Threshold Yoking/Grouping Proofs Based on CP-ABE for IoT Applications

Lyes Touati, Hamed Hellaoui, Yacine Challal

Threshold Yoking/Grouping Proofs based on CP-ABE for IoT Applications

Lyes Touati*, Hamed Hellaoui†, Yacine Challal†‡

*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
†Ecole nationale Supérieure d'Informatique ESI, LMCS Laboratory
16309 Oued Smar, El Harrach, Algiers, Algeria. Email: {h_hellaoui, y_challal}@esi.dz
‡Centre de Recherche sur l'Information Scientifique et Technique CERIST
05 Rue des Frères Aissou, Ben Aknoun, Algiers, Algeria. Email: ychallal@cerist.dz

***Abstract*—The Internet of Things (IoT) introduces a new vision in which objects are connected to the network. This paradigm is receiving much attention of the scientific community and it is applied in many fields. In some cases, it is useful to prove that a number of objects are simultaneously present in a group. For instance, a client might want to authorize NFC payment with his mobile only if k of his devices are present to ensure that he is the right person. This principle is known as yoking/grouping Proofs. However, existing grouping schemes are mostly designed for RFID systems and don't fulfill the IoT characteristics. In this paper, we tackle this issue and propose a threshold yoking/grouping proof for IoT applications. Our scheme uses the Ciphertext-Policy Attribut-Based Encryption (CP-ABE) protocol to encrypt a message so that it can be decrypted only if at least k nodes are simultaneously present. A security analysis and performance evaluation is conducted to show the effectivenesses of our proposal solution.**

***Index Terms*—IoT; Grouping proofs; CP-ABE; Pairing Cryptography;**

I. Introduction

The Internet of Things (IoT) emerges as a new paradigm in which physical objects (such as sensors, actuators, RFID Tags) are able to communicate with each other and interact over the network to reach common goals. This concept will allow the development of a smart world where every-day objects are interconnected. As the IoT creates unprecedented opportunities, it also raises new types of risks. Security issues are considered as one of the most aspects that attract attention of the IoT research community. Since objects can be limited in terms of energy, storage and computation, security solutions have to be adapted to cope with these challenging factors [1].

Yoking-proof is a security concept which was introduced by Juels in 2004 [2]. It aims to provide a proof that a pair of RFID tags has been scanned simultaneously. Although it originally addresses RFID systems, yoking-proof can be considered for IoT applications. For example, the manager of an emergency center might want to ensure that the ambulance doesn't leave the garage without a mobile scanner. In its basic description, yoking-proof addresses only a pair of devices. Given the necessity of considering this concept for a group of more than two nodes, the notion of grouping-proof was proposed as a generalization of the yoking-proof concept [3][4]. However, existing yoking/grouping schemes are mostly designed for FRID systems. Characteristics of IoT applications introduce other requirements that have to be considered by yoking/grouping schemes.

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [5] is a well-known security protocol which has been widely used to develop secure systems. It provides a public key encryption scheme which enables a fine-grained access control, a scalability management, and a flexible data distribution. However, CP-ABE is commonly employed in cryptographic access control solutions and not to check the presence of a group of devices. In this paper, we propose a threshold yoking/grouping system based on CP-ABE for IoT applications. Main contributions of our work are the following:

- We propose a new yoking/grouping scheme for IoT applications, based on a powerful security protocol (CP-ABE).
- We propose the notion of threshold in grouping schemes that consists in proving the simultaneous presence of at least k entities.
- We also introduce the concept of entity importance to give to the presence of a particular object more importance in the grouping proof.

The rest of this paper is organized as follows. Section II gives a review on related works about yoking/grouping proofs. A background of the CP-ABE protocol is presented in section III. Our solution to develop a threshold yoking/grouping proof scheme for the IoT using CP-ABE is explained is section IV. An analysis of the proposed approach is conducted in section V. Section VI presents applications that illustrate use case scenarios. Section VII concludes this paper and draws future work.

II. Related Works

The concept of yoking-proofs was proposed by Juels as means to prove that a pair of RFID tags has been present simultaneously [2]. The proposed technique uses the reading device to generate a proof that can be off-line verifiable by a third-party (verifier).

Several proposals and critics have been made on the yoking-proof protocol. Bolotnyy and Robine raised a security concern in [4]. Indeed, Jules's protocol reveals tags identifier which creates a privacy problem when the reader is untrusted. The authors introduced a new problem formulation called anonymous-yoking which requires preserving privacy in the yoking-proof protocol.

