Research Article

Smart offloading technique for CP-ABE encryption schemes in constrained devices

Mohammad Bany Taha[1] · **Hakima Ould-Slimane**[1] · **Chamseddine Talhi**[1]

Received: 27 May 2019 / Accepted: 18 January 2020 / Published online: 25 January 2020

Abstract
Nowadays, we are attending to the wide proliferation of IoT devices in various computing environments. Actually, this trend is the natural way to extend the M2M communication. However, these devices are facing serious resource constraints. To mitigate this problem, the huge amount of the exchanged data is collected from these devices for being stored and processed by the Cloud. Furthermore, since the Cloud is honest but curious, it may reveal personal information about users' habits owning these devices and can even lead to user profiling. Consequently, security mechanisms should be deployed at different levels to preserve data privacy so that only authorized users can gain access to the smallest piece of data according to the collection purpose. By assuming that the Cloud-is-honest-but-curious, we can not provide full or similar access to different untrusted Cloud services. Therefore, we should define the relevant privacy level implementing the data access control for each Cloud service according to some criteria. In this context, CP-ABE is a promising solution addressing this problem. This novel attribute-based public key encryption system provides a flexible fine-grained access control to data for any data requestor. However, performing all the related cryptographic operations on such devices is practically infeasible because of the resource constraints. For alleviating all the computation burden on these resource-limited devices, several schemes have been proposed. In this work, we propose a smart offloading technique that switches dynamically from full encryption to partial encryption according to a wise decision strategy considering the available resources and some crucial parameters like the number of attributes and the size of the data being encrypted. The relevant decision is based on a machine learning algorithm. To the best of our knowledge, this is the first paper proposing an adaptive CP-ABE scheme for constrained device optimizing the overall available resources.

Keywords CP-ABE · Offloading · IoT · Machine learning

1 Introduction

Recently, the market of IoT devices has known a spectacular boom. The number of these devices is expected to grow to 10 billion by 2020 and 22 billion by 2025. In fact, the emergence of this trend is mainly due to the evolution of wireless technology combined with Internet enhancement and the large demand for high-quality sensing devices enabling smart living applications. From a technological point of view, IoT can be seen as a consequence of machine-to-machine (M2M) architecture and its connectivity [1]. This key concept refers to the interconnection of machines via a network without any human intervention. As an extension of this technology, IoT is composed of billions of smart devices over a network connecting physical systems, people and smart applications intended to collect, process and share data. Furthermore, the flexibility and mobility of IoT devices are giving users

✉ Mohammad Bany Taha, an35670@ens.etsmtl.ca; Hakima Ould-Slimane, cc-hakima.ould-slimane@etsmtl.ca; Chamseddine Talhi, chamseddine.talhi@etsmtl.ca | [1]Department of Software Engineering and IT, École de Technologie Supérieure, University of Quebec, Quebec, Canada.

SN Applied Sciences (2020) 2:274 | https://doi.org/10.1007/s42452-020-2074-z

an easy and intuitive way to control their overall smart environment [2]. However, these devices are suffering from severe constraints on their basic resources related to processing, storage, and energy, which is rising serious performance issues.

In addition, the IoT devices are pervasively collecting and processing a huge amount of data leading to a critical capacity bottleneck at any of the resources involved. Fortunately, the Cloud constitutes the best alternative for solving this resource limitation problem. In this context, data owners will often outsource the collected data to take advantage from the huge storage and processing capacity of the Cloud. Usually, the Cloud is honest but curious [3], hence data owners can not fully trust any Cloud server. Indeed, since the data is outside of the controlled trusted area, any untrusted party (including the service providers) can potentially access to any sensitive data without the data owner's consent. Moreover, this uncontrolled access may reveal personal information about users' habits and can even lead to user profiling and privacy violation issues. Consequently, a strong security mechanism should be deployed at different levels to preserve data privacy and confidentiality to ensure that only authorized entities can gain access to the smallest piece of data according to the collection purpose and the data owner requirements.

To solve this problem, data owner should either use an authenticated access control system that allows only authorized users to access the data, or encrypt the data before being outsourced to the Cloud. However, using an authenticated access control system is not completely secure because intruders could still access the data using malicious software [4] or may tamper with the data [5, 6]. Therefore, preserving data privacy by adopting a strong encrypting mechanism is certainly more effective in such environment [7, 8].

