Batch-Based CP-ABE with Attribute Revocation Mechanism for the Internet of Things

Lyes Touati, Yacine Challal

Batch-Based CP-ABE with Attribute Revocation Mechanism for the Internet of Things

Lyes Touati and Yacine Challal
Université de Technologie de Compiègne - CNRS
Heudiasyc UMR 7253. BP 20529
60205 Compiègne, France
Email:{lyes.touati,ychallal}@hds.utc.fr

***Abstract*—Ciphertext-Policy Attribute-Based Encryption (CP-ABE) is an extremely powerful asymmetric encryption mechanism, it allows to achieve fine-grained access control. However, there is no solution to manage efficiently key/attribute revocation problem in CP-ABE scheme. Key revocation problem is very important in dynamic environment like Internet of Things (IoT), where billions of things are connected together and are cooperating without human intervention. Existing solutions are not efficient due to their overhead (traffic) and complexity (big access trees). Other solutions require the use of powerful semi-trusted proxies to re-encrypt data.**

The proposed solution in this paper called Batch-Based CP-ABE reduces the complexity and the overhead, and does not require extra nodes in the system. We propose to split time axis into intervals (time slots) and to send only the necessary key parts to allow refreshing the secrets keys. An analysis is conducted on the way to choose the best time slot duration in order to maximize system performances and minimize average waiting time.

***Index Terms*—CP-ABE, Internet of Things, Access Control, Attribute Revocation, Batch-Based,**

I. Introduction

Internet of things [1] is a new paradigm where many objects that surround us are provided with a unique identifier and sensing, actuating, computation and communication capabilities. These objects can be Radio-Frequency IDentification (RFID) tags, sensors, actuators, smart-phones, etc. All these entities will be able to interact with each other and cooperate together to reach common goals. It is expected that more than 50 billion devices will be connected to the Internet by 2020 (sensors, smart-phones, laptops, cars, clothes, wristwatches, etc.). The Internet of Things is an enabling technology for several applications like smart cities, domotic and home automation (smart homes), smart grid, smart healthcare and remote monitoring etc.

In the Internet of Things, all objects communicate between them without human intervention; Therefore, securing IoT communications is a tricky challenge, especially since security protocols developed for the internet, like IPsec or TSL, are not suitable for IoT environments. Indeed, these protocols have a high overhead that cannot be supported by constrained devices in the Internet of Things.

Attribute-based encryption (ABE) is a public key encryption mechanism that allows users to encrypt and decrypt messages based on descriptive user attributes. There are two main versions of ABE: Ciphertext-Policy Attribute-Based Encryption [2] and Key-Policy Attribute-Based Encryption [3]. In KP-ABE, attributes are used to describe the encrypted data and policies are built into user's keys; while in CP-ABE, the attributes are used to describe a user's private key, and an encryptor determines a policy on who can decrypt the data and include it in the encrypted data.

Ciphertext-Policy Attribute-Based Encryption [2] is an extremely powerful asymmetric encryption mechanism, it is a promising cryptographic solution to enforce access control in the Internet of Things. Using CP-ABE allows keeping encrypted data confidential even if the storage server is untrusted.

Key revocation, or more generally, attribute revocation is a challenging issue in the CP-ABE scheme since there are many entities that might match the decryption policy.

In this work, we define a batch-based CP-ABE with an efficient attribute revocation mechanism. Our solution reduces the complexity and the overhead comparing to other solutions, and it does not require proxies to re-encrypt messages.

The remainder of this paper is organized as follows. We discuss related works in section II. In section III we present some preliminaries of our work. We define the AMVVI problem in IV. Our solution is presented in V. Section VI presents a study of the impact of time slot duration on system performances. We conclude our paper in section VII.

II. Related Works

In [4], authors gave an idea on how attribute revocation could be implemented. The main idea of their proposition is to add expiration date to each attribute. Once this expiration date comes, the attribute authority rename the attribute and send it to all entities in the system, it regenerates all the secret keys to the non-revoked users (the revocation is materialized by not receiving a new secret key including the renamed attribute).

In [2], authors propose a way to transform numerical attribute to non-numeric attribute, by this way, they can express conditions on dates and include the revocation condition into the access tree. But this transformation makes access trees bigger than before, thus, the overhead considerably increases.