In [3] authors pointed out that Jules's protocol is vulnerable to replay attack and a yoking-proof can be obtained with one RFID tag. The authors proposed a version that introduces a timestamp generated by a database to prevent from replay attack. The authors proposed also a generalization of the yoking-proof protocol to a group of tags. In their scheme, the reader sends the timestamp to all tags participating to the protocol, which is used to parameterize the proof. However, because timestamp increases sequentially, an untrusted reader can take a future timestamp value, use it on one tag and wait that the database sends this value to exploit the old result generated by the tag with the current results of the rest of the group, which breaks the simultaneity character of the yoking proof. Although Piramuthu proposed a solution based on the use of a random number instead of a timestamp [6], an attacker can attempt a brute force attack method, especially if the range of the random number is not large (i.e. the attacker uses all possible numbers and stores results). This issue was addressed by Cho et al. in [7], where the authors proposed a solution based on dividing the random number into two numbers to raise complexity of the brute force attack in terms of storage space. But this will work well as long as attackers have limited resources. Another technique can be found in [8], where authors use so random number as RFID tags. However, the proposed solution reveals tags identifiers and creates a privacy problem.

In the same context, Bolotnyy and Robine proposed in [4] a solution based on the construction of a circular chain while polling tags. The purpose behind the use of a chain is to ensure that an untrusted reader will not be able to mount a replay attack or generate a proof if it breaks the chain. However, when dealing with a large set of tags, the completion of the proof would take a time and an adversary can take a tag once it is interrogated, breaking therefore the simultaneity character. This issue was tackled by Fuentes et al. in [9]. The authors proposed an approach based on dividing the set of devices (the authors consider IoT devices) into several subsets with low cardinality and poll each subset in unpredictable manner. The scheme operates in a number of rounds in such a way that different subsets are rebuilt in each round, which reduces the chance that an adversary takes a device without corrupting the proof. However, this kind of solution would increase the execution time of the proof.

Furthermore, in all the aforementioned propositions, the size of the generated proof increases proportionally with the size of entities. This could create a storage problem for the verifier as it receives the proof. The same can be said about the stored keys (each entity shares its secret key with the verifier so the latter can verify the proof). In this paper, we propose a new yoking/grouping proof system based on a powerful security protocol, the Ciphertext Policy Attribute-Based Encryption (CP-ABE). In fact, CP-ABE has been proved secure in [5]. Especially, data encrypted by CP-ABE can't be accessed by an unauthorized user. We use this protocol to encrypt a message in such a way that it can be decrypted only if at least k nodes are simultaneously present. For the best of our knowledge, this is the first work that tackle the grouping proof from this perspective.

III. Background

In this section we recall some preliminaries and notions used in the paper. Section III-A gives the definition of bilinear maps. Section III-B presents CP-ABE scheme in highlights. Finally, Section III-C introduces the concept of access structure used in CP-ABE scheme.

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. Ciphertext-Policy Attribute-Based Encryption

Ciphertext-policy Attribute-Based Encryption [5] is an encryption mechanism that allows to implement a cryptographic fine-grained access control. users' private keys are associated with a set of attributes, on the other side, data are encrypted over an access tree defining the access policy to the ciphertext. Only users' whose attributes sets satisfy the access policy can decrypt the ciphertext.

It consists mainly of four primitives:

- **Setup**. 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 takes the master key MK and list of attributes S. The 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 encrypts M and produces 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 cipher-text and outputs the recovered message M.

For more details about the primitives construction, we invite the reader to see [5].

C. Access Trees

Access trees [10] [5] are used in CP-ABE scheme to describe access policies for ciphertexts. Therefore, only users with attributes sets satisfying the access tree of a ciphertext are able to decrypt it. Figure 1 shows an example of access trees.

Each interior node of the access tree is a threshold gate (k out-of-N gate), and leaf nodes are associated with attributes. Figure 1 is an example of an access tree.

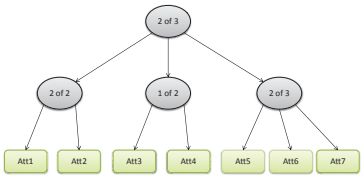

Figure 1: Example of Access Tree

The user's attributes set must satisfy the access policy, so as the user is allowed to decrypt the ciphertext. For example, a user with an attribute set $S1 = \{att1, att5\}$, is not able to decrypt a ciphertext with the access policy of the Figure 1. However, a user with $S2 = \{att1, att2\}$, $S3 = \{att3\}$, or $S4 = \{att5, att7\}$ has the ability to recover the original message from the ciphertext.

