Reinforcing IoT-Enforced Security Policies

Nouha Oualha
CEA, LIST, Communicating Systems Laboratory
91191 Gif-sur-Yvette CEDEX, France
Email: nouha.oualha@cea.fr

***Abstract*—To provide confidentiality protection and access control to large amount of data generated by the Internet of Things (IoT), one promising approach lies on the use of Attribute-Based Encryption (ABE). However, the ABE-based approach needs to cope with the resource constraints of IoT devices, by providing data encryption and access control only under small access policies. With the goal to mitigate this limitation inherent to the IoT, this paper introduces a new technique that relies on semi-trusted intermediate entities to reinforce large access policies derived from small policies enforced by IoT devices without revealing the encrypted data.**

***Index Terms*—Internet of Things, Attribute-based Encryption, access policy, encryption.**

I. Introduction

The number of devices connected to the Internet of Things (IoT) is predicted to grow rapidly, and accordingly increasingly large amount of data is being generated by these IoT devices. Related to different application domains e.g., demotic and home automation, remote monitoring, smart healthcare, smart cities, and smart grid, most of the generated data are sensitive information e.g., business, client, patient, or personal information, that require confidentiality protection and access control. The provided security solution should be flexible and fine-grained enough to handle a large number of users performing different tasks on data. One promising approach that allows to provide both data confidentiality and access control at once, is Attribute-based Encryption (ABE). ABE is a public key encryption cryptosystem that guarantees also fine-grained access control to the data. Only authorized data users holding the right set of attributes that satisfy the access policy associated with the data can decrypt the ciphertext.

In cloud computing and large-scale based applications, ABE has gained a lot of attention, but little consideration has been paid for IoT applications, despite the promising advantages of ABE. The proposed ABE schemes fall short in terms of performance. As demonstrated in [6], [7], and [8], it is possible, already today, to deploy ABE schemes on resource-constrained IoT devices like sensors, but for only smaller policies. The main limitation experienced with such devices, shown in [8], is their RAM size. This paper introduces operations on access structures that are used to build large access policies from small ones. More specifically, with the help of a semi-trusted entity (e.g., resource-rich IoT gateway, cloud service, etc.), a new ABE ciphertext is built under a new more restrictive access policy which is larger in terms of its number of attributes. The new ciphertext is derived from an ABE ciphertext, but without decrypting the ciphertext. As an illustrative example, we consider a ciphertext of data being ABE encrypted under an access policy:

$$\Gamma_0 = (organization_1 \text{ or } organization_2)$$

i.e., the ciphertext can be decrypted by users from either $organization_1$ or $organization_2$. Using our solution, a semi-trusted entity reinforce the ciphertext by inserting the policy Γ_1 that $organization_1$ would like to enforce. Thus, the new generated ciphertext will be encrypted under the policy:

$$\Gamma_0' = ((organization_1 \text{ and } \Gamma_1\ ) \text{ or } organization_2))$$

i.e., the ciphertext can be decrypted by users from either $organization_2$ or $organization_1$ only if these latter satisfy the policy Γ_1. By repeating the insertion operation (e.g., new access policy for $organization_2$), the proposed solution allows to build a ciphertext under a much more larger access policy.

Paper outline: The paper is organized as follows: section II surveys existing works on ABE and lightweight solutions, section III provides a background overview of access policies computed using Monotone Span Programs (MSPs) and ABE schemes, section IV describes the operations performed on MSPs, section V presents the system design, threat model, and the proposed solution for policy reinforcement, section VI evaluates the security and the performance of the proposed solution, and finally section VII concludes the paper.

II. Realted work

Research efforts on ABE have focused on devising new ABE cryptographic constructions for new ABE schemes (e.g., [1], [2], and [3]) or new features, e.g., multi-authority (e.g., [5]), key revocation, rather than on performance efficiency of ABE (e.g., [4]). Some efforts have been made to design lightweight ABE-based solutions adapted to the resource constraints of some IoT devices. For instance, authors in [4] propose to tradeoff the CPU usage with the memory space by applying pre-computing techniques to ABE such that the device uses the pre-computed values instead of computing expensive scalar multiplications.