In [5], authors addressed user revocation and key refreshing issue for CP-ABE in data-owner-centric environments like

those for cloud storage. Their solution named DURKR uses the proxy re-encryption mechanism (PRE [6]), it only considers user revocation and requires a cloud storage provider to re-encrypt a data for every user request.

In [7], authors considered data storage and delivery system. authors proposed a solution that consists on encrypting a data to be shared with symmetric key and splitting it into several slices via an (n,n) secret sharing scheme.

In [8], Yu et al. tried to resolve the challenging issue of key revocation in CP-ABE by considering practical scenarios like data sharing in which semi-trustable on-line proxy servers are available. Their solution integrates Proxy Re-Encryption (PRE [6]) technique with CP-ABE and enables the authority to revoke user attributes and to delegate laborious tasks to proxy servers. This solution necessitates to regenerate all users secret keys and re-encrypting data after every change occurred in the system.

III. Preliminaries

In this section we review some basic concepts as well as notions related to CP-ABE scheme [2].

A. Access Structure

Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $A \subseteq 2^{\{P_1,P_2,\cdots,P_n\}}$ is monotone if $\forall B, C$: if $B \subseteq C$ then $C \in A$. an access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) A of nonempty subsets of $\{P_1, P_2, \cdots, P_n\}$, i.e., $A \subseteq 2^{\{P_1,P_2,\cdots,P_n\}} \setminus \{\emptyset\}$. the sets in A are called the authorized sets, and the sets not in A are called the unauthorized sets.

B. Access tree

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 $T_x(\gamma) = 1$. We compute $T_x(\gamma)$ recursively as follows. if x is a non-leaf node, evaluate $T_{x'}(\gamma)$ 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 $att(x) \in \gamma$.

C. 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, $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. 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 $e\left(g^a, g^b\right) = e(u, v)^{ab}$.
2) Non-degeneracy: $e(g, g) \neq 1$.

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 $e\left(g^a, g^b\right) = e(g, g)^{ab} = e\left(g^b, g^a\right)$.

D. Lagrange coefficient

Let i be an element in $\mathbb{Z}_p$, and S a set of elements in $\mathbb{Z}_p$. We define the Lagrange coefficient as:
$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} (x - j) / (i - j)$.

E. Revocation in ABE schemes

In Attribute Based Encryption systems, attribute revocation is a challenge issue even more difficult then in Identity-Based Encryption because each attribute is conceivably shared by multiple users. Revocation of any single attribute for a user would affect other users who share that attribute.

IV. AMVVI: Attributes Management with Variable Validity Intervals

A. AMVVI Problem: Definition and Motivation

Let us consider a system where users unpredictably join and leave the system in a completely asynchronous and dynamic way. Users attributes validity periods are then unpredictable. We mean by AMVVI, the problem consisting of managing keys (granting and/or revoking attributes access) that supports variable validity intervals of relating attributes. In order to optimize and significantly reduce the complexity and the number of exchanged messages, a more practical solution is to allow the Attribute Authority (AA) to handle simultaneously a number of attribute-based access policy changes. This can be achieved by splitting time into intervals (referred to as *time slots* or simply *slots*) and letting the AA handle all attribute-based access policy changes that occur in the same interval at the beginning of the next one.

This solution may cause a delay for entities when joining the system. Indeed, an entity which requests a secret key for some attributes at the middle of a time slot must wait until the beginning of the next time slot to get its own secret key and effectively join the system. System performances closely depend on the choice of the time slot duration: a small time slot forces the Attribute Authority to frequently refresh users' keys and consequently increase the overall system overhead. Conversely, a high time slot increases the delay and reduces users' keys refresh.

The choice of the time slot duration is an important and a critical issue as it directly influences the overall system performances. Generally, the time slot duration depends on

the application scenario, it is very common to choose a day or a month as time slot.

Example: Consider a system with three entities (users) i.e. $U = \{U_1, U_2, U_3\}$, and a set of three attributes: $A = \{Att_1, Att_2, Att_3\}$ (Figure 1).

With a color continuous line (red, blue or green) we depict the *asked* validity period of each attribute for system's users. Below each continuous line, we represented with dotted lines the *delivered* validity period of the attribute.

