CP-ABE) scheme, we propose an efficient proxy-based immediate private key update which does require neither re-encrypting ciphertexts, nor affect other users' secret keys. The semi-trusted proxy assists nodes during the decryption process without having ability to decrypt users' data.

Finally, we analyze the security of our scheme and demonstrate that the proposed solution outperforms existing ones in terms of generated overheard.

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

I. Introduction

Access Control is a security service that allows to grant or deny the permission for a user to access some resource. It becomes a compulsory security service to prevent attacks against sensitive applications (such as Internet of Things (IoT) applications) that would have a deep impact and great damages on the systems. Users' privacy is another issue that requires fine-grained access control to avoid access to private information by third parties.

John Bethencourt et al. proposed the first construction of Ciphertext-Policy Attribute-Based Encryption in [1]. CP-ABE uses a set of attributes to define users' access scope and access structures are used to define the access policy to encrypted data. The attribute revocation is the mechanism by which one or more attributes are eliminated from the set of attributes of a specific user. Attribute revocation is a tricky issue [1], as the same attribute could be shared with many other users, and it is very difficult to update a user's key without affecting other users.

In this paper, we introduce new security requirements for a kind of applications and developed a proxy-based approach to efficiently achieve a real-time attribute revocation for CP-ABE by updating only concerned users' secret keys.

The rest of the paper is organized as follows. Section II introduces some background notions. We discuss related works in Section III. The proposed solution is presented in Section IV. We discuss security and performance analysis in Section V and Section VI respectively. Finally, we conclude our paper in Section VII.

II. Background

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.

B. CP-ABE scheme

CP-ABE [1] allows to implement fine-grained access control, it consists mainly of four primitives:

- **Setup**. It is run by the Attribute Authority at the bootstrap phase. It outputs the public parameters PK and a master key MK.
- **KeyGen**(MK, S). It is run by the Attribute Authority to generate users' secret keys. It takes the master key MK and a set of attributes S. It outputs a secret key SK corresponding to the set S.
- **Encrypt**(PK, M, γ). The encryption algorithm takes as input the public parameters PK, a message M, and an access structure γ. The primitive encrypts M and produces a ciphertext CT using γ.
- **Decrypt**(PK, CT, SK). It takes as parameters the public parameters PK, a ciphertext CT and a secret key SK. If the set of attributes S of SK verifies the access policy defined in CT, the primitive succeed and outputs the original message M.

C. Access tree

The access trees defines access policies. Each non-leaf node of them represents a threshold gate *k out-of n*, where k represents a threshold, and n represents the number of node's children ($1 < k_x \leq num_x$). Where num_x is the number of children of a node x, and k_x is its associated threshold value. Each leaf node x of the tree is described by an attribute and $k_x = 1$.

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. Children of a node y are numbered from 1 to num_y.

Satisfying an access tree. Let γ be an access tree with root r. γ_x is the sub-tree of γ rooted at the node x. If a set of attributes S satisfies the access tree γ_x, we denote it as $\gamma_x(S) = 1$. We compute $\gamma_x(S)$ recursively as follows. if x is a non-leaf node, evaluate $\gamma_{x'}(S)$ for all children x' of node x. $\gamma_x(S)$ returns 1 if and only if at least k_x children return 1. if x is a leaf node, then $\gamma_x(S)$ returns 1 if and only if $att(x) \in S$.

III. Related Works

Some solutions were proposed in the literature to tackle the problem of the attribute/key revocation.

Access Policy Based Revocation. This solution was proposed in [1] by J. Bethencourt et al. which consists in expressing the revocation condition in the access tree. This enlarges access trees, and therefore, the overhead considerably increases.

Renaming Attribute Based Revocation. M. Pirretti et al. proposed in [2] an idea implement attribute revocation for CP-ABE. The principle is to rename attributes by concatenating them with their corresponding expiration dates after each revocation. This approach induces a heavy overhead since all entities will be affected by the revocation.

Proxy Re-Encryption Based Revocation. It is a kind of solutions that consist in using a Proxy Re-Encryption technique (PRE [3]): A proxy is introduced to absorb the overhead due to the re-encryption. It is assigned a special secret key that allows it to re-encrypt data to a decrypt-able ciphertext for only authorized users (For example PIRATTE [4]).

