C-CP-ABE: Cooperative Ciphertext Policy Attribute-Based Encryption for the Internet of Things

Lyes Touati, Yacine Challal, Abdelmadjid Bouabdallah

C-CP-ABE: Cooperative Ciphertext Policy Attribute-Based Encryption for the Internet of Things

LYES TOUATI, YACINE CHALLAL AND ABDELMADJID BOUABDALLAH

Heudiasyc UMR CNRS 7253, Université de Technologie de Compiègne
Email:{lyes.touati,ychallal}@hds.utc.fr

***Abstract*—Ciphertext Policy Attribute-Based Encryption (CP-ABE) is an extremely powerful asymmetric encryption mechanism, but its complexity and its overhead cannot be neglected in an Internet of Things environment. Indeed, Internet of Things, by its heterogeneous nature, may contains highly resource-constrained devices that are not able to support the heavy overhead due to CP-ABE. Further, constrained devices, like sensors, often need to encrypt data as they are usually led to send sensitive data they collect to more powerful devices like storage servers. This paper proposes a novel approach for employing CP-ABE on highly resource-constrained sensor nodes in the IoT environments. The proposed approach exploits collaboration between heterogeneous nodes, to make feasible the implementation of CP-ABE in an IoT environment, by delegating costly operations to a set of assisting nodes. An analysis is conducted to verify that the proposed solution accomplishes safely and efficiently its objective.**

***Index Terms*—CP-ABE, Internet of Things, Access Control, Delegation**

I. INTRODUCTION

The Internet of Things (IoT) is a novel paradigm where many of the objects that surround us will be connected to the internet, these objects can be Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. -, and through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals. Inevitably, the Internet of Things, by its heterogeneous nature, raises lot of security issues; therefore, building an efficient security mechanism in IoT is primordial before its deployment.

Ciphertext Policy Attribute based Encryption (CP-ABE) [5], conceptually similar to traditional access control methods such as Role-Based Access Control (RBAC), is a powerful and efficient mechanism that can be widely applied to realize access control in many applications in the internet of things including medical systems and education systems. The only complaint we made to CP-ABE is its complexity and its high overhead. This drawback becomes more serious in an IoT context because of the presence of a highly resource-constrained devices.

In this work, we tried to exploit the heterogeneous nature of the internet-of-things to make feasible the use of the CP-ABE scheme in an Internet of things environment, the main idea of our contribution is to displace the burden from highly resource-constrained devices to unconstrained one by delegating heavy operations in CP-ABE scheme to neighbor unconstrained nodes.

The remainder of this paper is organized as follows. We discuss related work in Section II. In Section III we present an overview of the operations of CP-ABE scheme. We present our solution in Section IV. In Section V we give a security and performance analysis of our solution. Finally, we draw our conclusions and futures works in Section VI.

II. RELATED WORKS

The implementation of multi-authority Ciphertext Policy Attribute Based Encryption scheme [6] in resource-constrained environment (energy harvesting wireless sensor network) was proposed for the first time in [4]. Authors have exploited the surplus of the energy harvested, that cannot be stored in batteries, to compute some parameters that need lot of computing and which will be used in the near future. These parameters will be stored in the cache memory of the sensor and will be retrieved when it is needed.

The idea to outsource some cryptographic computations has already been proposed in [1], [2] and [3]. In [1] and [2], Ben Said, Y. et al. propose a collaborative session key exchange method, wherein a highly resource-constrained node obtains assistance from its more powerful neighbors when handling costly cryptographic operations. In [3], Hohenberger and Lysyanskaya present protocols for the computation of modular exponentiation (arguably the most expensive step in public-key cryptography operations). Their protocol requires the client to interact with two non-colluding servers.

The main idea of our solution is to distribute calculations of CP-ABE encryption primitive, so that resource-constrained objects can delegate the most consuming operations (mainly exponentiations) to unconstrained nodes of the network. This delegation preserves the security and robustness of CP-ABE under the assumption that each resource-constrained entity shares pairwise keys with at least two trusted unconstrained nodes in it neighborhood.