We notice here, that an attribute with a validity period starting at the middle of a time slot has its delivered validity period delayed until the beginning of the following time slot, ex. the first attribute (red line) of the first user $U1$ in figure 1. Similarly, if the asked validity period of an attribute ends at the middle of a time slot, the delivered validity period ends at the beginning of the following time slot, ex. second attribute (green line) of the fist user $U1$ in figure 1.

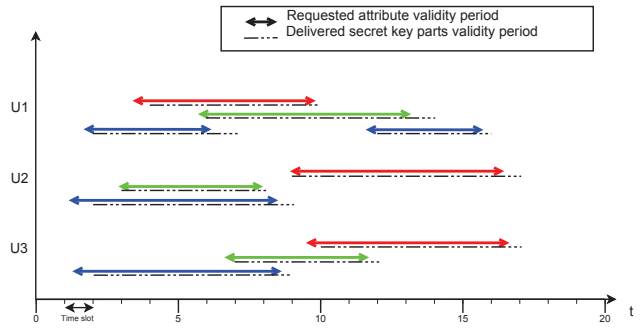

Figure 1: Example

B. Security requirements

In our solution, we aim to target the following security requirements:

- **Backward secrecy:** A new user joining the system should not have any access to previous encrypted data.
- **Forward secrecy:** A former user should have no access to current and future ciphertexts.
- **Collusion free:** Collusion resistance is a required property of any ABE system. Even if many users not satisfying the access policy collude, they can obtain no information about the plaintext of the ciphertext.
- **Data Confidentiality:** Unauthorized users who do not have the required attributes satisfying the access policy of a ciphertext must be prevented from accessing the plaintext of the data.

C. Application scenario

The application scenario that motivates our solution is security management of exchanged messages and data in a hospital. In this case, attributes could be administrative grades (Director, Department Chief, Secretary, Employee, ... etc.), Departments (Cardiology Department, Neurology Department, Emergency Department, ... etc.), functional grades (Nurse, Doctor, Trainee, ... etc.).

A trainee who has finished his/her internship in a hospital must see its secret key revoked, more precisely, the system should revoke its "Trainee" attribute. Thus he/she can not use its secret key part related to the attribute "Trainee" to decrypt ciphertexts. Similarly, a nurse who has moved from cardiology department to emergency one must lose her abilities to decrypt ciphertexts destined to cardiology department employee, this is resulting in revoking her "Cardiology Department" attribute.

Attribute revocation is an important task in Attribute Based Encryption systems. This is because systems are generally dynamic and are changing over time, so we cannot consider entities with a predefined and static sets of attributes during throughout the system life.

V. Our Approach

In this section, we present our approach. First, we present our motivations and the concept of our approach. Then, we define assumptions and system model.

A. Concept

Here we describe the basic idea of our solution to implement attribute revocation mechanism in CP-ABE scheme.

As we mentioned in section IV, our approach is a *batch-based* method, which means that time axis is split into intervals of the same duration called *time slots*, and policy access changes (granting and/or revoking access) occur only between two successive time slots.

To implement attribute revocation in CP-ABE, our solution is not based on renaming attributes or using access tree to include a policy that considers expiration time for an attribute as proposed in [2] and [4]. In our approach, the Attribute Authority send only the necessary attribute key parts every time slot to allow an entity to refresh its secret key. This technique reduces the overhead and the complexity of the solution comparing to the existing ones.

B. Syntax

Our Batch Based CP-ABE scheme consists of the following primitives:

- **Setup**. The setup algorithm 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 kept secret.
- **KeyGen***(MK, S)*. The KeyGen primitive is run by the Attribute Authority for each user joining the system. It takes as input the master key MK, a set of triplet S. Each element of the set S contains three information: an attribute $att \in D$ and the validity period materialized by T^{att}_{begin} and T^{att}_{end} [1].
- **Encrypt***(PK, M, γ, T)*. The encryption algorithm takes as input the public parameters PK, a message M, an access structure γ over the universe of attributes, and the current time slot number T. The algorithm will encrypt

[1] T^{att}_{begin} and T^{att}_{end} are two time slot numbers that represent respectively the first and the last time slot of the validity period of the attribute att.