There are two basic classes of encryption: symmetric mechanisms based on secret keys and asymmetric mechanisms based on public keys. Symmetric encryption is lighter than asymmetric encryption in terms of computation time [9]. On one hand, in symmetric encryption, the data owner should precisely know the identity of who requests the data to send him the right shared secret key needed for decryption. In IoT environments, this constraint is not feasible. In fact, once the data owner outsources his encrypted data to the Cloud, he has no information about the other users' identities or requests, so he is unable to provide any cryptographic material for any communication. On the other hand, in traditional asymmetric encryption, the data owner encrypts the data using a public key while the requester will use his own private key to decrypt the data. In this cryptosystem, the data owner should also identify who requests the data to correctly encrypt his data. Hence, traditional public-key systems are also not suitable to implement an effective encryption mechanism for IoT environments. Consequently, there is a need for an encryption system that can effectively handle access control in such environments. As a solution, a novel encryption mechanism called Attribute-Based Encryption (ABE) has been proposed [10]. This cryptosystem can specify then one-to-many encryption requirement regardless of the identity of the decryptors which is an inherent feature of a Cloud-based IoT environments. This mechanism is based on contextual information and identity attributes.

Attribute-Based Encryption (ABE) was first proposed in 2006 by Goyal et al. [11]. An attribute is a descriptive string attached to a user who may be characterized by multiple attributes. Hence, we can easily specify any group of users by a well-defined set of describing attributes. This concept fits very well with IoT environments since we can combine many attributes using logical operators to formulate a one-to-many access policy. The authors proposed a new form of asymmetric encryption called Key-Policy Attribute Based Encryption (KP-ABE). Later, in 2007, Bethencourt et al. [10] proposed a new type of Attribute Based encryption called Ciphertext policy Attribute Based Encryption (CP- ABE). In KP-ABE, the ciphertext is described with a set of attributes while the private keys are associated with an access structure specifying which ciphertext the users can decrypt. As a dual approach CP-ABE assigns attributes to private keys and attaches an access policy to the ciphertext so only users holding the set of attributes satisfying the access policy can decrypt the ciphertext.

The main advantage provided by ABE is the possibility of specifying flexible and expressive fine-grained access control policies over encrypted data. This feature satisfies the data minimization principle, a very important privacy requirement. In addition, this encryption system does not put any restriction on neither the number of authorized entities nor their identities. This crucial feature enables a reliable anonymous access control [12]. However, all the ABE mechanisms are still infeasible since the required encryption operations will lead to a heavy computation burden requiring high resources in terms of CPU, Memory, and energy consumption, which is a major issue for preserving privacy in such resource-constrained environments. Indeed, this infeasibility motivates the researchers to design variants of CP-ABE schemes that reduces the cryptographic computation burden and resource consumption to fit with the IoT environment constraints [13–17].

However, these previous contributions suffer from two major weaknesses. First, some of the proposed schemes are restricted to specific types of access policy. Thus, these schemes will be restricted to a limited range of applications. Secondly, all of these proposed schemes consider that the availability of resources on constrained devices is

constant all the time which is not realistic. Indeed, in practice, the availability of the resources is a variable parameter depending on: (1) the number of tasks and loads assigned to each device and, (2) the data size and the complexity of the access policy. For this purpose, we investigate the impact of this changing availability on the CP-ABE scheme performance in an IoT environment.

In this article, we propose a smart offloading approach that switches dynamically from full encryption to partial encryption according to a wise resource-based decision. More precisely, the proposed algorithm implements an adaptive CP-ABE scheme for constrained devices optimizing the overall available resources. We adopted a machine learning technique to select the appropriate encryption technique according to the resources availability, the complexity of access policy, and the data size. Finally, we validate the performance of our scheme in terms of execution time, CPU and memory utilization, and power consumption through several scenarios.

The rest of this paper discuss the related work in Sect. 2. The motivations of proposing adaptive CP-ABE scheme and the its challenges are discussed in Sect. 4. Then we present our proposed scheme in Sect. 6. Our results show in Sect. 7 followed by the conclusion in Sect. 9.

2 Related work

In this section, we discuss the main encryption schemes and the existing solutions to reduce computation cost under constrained devices.

In 2007, Bethencourt et al. [10] proposed the first CP-ABE scheme. The encryption algorithm for this scheme performs atomically all cryptographic operations to generate the final ciphertext. The evaluation results of this scheme show that the intensive required computations consume high resources (CPU, Memory, and energy). The researchers tried to find a way to reduce the overhead due to the encryption algorithm of CP-ABE as well as the execution time of the algorithm. Zhou et al. [18] proposed an efficient extension of CP-ABE scheme that securely outsources most of encryption and decryption operations to the Cloud without revealing data content and secret keys. They built their contribution by working on the access tree structure. Indeed, the size of the access policy is among the factors that significantly affect the computation complexity of CP-ABE. Since each access policy consists of a left sub-tree and a right sub-tree, Zhou et al. suppose that the left sub-tree of the access policy has more attributes than the right sub-tree. Accordingly, the users can encrypt their data with the right sub-tree of the access policy in order to generate the initial form of ciphertext CT_1. However, the efficiency of this work in terms of computation, communication, and storage is only possible by assuming that the root node of the access tree is always an "AND" gate, otherwise the scheme will not work.