III. BACKGROUND

In this section, we will give a short overview about cipher policy attribute-based access control [5].

A. Bilinear Maps

Let G_0 and G_1 be two multiplicative cyclic groups of prime order p. Let g be a generator of G_0 and e be a bilinear map, $e : G_0 \times G_0 \rightarrow G_1$. the bilinear map e has the following properties:

1) Bilinearity: for all $u, v \in G_0$ and $a, b \in Z_p$, we have $e\left(g^a, g^b\right) = e\left(u, v\right)^{ab}$.
2) Non-degeneracy: $e\left(g, g\right) \neq 1$.

We say that G_0 is a bilinear group if the group operation in G_0 and the bilinear map e are both efficiently computable. Notice that the map e is symmetric since $e\left(g^a, g^b\right) = e\left(g, g\right)^{ab} = e\left(g^b, g^a\right)$.

B. CP-ABE Model

In CP-ABE model [5], private keys will be identified with a set S of descriptive attributes. A party that wishes to encrypt a message will specify a policy (access tree) that private keys must satisfy in order to decrypt.

Access tree T. 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 $k_x = 1$.

Few functions are defined to facilitate working with access trees:

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

Satisfying an access tree. Let T be an access tree with root r. Denote by T_x the sub-tree of T rooted at the node x. Hence T is the same as T_r. If a set of attributes γ satisfies the access tree T_x, we denote it as $T_x\left(\gamma\right) = 1$. We compute $T_x\left(\gamma\right)$ recursively as follows. if x is a non-leaf node, evaluate $T_{x'}\left(\gamma\right)$ for all children x' of node x. $T_x\left(\gamma\right)$ returns 1 if and only if at least k_x children return 1. if x is a leaf node, then $T_x\left(\gamma\right)$ returns 1 if and only if $att\left(x\right) \in \gamma$.

C. CP-ABE Algorithm

An ciphertext-policy attribute based encryption scheme consists of four fundamental algorithms: *Setup*, *Encrypt*, *KeyGen*, and *Decrypt*. In addition, there is an optional fifth algorithm *Delegate*.

Let G_0 be a bilinear group of prime order p, and let g be a generator of G_0. In addition, let $e : G_0 \times G_0 \rightarrow G_1$ denote the bilinear map. A hash function $H : \{0,1\}^* \rightarrow G_0$ will also be used.

- **Setup:** The setup algorithm takes as input the implicit security parameter and it outputs the public parameters PK and a master key MK. it chooses a bilinear group 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:

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

and the master key is:

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

- **Encrypt(PK,M,T):** The encryption algorithm encrypts a message M under the tree access structure T. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree T. These polynomials are chosen in the following way in a top-down manner, starting from the root node 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 $s \in Z_p$ and sets $q_R\left(0\right) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x\left(0\right) = q_{parent}\left(x\right)\left(index\left(x\right)\right)$ and chooses d_x other points randomly to completely define q_x.
Let, Y be the set of leaf nodes in T . The ciphertext is then constructed by giving the tree access structure T and computing

$$CT = \Big(T, \tilde{C} = Me\left(g, g\right)^{\alpha s}, C = h^s, \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H\left(att\left(y\right)\right)^{q_y(0)}\Big) \quad (3)$$

- **KeyGen(MK,S):** The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK. The algorithm first chooses a random $r \in Z_p$, and then random $r_j \in Z_p$ for each attribute $j \in S$. Then it computes the secret key as:

$$SK = \Big(D = g^{(\alpha+r)/\beta}, \\ \forall j \in S : D_j = g^r.H\left(j\right)^{r_j}, D'_j = g^{r_j}\Big) \quad (4)$$