IV. Our Solution

A. Motivations and Application cases

This paper tackles the attribute revocation issue of CP-ABE in a particular kind of applications where the encryption time is not significant. The only thing that matters is the ciphertext's access policy and the current users' attributes list. Which means that, a user gaining some new attributes has the access right to all data encrypted with an access policy satisfied with the new user's set of attributes even if the data was encrypted before updating the user's attributes list. Likewise, a user who loses some attributes will not have the access right to data whose access policy is not satisfied by the new user's attributes list. **- Medical files management system:** is an illustration of such application case: each patient has his own medical file that lists all information about his health (Medical history, prescribed medication, ...). These information must be kept secret in order to protect user's privacy. When a doctor has to examine this patient, he needs patient's medical file, so he must be allowed to access to the whole medical file.

B. Security Requirements

1) **Definition 1: Attribute-Based Backward and Forward Accessibility.** After gaining new attributes, a user is able to decrypt all *old* and *future* ciphertexts encrypted with a policy which is satisfiable by his new set of attributes.
2) **Definition 2: Attribute-Based Backward and Forward Secrecy.** After losing one or many attributes, a user must have no access to *old* or *future* ciphertexts decryptable with its previous private key if its new attributes set doesn't satisfy the encryption policy.
3) **Collusion Resistance.** It means that the conspiracy of many non-authorized users for decrypting a ciphertext is useless, even if the union of their attributes sets satisfies the encryption policy of the ciphertext.
4) **Immediate Key Update.** All the changes made in a user's attributes set must take effect immediately.
5) **User's privacy.** The attribute management mechanism must preserve user's privacy.

C. Overview of our solution

We have introduced a semi-trusted proxy which maintains for each user a part of his secret key. The proxy is necessary for ciphertexts decryption process. The attribute revocation (key update) requires only the attribute authority to generate a new key based on the new user's set of attributes. The revocation is immediate once the proxy receives the part of the new key, the previous user's key will no longer be usable for decryption.

D. Network model

We assume the existence of a powerful semi-trusted proxy in the network (curious but honest). The network also contains a special entity called *Attribute Authority* that manages users' attributes and creates users' secret keys. All other entities are considered as users of the system.

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 crypto-system like RSA. The proxy holds in its memory the list of all users' identities and the symmetric keys $K_{p,i}$.

E. Assumptions

Let e be a non-degenerate bilinear map.

- The Fixed Argument Pairing Inversion 1 (FAPI-1) [5]: 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) [5]: Given $D_2 \in \mathbb{G}_2$ and $z \in \mathbb{G}_T$, compute $D_1 \in \mathbb{G}_1$ such that $e(D_1, D_2) = z$.

F. Scheme

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be a bilinear map (see Section II-A) The scheme consists of four primitives:

- **Setup**. The setup primitive is run by the Attribute Authority to generate a Public Key PK and a Master Key

Table I: Summary of notations.

Notation	Description
PK	Public Key generated by the Attribute Authority
MK	Master Key generated by the Attribute Authority
S	User's attributes set
SK	Secret key constructed by *Keygen* primitive
$SK^{(1)}$	User's part of the secret key SK
$SK^{(2)}$	Proxy's part of the secret key SK
M	Message to be encrypted
CT	Ciphertext obtained from M after encryption
$\{.\}_{-k}$	The content is encrypted with the k
$PrPK$	Asymmetric proxy public key
$PrSK$	Asymmetric proxy secret key
$K_{p,i}$	Symmetric key shared between the proxy and the i^{th} user
γ	Access tree with which the message M is encrypted
Y	Set of leaf nodes in the access tree γ
N	Number of users in the system
n	Number of attributes in the user's attributes set S

MK. It chooses a bilinear group $\mathbb{G}_1$ of prime order p. Then, it chooses two random $\alpha, \beta \in \mathbb{Z}_p$. The primitive outputs PK and MK:

$$PK = G_1, g, h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha} \quad (1)$$

$$MK = (\beta, g^{\alpha}) \quad (2)$$

- **KeyGen**(MK,S). It takes as input the user set of attributes S. The primitive chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. It constructs the secret key exactly as in [1] (Formula 3). Instead of giving it entirely to the user, the Attribute Authority splits it into two parts (Formula 4): the first one $SK^{(1)}$ contains all D_j and D'_j and it is for the user. The second one $SK^{(2)}$ consists only of D and it is sent to the proxy.