Jin et al. improved Zhou et al. [19] work by proposing a flexible and lightweight CP-ABE scheme on mobile devices. The restriction that arises from Zhou et al. scheme is fixed in Jin et al. scheme by adding a dummy attribute to the right sub-tree of the whole access policy which provides more expressive access policies. To reduce the computation overheads at mobile client and preserve the data privacy, Jin et al scheme delegates most of the intensive ABE operations to Mobile Cloud Computing (MCC) and guarantees that neither the Encryption Service Provider (EPS) nor the Cloud provider which hosting the data can reveal that data. In Jin et al scheme, the user first encrypts their data with the right sub-tree of the access policy which contains only a dummy attribute to generate CT_{Dum} then uploads it to EPS. EPS generates CT_{Acc} considering the right sub-tree and combines it with CT_{Dum} to generate the final ciphertext CT ($CT = CT_{Acc} \wedge CT_{Dum}$).

Wang et al [20] proposed a verified outsourcing ABE scheme for key generation, encryption and decryption operations. The user encrypts the data partially to generate EP_O and EP_L, EP_L never leaves the user device and EP_O is sent to EPS. EPS performs more operations which entails more computation cost. The scheme successfully reduces the execution time. However, it generates high communication costs between user and client machines.

Zhao et al. proposed a scheme similar to Wang et al. works [21]. Zhao et al. reduce the overhead of CP-ABE on user device by encrypting the message using symmetric encryption then the data owner uses ABE to encrypt the symmetric key with attributes and sends CT_1 with C_{SE} to EPS. Same technique as in [20] is applied for the remaining part of the algorithms.

Touati et al. [22] proposed a lightweight CP-ABE scheme. The scheme assumes that there is constrained and unconstrained nodes (assisting nodes) in IoT environments. In summary, the scheme performs CP-ABE encryption by delegating these operations to assisting nodes. The authors take advantage of heterogeneous nodes in the environment to distribute the expensive computation of CP-ABE operations. However, the authors assume that all these nodes are trusted. Moreover, sending and receiving parameters between the user node and assisting nodes increase the communication overhead on user node and cannot be neglected. Nguyen et al. [23] proposed CP-ABE scheme to outsource ABE cryptography operations. The user device performs only one exponentiation to generate the initial CT_1, where Delegee component (DG) performs most of ABE expensive operations. DG is responsible of encrypting the data with access policy. Then, if DG is compromised

and the attacker changes the access policy to satisfy a client's secret key (SK), the client will be able to decrypt the data since there is no relation between C' and the access policy that the user sends to DG.

The overhead cost and the time of encryption algorithm in CP-ABE schemes is related to number of exponentiation that user devices perform [22]. Accordingly, we compare the time and the cost (CPU, Memory, and power consumption) of several CP-ABE schemes that were proposed for IoT devices in Table 1. The results shown on the table are related to a file whose size is 500 MB and 30 attributes defining the access policy. Table 1 shows that Touati et al. scheme [22] performs 0 exponentiation in user device. This means that this scheme is the fastest and less resources consumption. Next, Nguyen et al. scheme [23] needs only one exponentiation to generate the ciphertext. In [18] work, the number of exponentiations is based on the number of attributes in the right sub-tree.

Number of exponentiations in scheme [18] is $2|\tau_r| + 2$, where $|\tau_r|$ is the length of attributes in the right access policy. The number of exponentiations in Bethencourt's scheme [10] is dependent on the number of attributes in the access policy.

Based on our literature review, we summarize our encryption scheme requirements as the following:

- Correctness: The scheme should allow only the authorized user to decrypt the data.
- Scalability: The scheme should be able to handle a wide range of applications and and access policies.
- Feasibility: The scheme should be usable on constrained devices.
- Flexibility: The scheme should take into consideration the context of encryption information such as, number of attributes, data size. Then, based on this information the scheme should have the ability to select the appropriate technique (whether perform full encryption in user device or delegate most of cryptography operations to another device).

3 Preliminaries

In this section, we present the technical terminology that we use in this article.