- **Decrypt(PK,CT,SK):** The decryption algorithm takes as input the public parameters PK, a ciphertext CT, which contains an access policy T, and a private secret key SK, which is a private key for a set S of attributes.
If the set S of attributes satisfies the access structure A then the algorithm will decrypt the ciphertext and return a message M.
The decrypt algorithm is not detailed here as we focused our work on the encryption algorithm, the reader can get more information about it in the original paper [5].
- **Delegate(SK,$\tilde{S}$):** The delegate algorithm takes as input a secret key SK for some set of attributes S and another set $\tilde{S}$ such that $\tilde{S} \subseteq S$. The secret key is of $SK = \left(D, \forall j \in S : D_j, D'_j\right)$. The algorithm chooses a random $\tilde{r}$ and $\tilde{r_k} \forall k \in \tilde{S}$. It outputs a secret key $\tilde{SK}$ for the set

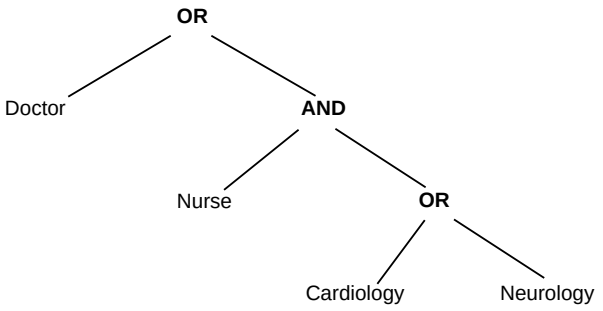

Figure 1: An example of an access tree

of attributes $\tilde{S}$. The new secret key is:

$$\tilde{SK} = \Big(\tilde{D} = Df^{\tilde{r}}, \\ \forall k \in \tilde{S} : \tilde{D}_k = D_k g^{\tilde{r}} H(k)^{\tilde{r}_k}, \tilde{D'}_k = D'_k g^{\tilde{r}_k}\Big)$$

D. How it works

Let us consider a system where a set of users $U = \{u_1, u_2, \cdots, u_n\}$ access a shared data depending on an access policy. The access policy is materialized by a access structure like the one illustrated in figure 1. Each user will be assigned a subset of attributes $D_i = \{d_1, d_2, \cdots, d_x\}$ according to his/her role in the system. The set of the whole attributes used in the system $A = \{a_1, a_2, \cdots, a_k\}$ (where $\forall i \in \{1, \cdots, n\}, D_i \subseteq A$) is maintained by an *Attribute Authority* AA.

In the bootstrap phase, the attribute authority generates a **Public Key** PK (formula 1) and publishes it to all the entities in the system, it creates also a **Master Key** MK (formula 2) which is kept secret. After that, the attributes authority assigns to each user in the system a **Secret Key** SK (formula 4) calculated based on its attributes. At the end of this phase, each user of the system has a secret key constructed based on its attribute set. Its also knows the public key and the set of whole attribute.

When a user wishes to share some data with some other users in the system verifying an access policy[1], it constructs the corresponding access tree and encrypts the data using the encryption algorithm (subsection III-C). After that, it sends the ciphertext to a remote server. The ciphertext is, then, stored in the remote server.

Users which want to access the data, must verify the access policy defined in the ciphertext, otherwise, it is impossible for them to decrypt it.

IV. Proposed solution

As stated in [5], The main source of overhead in the CP-ABE is in the *encryption* and the *generation of secret keys* algorithms, this is because of the number of exponentiation to be computed. Indeed, there are $2|Y| + 2$, and $2|S| + 2$ exponentiations to compute (formulas 3 and 4) in both encryption and secret key generation algorithms respectively, where Y and S are the set of leaf nodes in the access tree and the set of attributes associated to an entity respectively. The decryption algorithm poses no problem as authors have proposed an improvement in [5].