$$SK = \Big(D = g^{(\alpha+r)/\beta}, \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}\Big) \quad (3)$$

$$SK^{(1)} = \left(\forall j \in S : D_j, D'_j\right). \quad SK^{(2)} = D. \quad (4)$$

- **Encrypt**(PK,γ,M).
  The encryption primitive encrypts a message M under the tree access γ. The algorithm is very similiar to the one defined in [1]. First, it chooses a polynomial q_x for each node x in γ in the in a top-down manner, starting from the root R. For each node x, set the degree $d_x = k_x - 1$ of q_x.
  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 define q_x. Let Y be the set of leaf nodes in γ. The ciphertext is then constructed computing:

$$CT = \Big(\gamma, \tilde{C} = Me(g,g)^{\alpha s}, C = \{h^s\}_{-PrPK}, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\Big) \quad (5)$$

  The plaintext of h^s is encrypted with the Proxy Public Key $PrPK$.
- **Decrypt**(CT,SK). The decryption primitive decrypts the ciphertext CT using the private key SK which is associated with a set S of attributes. This primitive uses the *DecryptNode* function defined in [1], then sets $A = DecryptNode(CT, SK, r) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$. After that, the user sends the value of C encrypted with the symmetric key $K_{p,u}$ to the proxy. The proxy recovers h^s by double decrypting the user's message using $K_{p,u}$ and $PrSK$ respectively. Then, it computes $e(h^s, D)$ and sends it back to the user.
  Once the user receives $e(h^s, D)$, he can proceed to the decryption by computing the message M this way:

$$\tilde{C}/\left(e(h^s, D)/A\right) = \tilde{C}/\left(e\left(h^s, g^{(\alpha+r)/\beta}\right)/e(g,g)^{rs}\right) = M \quad (6)$$

V. Security Analysis

Proposition 1. *In our scheme, users require the proxy to be able to decrypt any ciphertext.*

proof:

Our scheme forces the user to solicit the proxy for computing $e(h^s, D)$ during the decryption process. In addition, the FAPI-1 assumption (Section IV-E) prevents the user from computing D using the intermediate result $e(h^s, D)$ calculated by the proxy. Furthermore, as the value of h^s is encrypted with $PrPK$, it is hidden to users. The FAPI-2 assumption (Section IV-E) ensures also that the user cannot get h^s from $e(h^s, D)$. Hence, it is not possible for any user to find a correlation between old h^{s_i} $(i = 1, 2, ...)$ and a new h^x.

Proposition 2. *Our scheme achieves Attribute-Based Backward and Forward Accessibility and Attribute-Based Forward and Backward Secrecy (Section IV-B).*

proof:

As soon as the Attribute Authority updates the user secret key and sends $SK^{(2)}$ to the proxy, and $SK^{(1)}$ to the concerned user, the latter is able to decrypt all old and future ciphertexts with an access policy satisfied by his new attributes set. Hence, Attribute-Based Backward and Forward Accessibility are verified (Definitions 1 Section IV-B). On another side, the user's previous key is no longer usable as the part $SK^{(2)}$ is updated in the proxy side. Indeed, all what the user could do with it is to compute $e(g,g)^{rs}$ using her/his $SK^{(1)}$. The r in the exponent is relative to the secret key, and it appears only in proxy's part $SK^{(2)}$. Therefore, the result $e(g,g)^{rs}$ is unusable as it is randomized with r.

In addition, as the proxy's secret part is updated, Proposition 1 ensures that it is no longer possible for the user to use his previous attributes set to decrypt any ciphertext.

Proposition 3. *Our scheme achieves collusion resistance property as defined in section IV-B.*

proof:

CP-ABE secret keys are constructed in a such way that collusion is not allowed. Indeed, each secret key is randomized

with a random number r (Equation 3). As our scheme is based on CP-ABE, this property is also included.