We define some functions 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 subtree 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 leaf node, return 1 if $att(x) \in \gamma$, 0 otherwise.
- If x is a non leaf node, we evaluate $T_{x'}(\gamma)$ for each child x' of x and compute the sum cpt of all returned values. Return 1 if $cpt \geq k_x$, 0 otherwise.

In this paper, we use only access trees of only two levels: level zero containing the root node and level one composed by the leaf nodes. The root node is a threshold gate with value $k = k_r$ representing the threshold. The number of the leaf nodes is the same as the number of entities in the group $N = num_r$.

IV. Our Approach

This section explains in detail the proposed threshold grouping proof protocol. First, Section IV-A describes the main concept of the scheme. Afterwards, Section IV-B presents the different entities involved in the system. Section IV-C presents the security requirements that a threshold grouping proof protocol must meet. Section IV-D describes in detail the construction of our protocol. Section IV-F proposes a solution for managing several groups of entities with our protocol. The notation in use throughout this paper is shown in Table I.

A. Overview

The main idea of our solution is to use Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [5] mechanism to implement our threshold grouping proof scheme in such a way to encrypt a secret so the latter can be decrypted only if at least k entities are simultaneously present. Fist, the Attribute Authority generates a secret key SK associated with the list of attributes $S = \{attribute\_1, attribute\_2, \cdots, attribute\_N\}$, where N is the number of entities in the group. The couple of elements (D_j, D'_j) associated to the attribute j is sent to the entity j, where $1 \leq j \leq N$. The element D of the secret key is sent to a proxy who is intended to recover the secret.

A Secret message is encrypted with the following policy: k of N $(attribute\_1, attribute\_2, \cdots, attribute\_N)$. Where k is the threshold $(1 \leq k \leq N)$. Our solution allows the use of many secrets, each one of them with a different threshold. This enables making different proofs with different thresholds.

Table I: Summary of notations.

Notation	Description
PK	Public Key generated by the Attribute Authority
SK	Secret Key generated by the Attribute Authority for each user from her/his attributes list
S	List of attributes used to construct the secret key
N_g	Number of groups
N	Number of entities in the group
CT	The secret intended to be recovered by the group of entities, it is also the result of encryption
γ	Access tree used to encrypt messages. $\gamma = k - of - N(Attribute\_1, Attribute\_2, \cdots, Attribute\_N)$
M	The message to encrypt
k	The threshold of secret recovery
E_j	The j^{th} entity in the group of entities

B. Network Model

- **Entities:** we consider a group of N entities with a predefined threshold k.
- **Proxy:** it is responsible of relating the group of entities in order to recover a secret.
- **Attributes Authority:** it is a special entity in the system. The Attribute Authority is responsible for configuring the system by creating Public and Master keys which are used after that for encrypting messages and creating secret keys.
- **Verifier:** which is responsible for generating the secret and verifying the proof.

C. Security requirements

- Collusion of any k or more entities of the group makes the secret message M easily computable.
- Collusion of any $k-1$ or fewer entities of the group leaves the secret message M completely undetermined.
- Group entities must disclose no information about their secret elements during the secret recovery process.
- The presence of entities must be in the same time (within a short interval).
- An adversary or an untrusted reader must be prevented from knowing if an object with a given identity is present during the proof.

D. Construction

For the sake of simplicity we will consider a single group of entities. It is easy to generalize our scheme to many groups (A simple way to do that is explained in Section IV-F). Let suppose we want to construct the proof for a group of N entities. We can split our protocol into three main phases: *System configuration* phase, *Sharing secret* phase and *Recovering secret* phase.

System configuration

The Attribute Authority starts by running the *Setup* primitive of CP-ABE to generate a public key PK and a master key MK. The Attribute Authority chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g. Next it will choose two random exponents $\alpha, \beta \in \mathbb{Z}_p$. The public key is published as:

$$PK = (\mathbb{G}_0, g, h = g^{\beta}, e(g,g)^{\alpha}) \tag{1}$$

and the master key MK is (β, g^{α}).

This operation is executed once at the beginning of the system configuration.

Given a group of N entities $GE = \{E_1, E_2, \cdots, E_N\}$, the Attribute Authority executes the *KeyGen* primitive to generate a secret key SK associated with the attribute set $S = \{attribute\_1, attribute\_2, \cdots, attribute\_N\}$. We suppose here that each attribute $attribute\_i \in S$ is hold by the entity E_i.