The key generation algorithm is executed at the bootstrap phase by the attribute authority (AA) to generate secret keys to each user (entity) considered in the system, so it doesn't raise any problem as the attribute authority is generally a powerful server and secret keys are generated only once. In contrast, the encrypt algorithm is executed whenever an entity wants to send/share sensitive data to other entities in the network, and in an Internet of Things context, resource-constrained objects such as sensors are often led to send encrypted data to more powerful entities.

In our work, we focus on the encryption algorithm and propose a computation offloading scheme to reduce the induced overhead at resource-constrained objects. The main idea is to delegate the computation of exponentiation to other trusted neighbor devices called "assistant nodes". When a resource-constrained device wants to encrypt a message, it looks for trusted unconstrained nodes in its neighborhood and it delegates to them the costly operations (exponentiations). Hence, the burden due to CP-ABE encryption primitive is displaced from resource-constrained devices to unconstrained ones.

A. Network model

As mentioned above, the IoT infrastructure considered here is heterogeneous i.e. the objects in the network do not all have the same capabilities in terms of computing power and energy resources. Indeed, we can distinguish three categories of nodes:

- *Resource-constrained devices,* unable to support the computational cost of the CP-ABE encrypt operation, while nevertheless must send sensitive data (e.g. sensor nodes).
- *Unconstrained devices,* therefore able to perform costly operations. These nodes may either be dedicated assisting servers or nodes belonging to the same local infrastructure, though being less impacted by energy constraints (e.g. having energy harvesting capability, line-powered devices).
- *Remote servers,* characterized by high computing power and high storage capacities and have a functional role in the system, they are intended to store encrypted messages received from system's objects and make them available to any request from any entity of the system.

B. Assumptions

1) For each resource-constrained device there are at least two trusted unconstrained devices in its neighborhood.
2) Every resource-constrained device shares pairwise keys with two or more unconstrained devices in it neighborhood. These keys may have been generated during a specific bootstrapping phase.

[1]The user does not know a priori the identity of those who are supposed to read the data

3) Every object in the system shares pairwise keys with the remote servers.

C. Per-phase Exchanges Descriptions

1) *Phase 1:* The resource-constrained device (noted A in Figure-2) which has to encrypt a message M, select n trusted nodes among those situated in its neighborhood, these nodes (assistant nodes) will assist the device A in the encryption operation. Notice that the existence of these trusted nodes in the neighborhood of A is assumed in the network model.
2) *Phase 2:* In this phase, the device A defines the access policy for the message M and constructs the corresponding access tree T. After that, it generates the polynomials q_y (for all $y \in Y$) assigned to nodes of T as described in the encryption algorithm in the subsection III-C. instead of calculating the cyphertext CT with the formula 3, the device A splits s (s represents $q_R(0)$) into n parts s_i, such that the sum of all s_i gives s (equation 5). Likewise, it splits every $q_y(0)$ into n parts $q_y^{(i)}(0)$ for all $y \in Y$ (equation 6), where Y is the set of leaf nodes of the access tree.

$$s = \sum_{i=1}^{n} s_i \tag{5}$$

$$\forall y \in Y : q_y(0) = \sum_{i=1}^{n} q_y^{(i)}(0) \tag{6}$$

3) *Phase 3:* To each assistant node P_i ($where\ i = 1, \cdots, n$), the device A securely sends s_i, and for all $y \in Y$, it sends $H(att(y))$ and $q_y^{(i)}(0)$. These information are transmitted encrypted with the shared key between A and the assistant node P_i (Assumption 2).
4) *Phase 4:* Each assistant node P_i computes $e(g,g)^{\alpha s_i}$ (formula 7), h^{s_i} and sends it back to the device A after encryption, it computes also $H(att(y))^{q_y^{(i)}(0)}$ and $g^{q_y^{(i)}(0)}$ for all $y \in Y$ and sends them encrypted to the server using the shared key between them (Assumption 3). We should call the reader's attention here to the fact that h, $e(g,g)^{\alpha}$ and g are parts of the public key PK, therefore the device A has not so send them in the previous phase.