M and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure during the T^{th} time slot will be able to decrypt the message. We will assume that the ciphertext implicitly contains γ and T.

- **Decrypt***(PK, CT, SK_T)*. The decryption algorithm takes as input the public parameters PK, a ciphertext CT, which contains an access policy γ and encryption time slot number T, and a private key SK_T, which is a private key for a set S_T of valid attributes at the T^{th} time slot. If the set S_T of attributes satisfies the access structure γ then the algorithm will be able to decrypt the ciphertext and return a message M.

C. Our model

Let A denotes the set of all attributes used by the Attribute Authority in the system, and T represents the set of all time slots numbers (we have $T \subset \mathbb{N}$). $\mathbb{G}_0$ is a bilinear group of prime order p. The one-way hash function H used in our scheme is defined as follows:

$$\begin{array}{rccc} H: & \mathbb{A} \times T & \longrightarrow & \mathbb{G}_0 \\ & (att, i) & \longmapsto & H(att, i) \end{array}$$

The hash function we use in our approach takes two parameters. The first parameter is an element from the set of all attributes maintained by the Attribute Authority, the second one is an integer that represents a time slot number.

We suppose that the probability of collision existence in the one-way hash function defined above is infinitely small. We mean by collision the existence of two different couples $(att_i, k), (att_j, l) \in \mathbb{A} \times \mathbb{N}$ (with: $(att_i, k) \neq (att_j, l)$), such that $H(att_i, k) = H(att_j, l)$. This assumption is described in the following formula:

$$\forall att_i, att_j \in \mathbb{A}, \forall k, l \in T : (att_i, k) \neq (att_j, l) \Rightarrow P(H(att_i, k) = H(att_j, l)) \approx 0 \quad (1)$$

D. Assumptions

1) Synchronization: We assume that the system is running a synchronization protocol that takes care to ensure synchronization between all entities in the system.

E. CP-ABE with attribute revocation

Let $\mathbb{G}_0$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_0$. In addition, let e : denote the bilinear map.

Setup. It chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g. Next it will choose two random exponents $\alpha, \beta \in Z_p$. The public key is published as:

$$PK = \left(\mathbb{G}_0, g, h = g^{\beta}, f = g^{1/\beta}, e(g, g)^{\alpha}\right) \quad (2)$$

and the master key is:

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

Note that f is used only for delegation, so we can omit it here as we do not talk about a delegation primitive. For more information we invite the reader to see [2].

KeyGen*(MK, S)*.

This KeyGen primitive takes as input the master key MK and a set S which contains a set of attributes and all the corresponding time slot numbers of their validity period. We can write the set S as:

$$S = \left\{\left(a, T^a_{begin}, T^a_{end}\right), \text{for all attribute } a\right\} \quad (4)$$

The Key generation algorithm chooses a random $r \in Z_p$, and then random $r_j \in Z_p$ for each attribute $j \in A$. Then it computes the key as

$$SK = \Big(D = g^{(\alpha+r)/\beta}, \forall j \in A, \forall k \in [\![T^j_{begin}, T^j_{end}]\!] : D_{j,k} = g^r H(j,k)^{r_j}, D'_j = g^{r_j}\Big) \quad (5)$$

Note here that the parameter $D_{j,k}$ is related to the attribute j for the time slot k.

Encrypt*(PK, M, γ, T)*.

The encryption primitive encrypts a message M under the tree access γ and the time slot T. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the access tree A. 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 ciphertext is then constructed by giving the tree access structure γ, the decryption time slot T and computing:

$$CT = \Big(\gamma, T, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y), T)^{q_y(0)}\Big) \quad (6)$$

Only users that satisfy the access tree γ during the time slot T can decrypt the ciphertext CT. We notice here that a user encrypting a message during the time slot T_1 can specify a different time slot T_2 for decryption.

Decrypt*(CT, SK_T)*.

The decryption primitive takes the ciphertext CT and a secret key SK_T, it is quite similar to the first form proposed in [2] with the unique difference in using our one-way hash function H defined in subsection V-C instead of a standard one.