Due to the increasing use of IoT devices, some implementations of ABE schemes have been proposed on smartphone devices. In [6], the authors have implemented a C library of some main ABE schemes for Android operating system. Similarly, the authors in [7] have implemented these ABE schemes in Java on an Atom-based Android phone, and evaluated the performance of the implemented schemes. With an acceptable amount of resources in terms of CPU and energy, both implementations have demonstrated the feasibility of effectively using ABE on smartphone devices. From their performance measurement results, the encryption algorithm is the less expensive algorithm in the implemented ABE schemes, even though the required resources increase with the number of attributes in the access policy. The authors in [8] considered a more resource constrained type of IoT devices: sensor devices. They demonstrated the feasibility of running ABE on resource-constrained sensors, and more importantly, their findings revealed that the major limiting factor of using ABE on these devices is their RAM size that limits the size of the access policy. In this paper, we propose a new technique to build ABE ciphertexts under large access policy from ABE ciphertexts with smaller access policies. The proposed technique is designed to lighten computations on resource-constrained devices like sensors and actuators, by considering small access policies for ABE encryption. To the best of our knowledge, this paper is the first to undergo such operations on access structures for access policies associated with ABE schemes.

III. Preliminaries

In the following, we provide a brief background on some math operations, an MSP-based representation of access policies, and a description of the Waters ABE scheme.

A. Kronecker product

The Kronecker product, denoted by $\otimes$, is an operation on two matrices of arbitrary size resulting in a block matrix. Let A be a $m_1 \times d_1$ matrix, B be a $m_2 \times d_2$ matrix, and C and D be two matrices. $A \otimes B$ is defined as an $m_1 m_2 \times d_1 d_2$ matrix:

$$A \otimes B = \begin{bmatrix} m_{11}B & \dots & m_{1d_1}B \\ \vdots & \ddots & \\ m_{m_1 1}B & \dots & m_{m_1 d_1}B \end{bmatrix}$$

The Kronecker product is the generalization of the outer product from vectors to matrices. It should not be confused with the Euclidian inner product of two vectors u, v, being $\langle u, v\rangle = u^T v$.

B. Monotone Span Programs

Access policies used in practice are represented using Boolean formulae (e.g., ((Patient or Relative) and Doctor)). Apart from Bethencourt et al.s ABE scheme [1] and few original proposals (e.g., [4]) that used tree structures, the majority of ABE schemes (e.g., [2], [3], [5]) have expressed access policies using a more general representation, called Monotone Span Programs (MSPs). The efficiency of these schemes did not decline by using the more general MSP-based representation.

The access policy is expressed using an MSP matrix M over the attributes in the policy. The authors in [5] (appendix G) showed a simple method to convert a Boolean formula with AND and OR gates into an MSP matrix with $0, \pm 1$ values. Using the Benaloh- Leichter secret-sharing scheme, the ABE scheme proposed in [18] can realize ABE MSPs with also $0, \pm 1$ values.

Definition 1 (Monotone Span Program) *[9] A Monotone Span Program (MSP) is a quadruple (F, M, t, ρ), where F is a finite field, M is a matrix (with m rows and $d \leq m$ columns) over F, $\rho : \{1, ..., m\} \to \{1, ..., n\}$ is a surjective function and $t = (1, 0, ..., 0)^T \in F^d$ is called target vector. The size of M is the number m of rows and is denoted as $size(M)$.*

For reasons of simplicity and brevity, the MSP quadruple (F, M, t, ρ) is shortened to (M, ρ) and sometimes to the matrix M. In the following, rows of a matrix M owned by an attribute or a set of attributes S are denoted as $(M)_S$, and columns indexed from i to j are denoted as $[M]_{i,...,j}$. A $m \times n$ zero matrix is represented by $0_{m,n}$.

C. Background on ABE

In the literature, ABE comes in two flavors: Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). The first type of ABE schemes associates data with the attributes and the access policy with the users, whereas, the second type inverses these associations. This makes the second type of schemes more flexible, since it allows the data owner (encryptor) to update the access policy associated with the data without updating the decryption keys associated with the users. Moreover, the second type of ABE schemes allows to better enforce attribute-based access control. In this paper, we focus on the second type of ABE schemes, CP-ABE.