$$e(g,g)^{\alpha s_i} = \left(e(g,g)^{\alpha}\right)^{s_i} \tag{7}$$

5) *Phase 5:* After receiving responses from the assistant nodes, the device A decrypts them and uses them to compute:

$$\begin{aligned}\tilde{C} &= M \prod_{i=1}^{n} (e(g,g)^{\alpha s_i}) \\ &= M e(g,g)^{\alpha(\sum_{i=1}^{n} s_i)} \\ &= M e(g,g)^{\alpha s}\end{aligned} \tag{8}$$

and send the result to the remote server after encrypting it with the shared key between it and the server (Assumption 3).
6) *Phase 6:* The server receives the the calculated parts form the assistant nodes and it uses them to compute:

$$C = \prod_{i=1}^{n} h^{s_i} = h^{\sum_{i=1}^{n} s_i} = h^s \tag{9}$$

$$\begin{aligned}\forall y \in Y,\ C_y &= \prod_{i=1}^{n} g^{q_y^{(i)}(0)} \\ &= g^{\sum_{i=1}^{n} q_y^{(i)}(0)} \\ &= g^{q_y(0)}\end{aligned} \tag{10}$$

$$\begin{aligned}\forall y \in Y,\ C'_y &= \prod_{i=1}^{n} H(att(y))^{q_y^{(i)}(0)} \\ &= H(att(y))^{\sum_{i=1}^{n} q_y^{(i)}(0)} \\ &= H(att(y))^{q_y(0)}\end{aligned} \tag{11}$$

It receives also the access tree T and $\tilde{C}$ from the device A and construct the ciphertext using all these parts as follows:

$$CT = \left(T, \tilde{C}, C, \forall y \in Y : C_y, C'_y\right) \tag{12}$$

Notation	Description
PK	Primary Key
MK	Master Key
G_0, G_1	Bilinear groups
$H(.)$	A one-way hash function
e	Bilinear map
g	Generator of G_0
T	An access tree
Y	The set of leaf nodes in the access tree
q_y	Polynomial associated to the node $y \in Y$
S	Set of attributes associated to a user
AA	Attribute Authority
M	Message to be encrypted
s	$q_R(0)$
K_{XY}	Pairwise key between X and Y
$\{M\}_K$	Message M encrypted by the key K

Table I: Notation table

V. Discussion

In this section we present a security and performance analysis of our proposed solution and prove its *safety* and *efficiency*.