Proposition 4. *Our solution ensures the user privacy (data and attributes set): The user's attributes set and data he wants to decrypt are hidden to the proxy and to any third party.*

proof:

All the communications between the user and the proxy are encrypted with symmetric shared key (Section IV-E: Assumption 1). Therefore, any third party cannot have any information about the exchanges.

The part of the secret key sent to the proxy $SK^{(2)}$ contains only the element D. It reveals no information about the user's attributes set.

Proposition 5. *Our scheme does not allow to a curious proxy to use users' privileges to access data.*

proof:

The proxy possesses only a part $SK^{(1)} = D$ of the original CP-ABE user's secret key SK. This part all alone is useless during the decryption process as it needs the elements D_j and D'_j associated to each attribute.

VI. Performance Analysis

We have chosen the pairings parameters defined in the file "a" [6] to analyze the performances of our solution. Sizes of elements are given in the table II.

Table II: Size of elements

	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_t$	$\mathbb{Z}_p$
Size (bytes)	132	132	132	24

A. Size of the proxy table

Let N be the number of the users managed in the system. The identifier of a user could be codified with $\log_2(N)$ bits. We assume that the symmetric key is a AES key codified in 128 bits. The size of D ($D \in \mathbb{G}_2$) is 132 bytes. In conclusion we have:

$$Size\,(Table) = (128 + |D| + \log_2(N)) \cdot Nbits. \quad (7)$$

Table III: Required storage size for the proxy

N	10	200	400	600	800	1000
Size (kB)	1.45	29.1	58.24	87.4	116.57	145.75

The size of the table does not have to bother much as our protocol allows to spread the decryption assisting overhead (storage and computation) across multiple proxies. Thereby, the load of computation and storage upon one proxy is lightened.

B. Size of User's Private Key

The user's secret key contains an integer n (Encoded in four (4) bytes) representing the number of attributes, and $n = |S|$ couples of elements $< D_j, D'_j >$ from $\mathbb{G}_2$ and $\mathbb{G}_1$ respectively. The average size of attributes is represented as a. CP-ABE SK contains also an element D from $\mathbb{G}_2$ (This element is stored by proxy in our solution). The private key in PIRATTE has in addition n elements D''_j from $\mathbb{G}_1$.

The table IV summarizes the size of the user's secret key for the three schemes. We notice that, our solution has the lowest size for each pairing parameter.

Table IV: Size of the user's secret key

Secret key size (bytes)		
CP-ABE [1]	PIRATTE [4]	Ours
$136 + (a + 264)n$	$136 + (a + 392)n$	$4 + (a + 264)n$

C. Decryption cost

The decryption process in our solution requires only one extra exchange (One sending and one reception) of messages between the user and the proxy. The overhead of symmetric cryptography is negligible compared to CP-ABE operations. Our solution allows the user to save up one pairing operation as the latter is executed by the proxy.

The original CP-ABE [1] is not concerned. As for PIRATTE [4], for every leaf node in the access tree, the user sends an element C'_y of $\mathbb{G}_2$ and receives from the proxy an element of $\mathbb{G}_2$ and another of $\mathbb{Z}_p$. Table V summarizes the sizes of exchanged messages. Our solution widely overcomes PIRATTE [4] in term of exchanged messages overhead.

Table V: Size of exchanged messages during decryption

Size of exchanged messages (bytes)					
CP-ABE [1]		PIRATTE [4]		Ours	
Sent	Received	Sent	Received	Sent	Received
-	-	$132 \cdot \|Y\|$	$156 \cdot \|Y\|$	132	132

VII. Conclusion

In this paper, we have proposed a proxy-based attribute/key revocation mechanism for ABE.

Our solution achieves efficiently immediate attributes/key revocation without affecting not concerned users. The only critic that it could be made to our solution is the required storage size in the proxy side; Nevertheless, our solution is able to spread the proxy's burden on many proxies.

VIII. Acknowledgment

This work was carried out in the framework of the Labex MS2T (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] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 99–112.

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

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

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

[6] "Pbc library: The pairing-based cryptography library," https://crypto.stanford.edu/pbc/, accessed: 2016-04-09.