To demonstrate the practicability of our solution, one popular CP-ABE scheme, the Waters scheme [2], is extended with our new algorithm based on the proposed reinforcement technique. The technique, of course, can be applied to other CP-ABE schemes (e.g., [1], [3], [5]) in the same fashion. The Waters scheme is briefly presented in the following using the MSP-based description provided in [3] (Appendix E with a scheme built using asymmetric bilinear maps):

- $Setup(1^\lambda) \to (pk, msk)$: The algorithm sets three groups G_1, G_2, and G_T of prime order $p = \Theta(\lambda)$ supporting a non-degenerate efficiently computable bilinear map $e : G_1 \times G_2 \to G_T$, in addition to two generators P and Q for respectively G_1 and G_2. It chooses two random exponents $\alpha, a \in Z_p$ and a set of random elements $H_1, \dots, H_U \in G_1$ that are associated with the U attributes in the system. Then, it computes $pk = (P, Q, a.P, e(P, Q)^\alpha, H_1, \dots, H_U)$, and $msk = (\alpha.P)$. The algorithm outputs (pk, msk).
- $KeyGen(msk, S) \to (sk)$: The algorithm picks a random number $t \in \mathbb{Z}_p$. Then, it computes $K = \alpha.P + t.(aP)$, $L = t.Q$, and $\forall y \in S \subseteq U : K_y = t.H_y$ as the key sk. The algorithm outputs sk.
- $Encrypt(pk, (M, \rho), msg) \to (ct)$: For an MSP matrix of m rows and d columns, the algorithm chooses random numbers $s, y_2, \dots, y_d, r_1, \dots, r_m \in \mathbb{Z}_p$. Let the vector $v = (s, y_2, \dots, y_d)$, $\forall 1 \leq i \leq m : \mu_i = \langle (M)_i, v\rangle$, where $(M)_i$ is the ith row of M. Then, it computes $C = e(P, Q)^{\alpha s}.msg$, $C' = s.Q$, and $\forall 1 \leq i \leq m : C_i =$

$\mu_i.(a.P) + (-r_i).H_{\rho(i)}$, $D_i = r_i.Q$ as the ciphertext ct. The algorithm outputs $ct = ((M, \rho), C, C', \forall 1 \le i \le m : C_i, D_i)$.

- $Decrypt(pk, ct, sk) \rightarrow (msg', \perp)$: If the set of attributes S in sk does not satisfy the MSP (M, ρ) in ct, then the algorithms outputs $\perp$. Otherwise, it exists a set of constants $\{\nu_i\}_{i\in I}$, such that $\sum_{i\in I} \nu_i (M)_i = (1, 0, \ldots, 0)$, where I is the set of indexes of the rows of M mapped to attributes in S. Then, it computes $msg' = C.e(\prod_{i\in I} C_i^{\nu_i}, L). \prod_{i\in I} e(K_{\rho(i)}, D_i^{\nu_i})/e(K, C')$ and outputs msg'.

IV. Composition of MSPs

In this section, we investigate useful operations on MSPs proposed in [9] and [10]. Such operations allow to begin with small access policies with a few attributes and build them up to large access policies with higher number of attributes.

A. Restrictions

Let Γ be a monotone access structure expressed with the MSP matrix M, and defined on the set of attributes U, and let $Q \subseteq U$. The restriction of Γ at Q, $\Gamma_{|Q}$, is a monotone access structure defined on $U \setminus Q$ such that for each $S \subseteq U \setminus Q$, $S \in \Gamma_{|Q}$ means $S \in \Gamma$. The authors in [9] showed that the MSP matrix $M_{|Q}$ of $\Gamma_{|Q}$ is formed by removing the rows in M, which correspond to the members of Q. For a CP-ABE ciphertext built under an access structure Γ, the sets that can be removed are defined as $Q \in S$ such that there exists a set $S \in U \setminus Q$ and $S \in \Gamma$, because, otherwise the ciphertext cannot be decrypted.

B. Insertions

We consider two monotone access structures Γ_1 and Γ_2 defined on the attribute sets U_1 and U_2 respectively. Γ_1 and Γ_2 are expressed with MSPs M_1 and M_2 of size m_1 and m_2 respectively. We consider also a function ρ_1 which gives for every row i in M_1 the associated attribute $a_i \in U_1$ (denoted by $\rho(i)$). For an attribute $a_x \in U_1$, the insertion operation of Γ_2 at a_x in Γ_1, denoted $\Gamma_1(a_x \rightarrow \Gamma_2)$, is defined to be the monotone access structure on the set $(U_1 \setminus \{a_x\}) \cup U_2$ such that for any $S \subseteq (U_1 \setminus \{a_x\}) \cup U_2 : S \in \Gamma_1(a_x \rightarrow \Gamma_2) \iff S \cap U_1 \in \Gamma_1$, or $((S \cap U_1) \cup \{a_x\} \in \Gamma_1$ and $S \cap U_2 \in \Gamma_2)$. In other words, $\Gamma_1(a_x \rightarrow \Gamma_2)$ is the monotone access structure Γ_1 with the attribute a_x "replaced" by the sets of Γ_2. As demonstrated in [9], there exists an MSP M computing the access structure $\Gamma_1(a_x \rightarrow \Gamma_2)$ of size equal to $m_1 + (m_2 - 1)|\rho^{-1}(a_x)|$. The matrix can be computed as:

$$M = \begin{bmatrix} (M_1)_{a_x} \otimes u & \tilde{M_2} \\ (M_1)_{U_1 \setminus a_x} & 0 \end{bmatrix} \tag{1}$$