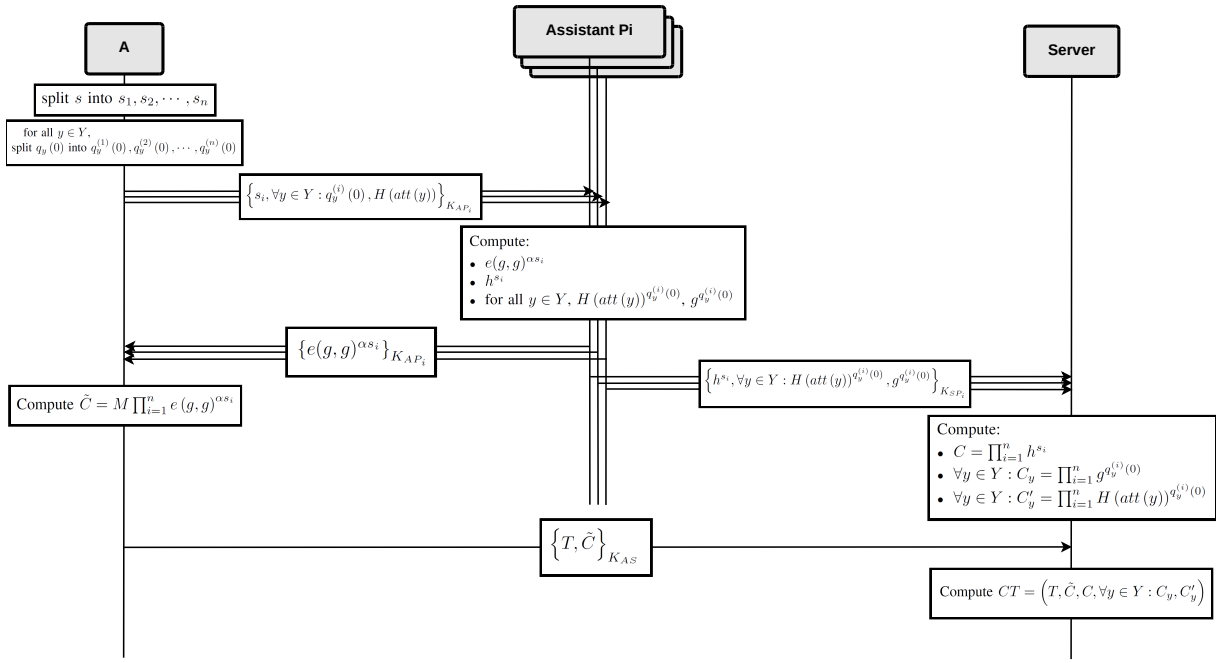

Figure 2: Collaborative CP-ABE Encryption Algorithm.

A. Security analysis

A stated in [5], to decrypt a message an attacker clearly must recover $e(g,g)^{\alpha s}$. Authors gave in [5] an analysis of the impossibility to an attacker to recover it thanks to the random values used in generating secret keys and encrypting messages.

Another way to decrypt the message, without having the necessary attributes, is to calculate/guess the value of s. An attacker who tries to eavesdrop the communication between the device A and the assistant nodes in the phase 2, cannot recover the value of s because of secure communication between the device A and the assisting assistant nodes (Assumption 2), and because of the trustworthiness of the assistant nodes, they will not collide to recover the value of s (Assumption 1 and 2).

It is important to notice that even the storage server cannot recover the original message, indeed, the partial part calculated by the assistant nodes are not sufficient to recover the value of s.

B. Performance analysis

1) Computation cost: It is important to notice that the comparison given in table II do no consider the step of the access tree T construction (the generation of a polynomial q_x for each node x of the tree T), as the latter generates the same overhead in both CP-ABE and C-CP-ABE.

According to table II we remark that the costly operations are displaces from the device A (resource-constrained node) to the assistant nodes and the server. Indeed, in our approach, the node A has no exponentiation to computes against $2 + 2|Y|$ in the standard CP-ABE. Figure 3 presents a comparison between the number of exponentiations computed by the node

		Number of Exponentiation	**Number of Multiplication**
CP-ABE	Device A	$2 + 2\|Y\|$	1
	Server	0	0
C-CP-ABE	Device A	0	n
	Assistant node	$2 + 2\|Y\|$	0
	Server	0	$(n-1) + (2n-2)\|Y\|$

Table II: Computation cost

A during the encryption primitive of CP-ABE scheme and our approach, we notice that our approach removes all the exponentiations, while the latter increases linearly with the number of leaf nodes in the access tree. The number of multiplications computed by the node A in our approach is the same as the number of assistant nodes chosen, but computing a multiplication is cheaper than exponentiation, and the number of assistant node is often chosen moderately small, and therefore, this will not degrade the performance of our solution.

Figure 4 shows how our approach displaces the burden (costly operations) from device A to the assistant nodes and the storage server. These data are obtained by setting $|Y| = 20$ (twenty leaf nodes in the access tree) and $n = 5$ (five assistant nodes).