It first chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$. Then it computes the key as

$$SK = \Big(D = g^{(\alpha+r)/\beta}, \forall j \in S : D_j = g^{r} \cdot H(j)^{r_j}, D'_j = g^{r_j}\Big) \tag{2}$$

The element D of the key SK is given to the proxy as it is responsible for doing the secret recovery process. Each couple (D_j, D'_j) is given to the corresponding entity E_j of the group holding the $attribute\_j$. Figure 2 illustrates the system configuration phase.

Sharing secret

To generate the grouping proof for a group of N entities, the group identifier has to be sent to the verifier. The latter encrypts a secret M with a particular access policy γ (k of N ($attribute\_1, attribute\_2, \cdots, attribute\_N$)). The access tree associated with γ has a node root with a threshold value $k_R = k$. The root node R has N children, all of them are leaves. Each leaf node is associated with one attribute $attribute\_i$. S is the set of leaf nodes.

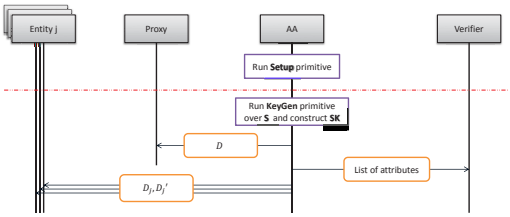

Figure 2: System configuration phase

The Encryption process begins with choosing a polynomial q_x for each node x in the tree γ. These polynomials are chosen in the following way in a top-down manner, starting from the root node R.

The degree of the polynomial corresponding to the root node is $d_R = k_R - 1 = k - 1$. All other polynomials associated to leaf nodes are with a degree equal to $d_x = k_x - 1 = 0$.

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 each leaf node x, the algorithm sets $q_x(0) = q_{parent(x)}(index(x)) = q_R(index(x))$.

The secret message is constructed as follows:

$$CT = \Big(\gamma, \tilde{C} = Me(g,g)^{\alpha s}, C = h^{s}, \forall y \in S : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\Big) \tag{3}$$

Once it is done, the verifier sends the secret message to the proxy and arms a timer. If the proxy could not recover the secret before the release of the timer, the verifier consider that the proof can not be constructed and the process stops.

Recovering secret

To recover a secret message M shared with the group of entities, the proxy starts by requesting assistance from the group entities. If k or more entities respond, the proxy can proceed with the recovery process, otherwise the secret recovery stops (Figure 3).

The proxy sends to each entity E_j in the group the couple (C_j, C'_j) of the ciphertext CT. Entities wishing to participate in the secret recovery process use their received values from the Attribute Authority to compute:

$$\begin{aligned} F_j &= \frac{e(D_j, C_j)}{e(D'_j, C'_j)} \\ &= \frac{e(g^{r} \cdot H(i)^{r_j}, h^{q_j(0)})}{e(g^{r_j}, H(j)^{q_j(0)})} \\ &= e(g,g)^{r q_j(0)} \end{aligned} \tag{4}$$

We suppose that the proxy receives at least k intermediate results $(e(g,g)^{rq_j(0)}\ \ (1 \le j \le N))$. Let Q be the set of attributes corresponding to participating entities. P is a subset of Q that has a cardinal number equals to k. We have $P \subseteq Q \subseteq S$ and $|Q| \ge |P| = k$. The proxy proceeds with the decryption process by computing:

$$\begin{aligned} A &= \prod_{z \in P} (e(g,g)^{r \cdot q_z(0)})^{\Delta_{i,P'}(0)} \\ &= \prod_{z \in P} (e(g,g)^{r \cdot q_{parent(z)}(index(z))})^{\Delta_{i,P'}(0)} \\ &= \prod_{z \in P} (e(g,g)^{r \cdot q_R(i) \cdot \Delta_{i,P'}(0)}) \\ &= e(g,g)^{r \cdot \sum_{z \in P} q_R(i) \cdot \Delta_{i,P'}(0)} \\ &= e(g,g)^{r \cdot q_R(0)} \\ &= e(g,g)^{rs} \end{aligned} \tag{5}$$

Where $P' = \{index(z) : z \in P\}$, $i = index(z)$, and $\Delta_{i,P'}(0)$ is the Lagrange Coefficient [1].