Where, $(M_1)_{a_x}$ is the set of rows owned by the attribute a_x, $(M_1)_{U_1\setminus\{a_x\}}$ is the remainder of rows of M_1, u is the first column of M_2 (i.e., $M_2 = [u (M_2)_{U_2\setminus\{u\}}]$), and $\tilde{M_2}$ is the remainder $(M_2)_{U_2\setminus\{u\}}$ repeated diagonally with the number of rows mapped to a_x (i.e., $\rho_1(a_x)$):

$$\tilde{M_2} = \begin{bmatrix} (M_2)_{U_2\setminus\{u\}} & 0 & \cdots \\ 0 & (M_2)_{U_2\setminus\{u\}} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{2}$$

It is worth noting that the computed MSP matrix M according to equations (1) and (2) still includes the MSP matrix M_1 of Γ_1 without the rows owned by a_x. As per the CP-ABE scheme built with an MSP-based approach, a ciphertext includes group elements that hide the used random vector $v = (s, y_2, \ldots, y_{d_1})$ applied to the MSP matrix. Hence, one way to operate an insertion of a new access policy Γ_2 over a ciphertext computed under the access policy Γ_1 without decrypting it, is to choose an access policy Γ_2 such that the elements from the original ciphertext are kept and used either without change or mapped with some affine transformation that preserves linear combinations between the group elements e.g., a translation transformation.

A non-trivial example of such insertion operation is the one given by $\Gamma_1(a_x \rightarrow (a_x \text{ and } \Gamma_2))$. The matrix M of the access policy $\Gamma_1(a_x \rightarrow (a_x \text{ and } \Gamma_2))$ can be computed as:

$$M = \begin{bmatrix} (M_1)_{a_x} & \tilde{M_2} \\ 0 & \\ (M_1)_{U_1\setminus\{a_x\}} & 0 \end{bmatrix}$$

The rows of the matrix M can be reorganized to give the matrix M defined below by the formula (3) that will be used in the remainder of this paper in order to compose a new policy $\Gamma_1(a_x \rightarrow (a_x and \Gamma_2))$ from the policy Γ_1.

$$M = \begin{bmatrix} & 0 \\ M_1 & (\tilde{M_2})_{a_x} \\ 0 & (\tilde{M_2})_{U_2\setminus\{a_x\}} \end{bmatrix} \tag{3}$$

C. Products

There are several special cases of the use of the insertion operation. We will consider one of them, which is the product of two access policies. If Γ_1 and Γ_2 are defined on U_1 and U_2 respectively. The product $\Gamma_1 \times \Gamma_2$ is defined as the monotone access structure on $U_1 \cup U_2$ such that for $A \subseteq U_1 \cup U_2$, $A \in \Gamma_1 \times \Gamma_2$ means $(A \cap U_1 \in \Gamma_1$ and $A \cap U_2 \in \Gamma_2)$. Let Γ_1 and Γ_2 be monotone access structures with MSPs M_1 of size m_1 and M_2 of size m_2 respectively. The authors in [9] have demonstrated that there exists an MSP M of size $m_1 + m_2$ computing the product $\Gamma_1 \times \Gamma_2$. The MSP matrix M can be computed by applying two times the insertion operation to the following matrix representing the access structure (a_1 and a_2):

$$\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

The matrix M is derived as:

$$M = \begin{bmatrix} u & -u & (M_1)_{U_1\setminus\{u\}} & 0 \\ 0 & v & 0 & (M_2)_{U_2\setminus\{v\}} \end{bmatrix} \tag{4}$$

Where u is the first column of M_1 (i.e., $M_1 = [u \quad (M_1)_{U_1\setminus\{u\}}]$), and v is the first column of M_2 (i.e., $M_2 = [v \quad (M_2)_{U_2\setminus\{v\}}]$).