We first define a recursive function $DecryptNode\left(CT, SK_T, x\right)$ that takes as input a ciphertext $CT = \left(A, T, \tilde{C}, C, \forall y \in Y : C_y, C'_y\right)$, a secret key $SK_T = \left(D, \forall j \in S_T : D_{j,T}, D'_j\right)$, which is associated with a set S_T of valid attributes at the time slot T, and a node x from the access tree A.

If the node x is a leaf node then we let $i = att\left(x\right)$ and define $DecryptNode$ as follows: If $i \in S_T$, then

$$\begin{aligned} DecryptNode\left(CT, SK_T, x\right) &= \frac{e\left(D_{i,T}, C_x\right)}{e\left(D'_i, C'x\right)} \\ &= \frac{e\left(g^r \cdot H\left(i,T\right)^{r_i}, g^{q_x(0)}\right)}{e\left(g^{r_i}, H\left(i,T\right)^{q_x(0)}\right)} \\ &= e\left(g,g\right)^{rq_x(0)}. \end{aligned}$$

If $i \notin S_T$, then $DecryptNode\left(CT, SK_T, x\right) = \perp$.

Now, we consider the recursive case when x is a non-leaf node. The algorithm $DecryptNode\left(CT, SK_T, x\right)$ then proceeds as follows: For all nodes z that are children of x, it calls $DecryptNode\left(CT, SK_T, z\right)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, we compute

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}; \text{ Where: } i = index\left(z\right), S'_x = \{index\left(z\right) : z \in S_x\} \\ &= \prod_{z \in S_x} \left(e\left(g,g\right)^{r \cdot q_z(0)}\right)^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} \left(e\left(g,g\right)^{r \cdot q_{parent(z)}(index(z))}\right)^{\Delta_{i,S'_x}(0)} \text{ (by construction)} \\ &= \prod_{z \in S_x} e\left(g,g\right)^{r \cdot q_x(i) \cdot \Delta_{i,S'_x}(0)} \\ &= e\left(g,g\right)^{r \cdot q_x(0)} \text{ (Using polynomial interpolation)} \end{aligned}$$

and return the result.

After defining our function DecryptNode, we can now define the decryption algorithm. The algorithm begins by calling the function on the root node R of the tree A. If the tree is satisfied by S_T we set $A = DecryptNode\left(CT, SK_T, r\right) = e\left(g,g\right)^{rq_R(0)} = e\left(g,g\right)^{rs}$. The algorithm now decrypts by computing

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

VI. Performance Evaluation

In this section, we present a series of simulations that we carried out in order to study the effect of the time slot duration on system performances. We consider two system performance parameters: (i) the Average Number of Time slots in an attribute validity period (ANT). This is because the number of time slots in a validity period determines the number of associated secret key parts to be sent, (ii) the Average Waiting Time that refers to the average delay between access request to an attribute and the beginning of the true validity period for that attribute.

A. Simulation Model

For the sake of simplicity, we considered a group of users which ask gaining the access right to only one attribute. We modeled users' requests, which represent starting dates of attribute validity periods, by Poisson process with the parameter λ. The attribute validity periods durations for all users follow exponential distribution with parameter μ.

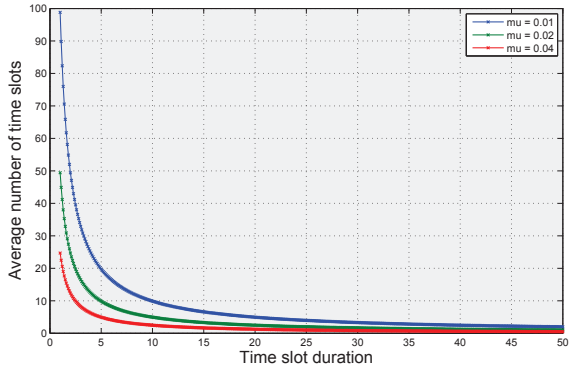

Figure 2: Average number of time slots

B. Average Number of time slots (ANT)

The figure 2 shows the impact of the time slot duration on the system performance i.e. the average number of time slots per user. The figure shows also the impact of the parameter μ on the average number time slots per user. From this figure and figure 3, we can deduce empirically that the average number of time slots ANT is inversely proportional to the chosen time slot duration Δt:

$$ANT \approx \frac{1}{\mu \Delta t} \tag{7}$$

C. Average Waiting Time (AWT)

We are also interested on the average waiting time AWT, figure 4 shows the result of a experiment simulating the variation of the overall average waiting time of all users depending on the the time slot duration. We set $\lambda = 0.01$. We note that the average waiting time AWT is independent of the Poisson process parameter λ, on the other side it increases linearly with the time slot duration Δt, we can deduce empirically:

$$AWT \approx 1/2 \cdot \Delta t \tag{8}$$

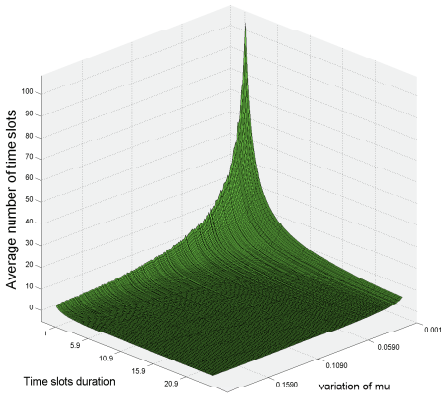

Figure 3: Average number of time slots

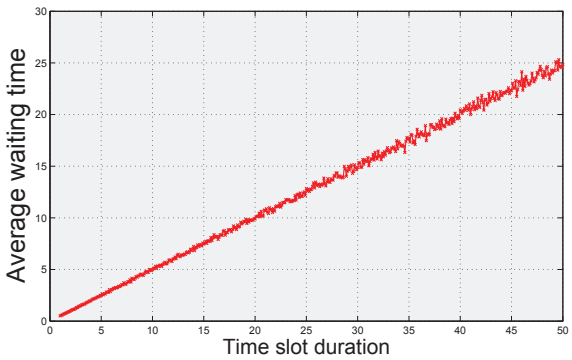

Figure 4: Average waiting time

D. ANT vs AWT

We notice here that we are trying to minimize both of the Average Waiting Time AWT and the Average Number of Time slots per user ANT which are two conflicting objectives i.e. minimizing AWT leads to maximizing ANT and vice versa. In real life applications we choose a tradeoff between performance and delay.

From formulas 8 and 7, we can deduce that:

$$ANT \approx \frac{1}{2\mu AWT} \tag{9}$$

and this is shown in figure 5.

In real applications we define some conditions and constraints related to the system requirements such as:

$$\begin{cases} AWT < \alpha/\mu \\ ANT < \beta/\mu \end{cases}$$

α and β are to be determined according to the application requirements. In figure 5, we show in red color the set of solutions that satisfy these conditions where $\alpha = 1/10$ and $\beta = 1/5$.

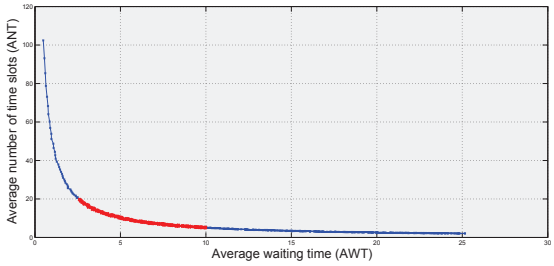

Figure 5: AWT vs ANT

VII. Conclusion

In this paper we have proposed a batch-based version for Ciphertext Policy Attribute-Based Encryption to achieve attributes revocation in an Internet of Things environment. Our solution reduces the overhead and the complexity comparing to previous solutions, and it does not require to re-encrypt data every attribute policy change. We carried out some simulations to show the effect of the variation of the time slot duration on the system performances.

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] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2010.05.010

[2] 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.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 89–98.

[4] 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*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 99–112. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180419

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

[7] Y. Cheng, Z.-y. Wang, J. Ma, J.-j. Wu, S.-z. Mei, and J.-c. Ren, "Efficient revocation in ciphertext-policy attribute-based encryption based cryptographic cloud storage," *Journal of Zhejiang University SCIENCE C*, vol. 14, no. 2, pp. 85–97, 2012. [Online]. Available: http://dx.doi.org/10.1631/jzus.C1200240

[8] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '10. New York, NY, USA: ACM, 2010, pp. 261–270. [Online]. Available: http://doi.acm.org/10.1145/1755688.1755720