The algorithm now recovers the secret by computing

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

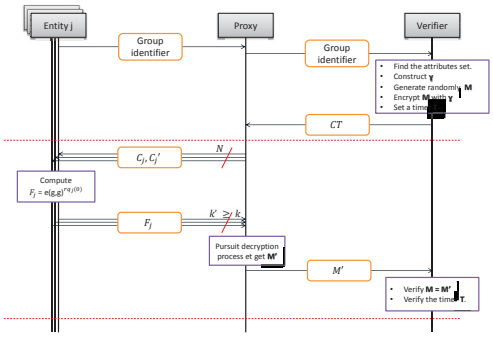

Figure 3: Sharing and recovering secret phases

Once recovered, the proxy sends back the secret to the verifier which issued the proof.

E. Group dynamics

Our solution allows easily and efficiently adding and removing entities to and from the group.

Adding entities to the group. In order to add an entity E_{N+1} to the group of entities, the Attribute Authority must generate parts of the secret key D_{N+1}, D'_{N+1} related to the attribute $attribute\_(N+1)$ and send them to the new entity. To be able to do that, the Attribute Authority must have saved the random element r related to the CP-ABE secret key's group, thus, it has just to pick a random number r_{N+1} from $\mathbb{Z}_p$ and compute the couple (D_{N+1}, D'_{N+1}) this way:

$$\begin{cases} D_{N+1} = g^r \cdot H\left(attribute\_(N+1)\right)^{r_{N+1}} \\ D'_{N+1} = g^{r_{N+1}} \end{cases} \tag{7}$$

Once an entity is added to the group, the Verifier will add the corresponding attribute to the access tree, so as, the new entity could participate to the decryption process.

The access tree will henceforth be like:

$$\begin{aligned} \gamma = k - of - (N+1)\Big(&attribute\_1, attribute\_2, \cdots, \\ &attribute\_N, attribute\_(N+1)\Big) \end{aligned} \tag{8}$$

Removing entities from the group. In order to remove an entity E_e from the group of entities it is just enough to inform the Verifier that the entity is no longer part of the group, and he will remove its corresponding attribute $attribute\_e$ from the access tree. The new access tree will be:

$$\begin{aligned} \gamma = k - of - (N-1)\Big(&attribute\_1, \cdots, attribute\_(e-1), \\ &attribute\_(e+1), \cdots, attribute\_N\Big) \end{aligned} \tag{9}$$

The proxy should be aware too as it is responsible of interrogating the entities during the decryption process.

As the attribute corresponding to the removed entity does not appear in the access tree, the entity could not participate to the decryption process even if it still has a part of the secret key.

F. Managing many groups of entities

The proposed scheme works well for one group of entities. Managing several groups of entities with the same system configuration requires to define a strategy of naming the attributes used in generating shared secret keys. A simple way to do that, is to associate an identifier j for each group of entities. The set of attributes used for the j^{th} group of entities will be $\{attribute\_ < j > \_1, attribute\_ < j > \_2, \cdots, attribute\_ < j > \_N_j\}$. Where N_j is the number of entities in the j^{th} group.

V. Analysis

This section presents security and performance analysis of our scheme. First, Section V-A explains how our scheme meets the security requirements presented in Section IV-C. We highlight some advantages of our solution in Section V-B. Then, Section V-C studies the performance analysis of our solution and presents how to overcome its shortcomings.

A. Security Analysis

Once a secret message M is encrypted with the policy $\gamma = k - out - of - N(attribute\_1, attribute\_2, \cdots, attribute\_N)$ it is not possible to any third party to recover its value (None has a secret key with an attribute set satisfying γ).

As the threshold defined for this ciphertext is k out of N attributes $(attribute\_1, attribute\_2, \cdots, attribute\_N)$, the

[1] $\Delta_{i,S}(x) = \prod_{j \in S, j \ne i} (x-j)/(i-j)$.

collusion of any $k-1$ or less of entities belonging to the targeted group is pointless. Indeed, CP-ABE scheme prevents decrypting a ciphertext when the access policy is not satisfied by the attributes set of the secret key used. This property of our scheme meets the second security requirement (section IV-C). However, The collusion of at least k entities will end up by decrypting the ciphertext as it is detailed in section IV-D.

An entity participating in the secret recovery does not reveal any information about the secret elements that she received from the Attributes Authority. Indeed, she computes $e(g,g)^{rq_j(0)}$, and sends it back to the proxy for recovering the secret. It is obvious that based on this information, neither the proxy nor a third party can get any information about $D_j = g^r \cdot H(j)^{r_j}$ or $D'_j = g^{r_j}$.