From an already generated CP-ABE ciphertext under policy Γ_1, it is obvious to see that CP-ABE ciphertext under the policy product $\Gamma_1 \times \Gamma_2$ can be produced from the ciphertext under policy Γ_1, without having to decrypt the ciphertext. Indeed, new random values $(y_{d_1+1}, \ldots, y_{d_1+d_2})$ can be used to mask the rest of the produced matrix M. The columns of the matrix M computed in formula (4) can be reorganized to have the matrix hereafter that will be used throughout the rest of the report to build a new policy $\Gamma_1 \times \Gamma_2$ from an initial one Γ_1:

$$M = \begin{bmatrix} M_1 & -u & 0 \\ 0 & v & (M_2)_{U_2 \setminus \{v\}} \end{bmatrix} \tag{5}$$

V. Proposed reinforcement technique

This section describes the technique of reinforcement of the access policy that allows reducing the size of the access policy used for data encryption at end devices, such as IoT devices. The proposed technique relies on intermediate entities to reinforce the access policy, particularly with new attributes. To this end, we will first present a scenario of reference to introduce, for instance, new actors (i.e., intermediate entities) and their associated threat model, and then describe the new technique.

A. Scenario model

Our base scenario (using the AAA framework terminology [12]) includes: a cryptographically policy enforcement point (PEP) that encrypts data (e.g., an IoT device), a policy reinforcement point (PRP) that reinforces the ciphertext policy (e.g., IoT gateway, Cloud service), a policy administration point (PAP) that manages and delivers policies (e.g., IoT administration server) to both PEP and PRP, and finally a data user that decrypts data (if it has sufficient attributes). The proposed scenario model is illustrated in Figure 1. While the PAP is not required to be online, the PRP is required to reinforce policies between the PEP and the data user. The PAP and the PRP could be co-localized.

We assume that all communications between any two actors are at least integrity protected, i.e., all messages are authenticated and protected against replay attacks. Moreover, since the access policy used by the PEP to encrypt the data is less restrictive than the final ciphertext returned by the PRP, the PEP should use a supplementary encryption layer to protect the ciphertext, by means of common security measures (e.g., symmetric encryption) using keys pre-established between the PEP and the PRP. The PAP should ensure that the policy enforced by the PEP, Γ_0, is not satisfied by the attributes associated with the PRP if this latter is not authorized to access the encrypted message i.e., its attributes do not satisfy Γ'.

B. Threat model

This subsection describes the considered threat model in our proposal. Compared to a typical CP-ABE scheme model, our scenario model adds a new actor, the PRP. The considered adversary model associated to this actor is provided with definition 2. The definition is an informal description of the

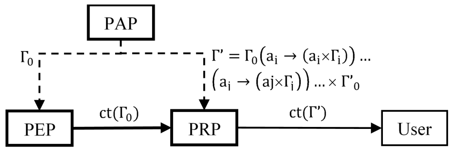
PAP
Γ_0
$\Gamma' = \Gamma_0(a_i \rightarrow (a_i \times \Gamma_i)) \ldots$
$(a_i \rightarrow (aj \times \Gamma_i)) \ldots \times \Gamma'_0$
PEP
ct(Γ_0)
PRP
ct(Γ')
User

Fig. 1. Scenario model

adversary model. A more formal definition of such model can be found in [12].

Definition 2 (Honest-but-curious adversaries) *[11] The honest-but-curious (HBC) adversaries follow correctly the protocol specification. However, the adversary keeps a record of all intermediate computations in order to learn information that is supposed to remain private.*

C. Policy reinforcement algorithm

We propose to extend ABE schemes with a new algorithm, the policy reinforcement algorithm. This algorithm is executed by the PRP, with the aim to reinforce the ciphertext with a new policy issued by the PAP.

Let $PolicyReinforceGen$ be a policy generator that on input the policy Γ_0, returns a new policy Γ'. The algorithm whereby a set of policies of this form $(a_i \times \Gamma_i)$ are inserted to replace a number of attributes $\{a_i\}$ in Γ_0, and also a number of policies Γ'_0 are multiplied to the result, is executed by the PAP. Restriction operations can be also applied to the result. The new policy (M', ρ') is computed as described in section IV, and transmitted to the PRP.

- $PolicyReinforce(pk, ct, (M', \rho')) \rightarrow ct^*$: The algorithm takes the input comprising the public key pk, the ciphertext ct that encloses an access policy (M, ρ), and a new access policy (M', ρ'), and produces a new ciphertext ct^* under the new access policy (M', ρ') as the output.