3.1 Bilinear map

Most of the CP-ABE schemes are based on a bilinear map. Assume $\mathbb{G}_0, \mathbb{G}_1$ are two multiplicative groups of prime order p. Assume g is a generator of $\mathbb{G}_0$ and e is a bilinear map, e : $\mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. The properties of a bilinear map are:

- Bilinear: for all a,b *in* $\mathbb{G}_0$ and c,d $\in \mathbb{Z}_p$, $(a^c, b^d) = (a, b)^{cd}$
- Non-degenerate: e(g,g) $\neq$1

$\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map e : $\mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ are both efficiently computable. Map e is also symmetric: $e(g^a, g^b) = e(g, g))^{ab} = e(g^b, g^a)$

3.2 CP-ABE

In this scheme, the access policy is embedded in the ciphertext (i.e., the encrypted data) and private keys are generated according to a set of attributes. To decrypt the ciphertext, the user should own the private key related to a set of attributes satisfying the access policy. The original form of CP-ABE [10] consists of four algorithms:

- Setup (λ)$\rightarrow$ (PK, MSK). The algorithm uses security parameters (λ) to generate a public key (PK) and master secret key (MSK).

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

  Equation 1 is the Public Key (PK) equation. The Master Secret Key (MSK) is (β, g^{α}). Where α and β are random exponents ($\alpha, \beta \in \mathbb{Z}_p$). $\mathbb{G}_0$ is a bilinear group of prime order p with generator (g).
- KeyGen (PK, τ, MSK) $\rightarrow$ SK. The algorithm uses the public key, τ, MSK as input and generates the secret key

Table 1 Number of cryptography exponentiations needs to perform in user deviceß

Scheme	*NumberofExp*	Time (s)	CPU (MiB)	Memory (MiB)	Power consumption (*J*)
First CP-ABE Scheme [10]	$\|\tau\| + 2$	4.79	4.7	6.2	0.87
PP-CP-ABE [18]	$\|\tau_r\| + 2$	4.82	5.1	6.4	0.91
SL-CP-ABE [19]	2	0.332	3.6	2.7	0.424
Verifiable-Outsourced-CP-ABE [20]	4	0.83	4.1	4.8	0.81
VOC-CP-ABE [21]	4	0.79	4.2	5.1	0.781
OEABE [23]	1	0.24	2.1	2.6	0.39
C-CP-ABE [22]	0	0.189	1.9	2.6	0.388

(SK). The SK contains D, D_j, and D'_j components. τ is the client's attributes list.

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

The SK algorithm selects r and r_j random (r and $r_j \in \mathbb{Z}_p$) for each attribute $j \in \tau$. H is a hash function.

- Encryption (PK, M, $\mathbb{A}$)→CT, where CT is the ciphertext, $\mathbb{A}$ is the user's access policy, and M is the message that the data owner wants to encrypt.

$$\begin{aligned} CT =& \left(\mathbb{A}, C' = Me(g,g)^{\alpha s}, C = h^s, \forall y \in Y :\right. \\ C_y =& g^{\epsilon_y}, \left. C_{yp} = H(att(y))^{\epsilon_y}\right) \end{aligned} \tag{3}$$

In Eq. 3, the encryption algorithm generates random value (s) to calculate the shared value (ϵ_y) for each attribute in the access policy ($\mathbb{A}$) using linear secret sharing. Blinding each attribute in $\mathbb{A}$ with their share (ϵ_y) is preventing a collusion attack [10].

- Decryption(CT,SK)→ M. The decryption algorithm decrypts the ciphertext (CT) using (SK) to output the message (M).

$$M = \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \tag{4}$$

Definition 1 Assume that node Z is a leaf node and let j = att(z), The decryption algorithm works as the following :

$$\begin{aligned} Decrypt_Z =& \frac{e(D_j, C_x)}{e(D'_j, C'_x)} \\ =& \frac{e(g^r \cdot H(j)^{r_j}, h^{\epsilon_Z})}{e(g^{r_j}, H(j)^{\epsilon_Z})} \\ =& e(g,g)^{r\epsilon_Z} \end{aligned} \tag{5}$$

These same steps are repeated for $j \in \tau$. If the attributes that the SK blinds with satisfy the policy $\mathbb{A}$, then the algorithm will be able to decrypt CT. Otherwise, the algorithm will return $\perp$.