During the grouping process, objects don't use their identifiers to generate the proof. Even knowing object's attribute the proxy cannot know its identifier since attributes have no direct link with objects to whom they are associated. Thus, an adversary or an untrusted reader is unable to know if a given object is present during the proof.

The verifier uses a timer to ensure the presence of entities in the same time. If the secret could not be decrypted within a short time, the proof will not be constructed.

To attempt a replay attack with an untrusted proxy, the latter has to return the right secret message to the verifier. However, since the secret is encrypted with CP-ABE, the proxy cannot have access. Even brute force attack is difficult to consider because the proxy does not know what to return to the verifier.

B. Advantages

- **Variability of the threshold k:** The value of the threshold k is determined at the time of encryption. Which means that, for each secret we can define a different threshold with the same system configuration.
- **Variability of users' importance:** Our scheme has the ability to grant to some entities more importance than the others. In other words, the participation of an entity E_i during the secret recovery process could be equivalent to the participation of several other simple entities. This could be achieved by according so many attributes as the importance the entities.
- **Possibility of easily adding and removing entities to/from the group:** Once the system configuration is done, it is still possible to add/remove easily entities to/from the group (See Section IV-E).
- **One Setup Many Uses:** Once the system is setup, many proofs can be obtained from the configuration. Indeed, we can decrypt as much as we want with the secret key shared between different objects of the group without reconfiguring the system.

We present in Table II a comparison between our proposal approach with previously presented related works. The solution we propose preserves privacy and resists against replay and brute force attacks. Contrary to chain-based techniques, the proxy can interrogate entities without waiting a response from another entity. Furthermore, our approach introduces original proprieties, which are the variability of the threshold k and the importance of users' entities.

Table II: Comparison between approaches

	Privacy	*Replay attack*	*Brute force attack*	*Time with scalability*	*Variability of the threshold k*	*Variability of entity importance*
[2]	-	-	-	/	-	-
[3]	-	+	-	+	-	-
[6]	+	+	-	+	-	-
[7]	+	+	-	+	-	-
[8]	-	+	+	+	-	-
[4]	+	+	+	-	-	-
[9]	+	+	+	-	-	-
Ours	+	+	+	+	+	+

C. Performance Analysis

Performing our proposal scheme requires encrypting a random message (by the verifier) and recovering the secret (by the entities and the proxy). Number of operations that costs each node during each phase is summarized in Table III.

Table III: Performance Analysis: Number of operations

		Nb. Pairing	**Nb. Exp.**	**Nb. Mul.**	**Nb. Div.**
System Configuration Phase	AA	1	$(3)+(3N+1)$	1	1
Sharing Secret Phase	Encryptor	-	$2(N+1)$	1	-
Secret Recovery Phase	Each Entity	2	-	-	1
	Proxy	1	k	$k-1$	2
CP-ABE decryption	Decryptor	$2N+1$	k	$k-1$	$N+2$

Encryption

CP-ABE is known to be very complex and expensive in terms of computation and energy consumption, especially the encryption primitive. Indeed, there are $2*N+2$ exponentiation and one (1) multiplication to do during the encryption process. However, in our proposal scheme this operation is performed by the verifier which commonly has considerable resources. When this primitive is executed by a resource-constrained device, this could drain quickly it battery and therefore reduce considerably its lifetime. However, In [11] a solution has been proposed to reduce this computation overhead and the overall energy consumption. It leverages the heterogeneity of the IoT environment to delegate costly operations namely exponetiations to more powerful trusted devices in the neighborhood.

Other solutions were proposed in [12] [13]. These solutions consist on splitting the encryption primitive into two steps. The first one which does the most of heavy operations is executed when the device is online (using harvested energy or line power). The second step is done offline when the device is not connected to an energy source (Offline), it continue executing the rest of the encryption primitive.

Secret Recovery

Our scheme allows to split the overhead of CP-ABE decryption primitive between the group of entities and the proxy.

Therefore, each entity computes only two pairings and one division, rather than $2N+1$ pairing and $N+2$ divisions. The exponentiation and the multiplication operations are performed by the proxy which has commonly significant resources. This particularity of our schema is very interesting when it is implemented in resource-constrained devices.

We have made a comparison between our approach and the original CP-ABE in term of decryption primitive performances. we considered an application case where the number of entities in the group N = 20, the threshold for decryption k = 10. The numbers of operations executed by an entity in both original CP-ABE and our approach are illustrated in Figure 4.