Let the MSP matrix M' produced by the $PolicyReinforceGen$ generator be a $m' \times d'$ matrix and the original MSP matrix M be a $m \times d$ matrix. Based on the equations (3) and (5), the rows and columns of M' are organized such that: $([M']_{1,\ldots,d})_{1,\ldots,m} = M$ and $([M']_{1,\ldots,d})_{m+1,\ldots,m'} = 0_{m'-m,d}$. The ciphertext ct^* is derived from ct without decrypting it. The vector v' that is applied to MSP matrix M' to produce exponents to be used in ct^*, is defined as the vector v_0 used in ct appended with $(d' - d)$ new random numbers, expressed as a vector v of size $(d' - d)$ (i.e., $v' = [v_0 \quad v]$). This leads to:

$$\langle (M')_{1,\ldots,m}, v' \rangle = \langle M, v_0 \rangle + \langle ([M']_{d+1,\ldots,d'})_{1,\ldots,m}, v \rangle$$
$$\langle (M')_{m+1,\ldots,m'}, v' \rangle = 0 + \langle ([M']_{d+1,\ldots,d'})_{m+1,\ldots,m'}, v \rangle$$

In the above sums, the left part is already computed in ct or null, and the right part that does not require any knowledge of v_0 is computed by the PRP during policy reinforcement. The new algorithm extended to Waters' scheme is described as follows:

- $PolicyReinforce(pk, ct, (M', \rho')) \rightarrow ct^*$: Let the MSP matrix M' be of m' rows and d' columns, the ciphertext

$ct = ((M, \rho), C, C', \forall 1 \leq i \leq m : C_i, D_i)$, and the original MSP matrix (M, ρ) in ct be of m rows and d columns. The algorithm chooses the following random numbers $(r_{d+1}, \dots, r_{d'}, y_{d+1}, \dots, y_{d'}) \in \mathbb{Z}_p$. Let $v = (y_{d+1}, \dots, y_{d'})$. Then, the algorithm computes $\forall 1 \leq i \leq m' : \mu_i = \langle ([M']_{d+1,\dots,d'})_i, v \rangle$, where $([M']_{d+1,\dots,d'})_i$ is the ith row of $[M']_{d+1,\dots,d'}$. Then, the algorithm computes $\forall (m+1) \leq i \leq m' : C_i = (-r_i).H_{\rho(i)}, D_i = r_i.Q$, and then, $\forall 1 \leq i \leq m' : C^*_i = \mu_i.(a.P) + C_i$. It outputs $ct^* = ((M', \rho'), C, C', \forall 1 \leq i \leq m' : C^*_i, D_i)$.

VI. Evaluation and discussion

This section provides an informal analysis of the security of the proposed solution. We argue that the proposed technique does not weaken the CP-ABE scheme and does not reveal information about the encrypted message, while achieving energy savings. Additionally, this section provides a performance analysis of the technique implemented on a real-world sensor platform.

A. Security considerations

For the analysis of the security of the proposed solutions, we distinguish between two types of attackers: an outsider attacker (i.e., not part of the scenario model) and an insider attacker (i.e., one of the actors in the scenario model).

In the considered scenario model, an outsider attacker has only access to the ciphertext $ct(\Gamma')$ assuming that the communications between the PEP and PRP are both confidentiality and integrity protected. Thanks to the security of the used CP-ABE scheme, the attacker cannot disclose the plaintext. Only authorized users with attributes satisfying the access policy are able to decrypt the ciphertext.

The scenario model adds a new actor, the PRP, that reinforces the ciphertext with a new policy. The PRP is assumed to be an honest-but-curious party. If this new actor is an insider attacker, it may attempt to guess the plaintext from received ciphertexts produced by PEPs, while performing tasks according to the desired scenario model and returning correct results from the PolicyReinforce algorithm. If the PRP is not authorized to access the ciphertext, its attributes should not satisfy the access policy enforced by the PEP, Γ_0, as ensured by the PAP, so that the PRP cannot decrypt the ciphertext thanks to the security of the employed CP-ABE scheme. Due to the definition of honest-but-curious model that we consider in this paper, collusion between the PRP and users that may verify the unreinforced policy Γ_0, are out of the scope of the paper.

B. Performance analysis

In this subsection, we describe our software implementation of the Waters scheme on real-world sensor platform. This performance analysis demonstrates the significant challenge posed by the access policy size i.e., the number of attributes.