2) Communication cost: The solution we proposed for implementing CP-ABE in an Internet of Things environment achieves its objective through some simple exchanges with

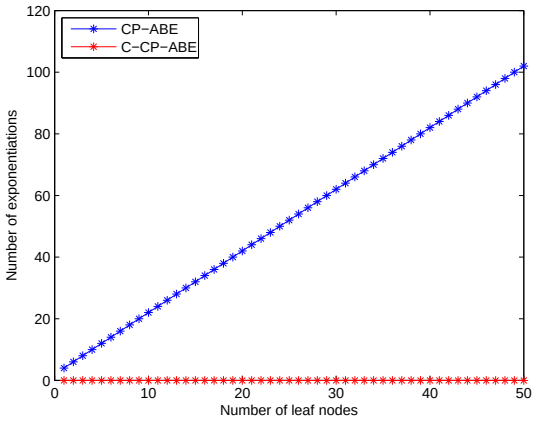

Figure 3: Comparison between CP-ABE and CCP-ABE in terms of exponentiations computed by device A

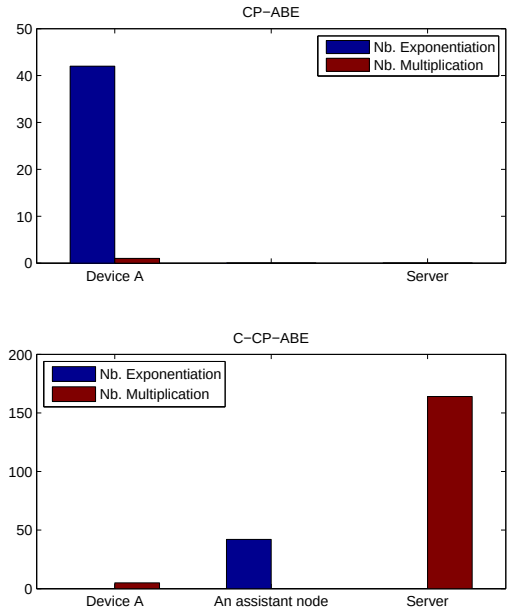

Figure 4: Comparison between CP-ABE and CCP-ABE in terms of exponentiations and multiplications for each part

neighbor nodes of the system. The energy consumed while exchanging with assistant nodes is less than the energy saved by delegating costly operation of CP-ABE encryption primitive to neighbor unconstrained nodes of the system. We believe that choosing an adequate number of assistant nodes, for example five ($n = 5$), is a good compromise between robustness and efficiency.

VI. Conclusions and Future Works

This paper presented a novel collaborative approach for using CP-ABE scheme in resource-constrained nodes. Our solution takes advantage from the heterogeneity of the network to distribute the overhead due to CP-ABE encryption primitive. Furthermore, our approach achieves its objective through simple exchanges with neighbor unconstrained nodes which are less energy consuming than CP-ABE encryption primitive.

In the future, it would be interesting to carry out simulations, in order to precisely quantify the energy savings. An automatic verification of security with an automated validation tool like AVISPA is also in our perspectives.

VII. 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] Yosra Ben Saied, Alexis Olivereau and Maryline Laurent, *A Distributed Approach for Secure M2M Communications*, 2012.

[2] Y. Ben saied, A. Olivereau and D. Zeghlache, *Energy Efficiency in M2M Networks: A Cooperative Key Establishment System*, 3rd Int. Congress on Ultra Modern Telecommunications and Control Systems (ICUMT), 2011.

[3] S. Hohenberger and A. Lysyanskaya. How to securely outsource cryptographic computations. In Proceedings of TCC, 2005.

[4] Giuseppe Bianchi, Angelo T. Capossele, Chiara Petrioli, Dora Spenza, *AGREE: exploiting energy harvesting to support data-centric access control in WSNs*,2013.

[5] J. Bethencourt, A. Sahai, B. Waters, *Ciphertext-Policy attribute-based encryption*, in: Proc. of IEEE SP 2007, Oakland, California, USA, 2007, pp. 321-334.

[6] A. Lewko, B. Waters, *Decentralizing attribute-based encryption*, in: Proc. of ACR EUROCRYPT 2011, Tallinn, Estonia, 2011, pp. 568-588.