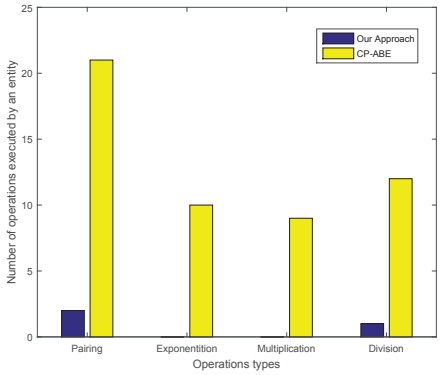

Figure 4: Number of operations executed during the decryption process by each entity

We notice that our approach reduces considerably the number of operations to execute during the decryption process. Indeed, it squarely remove exponentiations and multiplications as they are executed alongside the proxy. It also reduces the number of pairing operations and divisions to two (2) and one (1) respectively.

Secret Key Storage

Sizes of all elements used PBC library [14] are given in Table IV. We recall that these sizes are specific to the pairing parameters in the file *"f.param"*.

Table IV: Elements Sizes

Group	Size (bytes)
$\mathbb{G}_1$	44
$\mathbb{G}_2$	84
$\mathbb{G}_T$	244
$\mathbb{Z}_p$	24

Table V shows the size of secret elements sizes of both entity and proxy for our scheme, and for original CP-ABE. N_g notices the number of considered group and N_i denotes the number of entities in the i^{th} group ($1 \leq i \leq N_g$). We notice that an entity in our scheme requires a fixed storage size per group, regardless of the number of entities. Indeed, each entity stores only two elements from $\mathbb{G}_1$ and $\mathbb{G}_2$. The proxy has to store only one element of $\mathbb{G}_T$ (244 bytes) for every group of entities.

Table V: Performance Analysis. Required Storage Capacity

		Size of secret elements (bytes)
CP-ABE	Entity	$128 * N_i + 244$
Our Approach	Each Entity	128
	Proxy	$244 * N_g$

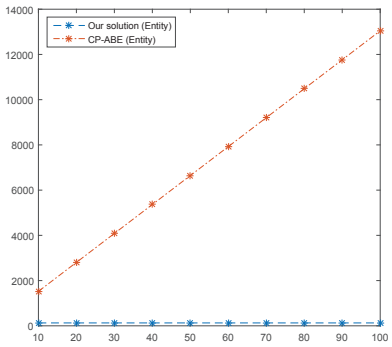

Figure 5: Size of secret elements for both the proxy and an entity in both of our approach and original CP-ABE

Figure 5 shows the variation of the size of an entity's and proxy's secret elements in both of our approach and original CP-ABE with respect to the number of attributes in secret key (The number of entities in the group). In our approach, the required storage size for an entity is independent from the number of entities. Furthermore, unlike existing grouping-proof schemes, entities don't need to share their keys with the verifier. The latter uses group's attributes to generates the secret message which are so lightweight compared to keys. Even the generated proof has a fixed size ($|\mathbb{G}_T|$ = 244 bytes) regardless of the number of entities or groups. This is a very important characteristic when it comes to large scale groups.

Adding and Removing Entities

In original CP-ABE, adding and removing attributes to/from a secret key requires appealing to an attribute revocation mechanism [5]. The latter is known to be difficult to develop. Trivial solution [15] of this problem is to rename the attributes by adding an expiration date for each attribute. Many others solutions have been proposed in the literature [16], [17], [18], but they are not efficient and some of them are even not applicable at all as they do not meet requirements of target application. Indeed, some of them require the re-encryption of data using the Proxy Re-Encryption Technique [19]. Others propose to re-generate all secret keys sharing the concerned attribute. These solutions induce high overheads in terms of computation and bandwidth.

In this paper, we took advantage of our application context to propose a simple key update mechanism which is specific to our Threshold Grouping Proof approach (See Section IV-F). As shown in the previous section, our solution is very simple and very efficient comparing to solutions in the literature.

VI. Application cases in the IoT

In the following, we provide two case scenarios of the use of the proposed scheme in IoT environment: control access to a location, and Mobile NFC payment.