1) Experimentation tools and platforms:: In our implementation, for the sensor network part, we used the Zolertia RE-Mote Revision B platform [13] that features an ARM Cortex M3 microcontroller that runs up to 32 MHz and includes 32 Kbytes of RAM and 512 Kbytes of Flash. We opted for the Contiki OS [14] as the operating system, which is an open source operating system for the IoT. The PC is a Dell Latitude-E6420 featuring Core i5 2,5GHz processor, with Ubuntu 14.04 LTS installed. All messages exchanged between the sensors and the PC are transmitted using the Constrained Application Protocol (CoAP) which is an Internet application protocol for constrained devices specified in RFC 7252 [20]. The sensor runs a CoAP server, and the user runs a CoAP client implemented on Python using the CoAPthon library [21]. Our system setting consists of five entities:

- Key distribution center (KDC): The KDC is not only responsible for the scheme setup and key distribution to data users, but also policy administration, i.e., the PAP role, and secure distribution of access policies to the PEP and the PRP. In our implementation, the KDC is emulated as a Linux PC. The Charm framework [15] is used to implement the setup and key generation algorithm.
- Gateway: The sensor part of the gateway is based on the Contiki OS on an OpenMote sensor platform [19] connected to the PC by USB. The gateway connects the sensor device to the PC.
- Sensor: The sensor, as the PEP, is also based on the Contiki OS on a Zolertia RE-Mote Revision B platform. The device communicates with the gateway using the IEEE 802.15.4 protocol. The encryption algorithm is implemented at the device using the cryptographic library Relic-toolkit [16].
- Proxy: The proxy, as the PRP, is connected to the IoT network through the PC part of the gateway, and its role is to re-encrypt the ciphertext transmitted by the sensor using an additional policy handed out by the KDC. Re-encryption implemented using the Charm framework.
- User: the data user is emulated as a Linux PC. The decryption algorithm is implemented based on the Charm framework.

2) Performance measurements and discussion:: This section is dedicated to analyzing the performance of the encryption algorithm on the sensor device. Energy consumption is of a primary importance for battery-powered devices. In order to estimate the energy consumption, we used tools provided by the contiki OS [17], and using the formula described in [22] (subsection VI-B). Even though, aside from a CoAP server, no other application is running on the sensor device that may use the 32 Kbytes of RAM memory, the maximum number of attributes in the ciphertext cannot reach more than 9 attributes. A maximum number of 12 attributes is obtained in [8] running a Content-Centric Networking stack directly over 802.15.4 radio without any underlying UDP or IP layers. This makes the communication stack lightweight leaving more RAM and processor for ABE encryptions. The authors demonstrated that

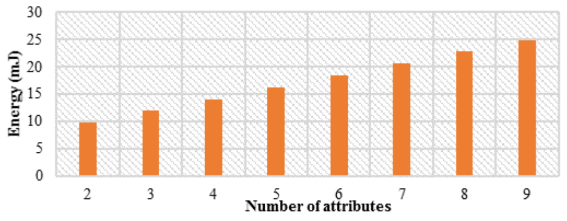

Fig. 2. Energy consumption of the Waters encryption algorithm.

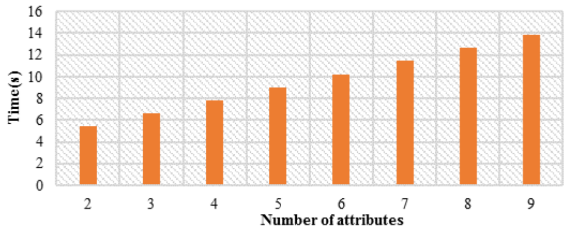

Fig. 3. Time consumption of the Waters encryption algorithm.

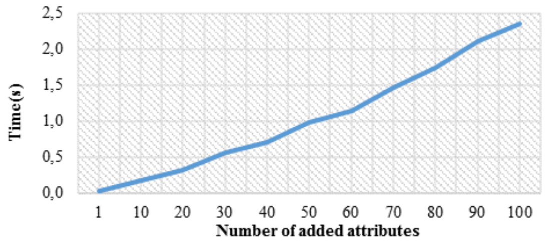

Fig. 4. Time consumption of proxy re-encryptions (Waters scheme).

this number falls to a maximum of 6 attributes for multi-authority ABE encryption operations.

Figure 2 shows the total energy consumption of the Waters scheme [2] encryption algorithm, varying the number of attributes in the access policy. The figure demonstrates that the energy consumption increases, approximately linearly, with the number of attributes. From 2 to 9 attributes, the consumed energy rises from around 9mJ to around 25mJ.