- **Location access control:** we consider a security system that controls location access for authorized persons. In order to provide a high security level, the control system authorizes the access only if the user holds, simultaneously, k nodes of his authenticated devices (e.g. smart phone, smart bracelet, and smart glasses). In this way, even if one device is stolen, the thief will not be allowed to access the location. The system is composed of a proxy that is charged of interrogating user's devices and a verifier entity which receives the proof and decides to authorize the access or not. When the user asks for access, the request is directed to the verifier and the latter starts the proof (the simultaneous presence of at least k of user's devices). Therefore, the verifier encrypts a random message and sends it to the proxy. In its turn, the proxy requests the group of devices and sends to each one of them its related part of the ciphertext. If k or more nodes respond, the message can be decrypted. By sending the decrypted message to the verifier, the latter considers that the user is authorized and the access is allowed. Otherwise (timeout release), the user will not have access to the location.

- **Mobile NFC payment:** NFC payment allows users to issue payment transactions using their NFC-enabled mobiles. The use of mobile devices helped to enhance the security level and make transactions more secure than NFC cards (require certification for applications, use of authentication for users, etc.). However, if the phone gets stolen, payments can be issued by an unauthorized person. Indeed, some clients may deactivate protection control features because they are often centered on user's interaction (e.g. entering the PIN code to issue the transaction). Using our proposed scheme, the user will be authenticated through the simultaneous presence of his devices without requiring his interaction. Thus, when the user requests a NFC payment transaction, the mobile encrypts a random message and sends it to each user's device its related part of the ciphertext (in this example, the mobile is considered as proxy and verifier). the secret can be decrypted only if at least k devices respond before the release of timer. If so (secret decrypted), the mobile continue the transaction.

VII. Conclusion

In this paper we have presented a solution to implement a threshold yoking/grouping proof scheme using Ciphertext-Policy Attribute-Based Encryption. Our approach provides more resistance against attacks by encrypting a secret in a such way that it can be recovered only if at least k entities are present at the same time. In addition to standard yoking/grouping proof proprieties, our solution introduces other features, such as the variability of the threshold k and the importance of user's entities.

It would be also interesting to consider other access trees more complex than *k-out-of-N*: an access tree implying **AND** and **OR** gates. For example, the access tree γ: *President* **AND** k of N $(attribute\_1, \cdots, attribute\_N)$ imposes the participation of the *President* entity to the secret recovery process and any k of N other entities.

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," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[2] A. Juels, ""yoking-proofs" for rfid tags," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, March 2004, pp. 138–143.

[3] J. "Saito and K. Sakurai, "Grouping proof for rfid tags," in *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, March 2005, pp. 621 – 624.

[4] L. Bolotnyy and G. Robins, "Generalized "yoking-proofs" for a group of rfid tags," in *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*. IEEE, 4 2007.

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, May 2007, pp. 321–334.

[6] Piramuthu, "On existence proofs for multiple rfid tags," in *Pervasive Services. the 2006 ACS/IEEE International Conference on*, June 2006, pp. 317 – 320.

[7] J.-S. Cho, S.-S. Yeo, S. Hwang, S.-Y. Rhee, and S. K. Kim, "Enhanced yoking proof protocols for rfid tags and tag groups," in *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, March 2008, pp. 1591–1596.

[8] D. N. Duc and K. Kim, "Grouping-proof protocol for rfid tags: Security definition and scalable construction," 2009.

[9] J. M. de Fuentes, P. Peris-Lopez, J. E. Tapiador, and S. Pastrana, "Probabilistic yoking proofs for large scale iot systems," *Ad Hoc Networks*, vol. 32, pp. 43 – 52, 2015, internet of Things security and privacy: design methods and optimization.

[10] A. Beimel, *Secure schemes for secret sharing and key distribution*. Technion-Israel Institute of technology, Faculty of computer science, 1996.

[11] L. Touati, Y. Challal, and A. Bouabdallah, "C-cp-abe: Cooperative ciphertext policy attribute-based encryption for the internet of things," in *Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on*, June 2014, pp. 64–69.

[12] G. Bianchi, A. T. Capossele, C. Petrioli, and D. Spenza, "Agree: Exploiting energy harvesting to support data-centric access control in wsns," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2625–2636, Nov. 2013.

[13] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Public-Key Cryptography - PKC 2014*, ser. Lecture Notes in Computer Science, H. Krawczyk, Ed. Springer Berlin Heidelberg, 2014, vol. 8383, pp. 293–310.

[14] "The pairing-based cryptography library," https://crypto.stanford.edu/pbc/, accessed: 2016-04-26.

[15] 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.

[16] L. Touati and Y. Challal, "Batch-based cp-abe with attribute revocation mechanism for the internet of things," in *Computing, Networking and Communications (ICNC), 2015 International Conference on*, Feb 2015, pp. 1044–1049.

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

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

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