Figure 3 consolidates the result of the linear increase of the used resources with the number of attributes in the policy. For instance, the execution time increases from around 5s to around 14s by increasing the number of attributes from 2 to 9. For 9 attributes in the access policy, the execution time of the encryption algorithm at the sensor device reaches 13s. Whereas, as shown in Figure 4 adding 100 attributes to the access policy through proxy re-encryption, requires just around 2.4s. At the proxy, the execution time of the re-encryption operation increases almost linearly with the number of added attributes, but the slope of the line is smaller by a factor of 75, than for a resource-constrained device.

VII. Conclusion

The more devices are connected to the IoT, the more data are generated. The security solutions provided to such data, in several application domains, should be carefully designed, in particular to manage the different types of IoT devices, data users, and uses. In this context, CP-ABE offers an elegant approach to provide both data confidentiality protection and fine-grained access control. In this paper, we described a technique that allows to continue to benefit from the high degree of granularity of access policies while enforcing these policies on resource-constrained IoT devices. The proposed technique allows to build from ciphertexts under small access policies enforced, for instance, by IoT devices, ciphertexts under more complex and large access policies. This is achieved without ciphertext decryption, but by means of policy reinforcement at semi-trusted intermediate entities, such as IoT gateways or Cloud services hosting the IoT data. As future work, we plan to demonstrate the performance gains of the proposed technique with a more elaborated sensor network setting composed of multiple sensors and data users to measure the performance of ABE on such setting. We plan also to consider other ABE schemes and extensions proposed in the literature.

References

[1] J. Bethencourt, A. Sahai, and B. Waters, Ciphertext-Policy Attribute-Based Encryption, In IEEE SP 07, 2007.

[2] B. Waters, Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization, In 14th PKC, 2011.

[3] S. Agrawal and M. Chase, FAME: Fast Attribute-based Message Encryption, In ACM CCS, 2017.

[4] N. Oualha and K. T. Nguyen, Lightweight Attribute-based Encryption for the Internet of Things, In 25th ICCCN 2016, 2016.

[5] A. Lewko and B. Waters, Decentralizing attribute-based encryption, In 30th EUROCRYPT, 2011.

[6] M. Ambrosin, M. Conti, and T. Dargahi, On the Feasibility of Attribute-Based Encryption on Smartphone Devices, In Workshop IoT-Sys, 2015.

[7] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, Performance evaluation of attribute-based encryption: Toward data privacy in the IoT, In ICC'14, 2014.

[8] J. Borgh, E. Ngai, B. Ohlman, and A. M. Malik, Employing Attribute-Based Encryption in Systems with Resource Constrained Devices in an Information-Centric Networking Context, In GIoTS, 2017.

[9] V. Nikov, S. Nikova, New monotone span Program from Old, Cryrtology ePrint Archive:Report, 2004/282.

[10] J. Xu and X. Zha, Secret Sharing Schemes with General Access Structure Based on MSPs, Journal of Communications, Volume 2, No. 1, 2007.

[11] Oded Goldreich, Foundations of cryptography: volume 2, basic applications, Cambridge university press, 2009.

[12] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence, AAA Authorization Framework, IETF RFC 2904, August 2000.

[13] Zolertia: https://github.com/Zolertia/Resources/wiki/RE-Mote

[14] A. Dunkels, B. Gronvall, and T. Voigt, Contiki-A Lightweight and Flexible Operating System for Tiny Networked Sensors, In LCN, 2004.

[15] J. A. Akinyele, Charm: A framework for rapidly prototyping cryptosystems, J. Cryptograph. Eng., vol. 3, no. 2, pp. 111-128, 2013.

[16] Relic-toolkit, an efficient library for cryptography, https://code.google.com/p/relic-toolkit/.

[17] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, Software-based on-line energy estimation for sensor nodes, In workshop EmNets'07, 2007.

[18] J. Crampton and A. Pinto, Attribute-Based Encryption for Access Control Using Elementary Operations, In IEEE CSF '14, 2014.

[19] OpenMote platform: http://openmote.com/

[20] Z. Shelby, K. Hartke, and C. Bormann, The Constrained Application Protocol (CoAP), IETF RFC 7252, June 2014.

[21] G. Tanganelli, C. Vallati, and E. Mingozzi, CoAPthon: Easy development of CoAP-based IoT applications with Python, In WF-IoT15, 2015.

[22] K. T. Nguyen, N. Oualha and M. Laurent, Lightweight Certificateless and Provably-Secure Signcryptosystem for the Internet of Things, In IEEE Trustcom/BigDataSE/ISPA, 2015.