3.3 Access tree

The access tree is used to describe the access policy. The access tree consists of a set of nodes. The top node is called the root node whereas the inner nodes are either the logical operator (AND, OR, or OF) or leaf node. The leaf node represents the attributes and it is usually the lower level of the tree. Figure 1 shows samples of the access tree. The left tree in Fig. 1 is the original access tree (τ) where the policy is $\mathbb{A} = \left((A\ AND\ C)\ OR\ (B AND\ Z)\right)$. $\mathbb{A}$ is the access policy, A, C, B, and Z are the attributes that the CT encrypts with. The right access tree τ_D is the same access tree on the left of Fig. 1 but with an extra dummy attribute $\mathbb{A}_{\mathbb{D}} = \left(((A\ AND\ C\ )\ OR\ (B AND\ Z))ANDDummy\right)$ where $\mathbb{A}_{\mathbb{D}}$ is the access policy with an extra dummy attribute. A dummy attribute has the same features as any other attributes and it might be owned by any user. The user who want to decrypt the data must have this attribute. To prevent a collision attack, each attribute blinds with a secret share ϵ in the encryption algorithm [24]. We will further discuss the encryption algorithm in Sect. 6.

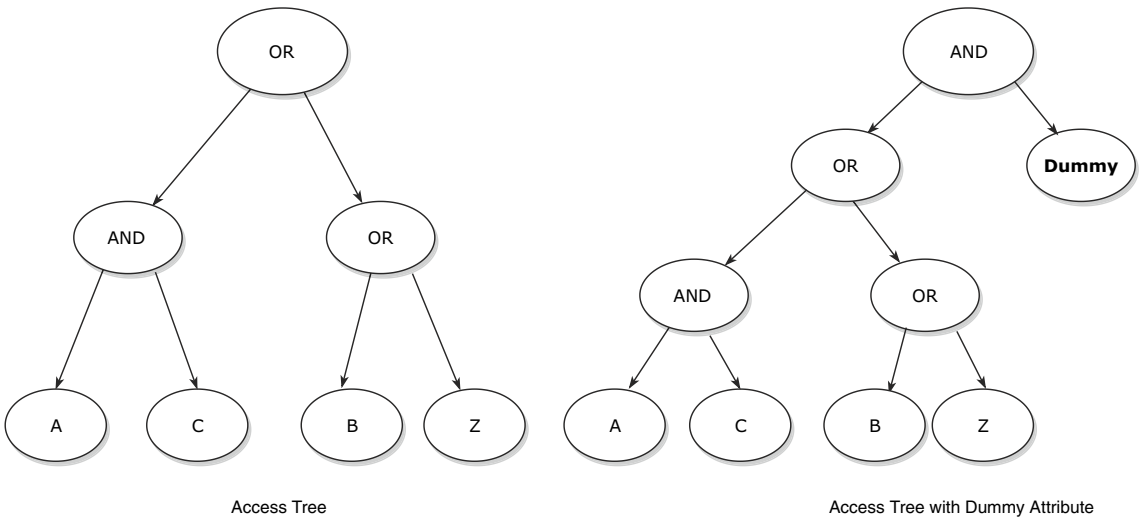

Fig. 1 Access tree τ (left side), Access tree with dummy attribute τ_D (right side)

4 Full versus partial encryption: incentive for adaptive scheme

The majority of related work shows that the ABE computation is too heavy to be performed on constrained devices and thus, prior contributions were focusing on delegating all or most of ABE encryption/ decryption tasks to a remote/proxy machine. An efficient solution was proposed by Jin et al. [19] where partial, on-device CP-ABE encryption is performed using a one attribute (dummy attribute) policy and remote CP-ABE encryption is performed using the complete policy. However, recent revolution in hardware as well as software systems dedicated to constrained devices is enabling on-device execution of all ABE tasks. However, the efficient management of constrained devices should take into consideration the variation of resources available (e.g., CPU, Memory, and battery). In other words, it will be preferable to perform all ABE tasks on the constrained device (full encryption) in some situations and in other situations, it will be better to perform only a few tasks (encryption based on the dummy attribute, called partial encryption) and offload the remaining tasks (encryption based on the original policy) to a remote server/proxy. We have investigated this aspect with the first experiment comparing the execution time, CPU utilization, and the power consumption of full and partial CP-ABE encryption respectively by varying the attributes number from 2 to 500.

Our results show that the full encryption scheme [10] is faster than the partial encryption scheme [19] as shown in Fig. 2. In the full encryption scheme, the total time is the time needed to perform CP-ABE operations before uploading CT to the Cloud. In partial encryption, the total time is the time of CP-ABE operations in the user device and proxy machine as well as the transmission time between two machines (part A and B of Fig. 4 show the full and partial encryption schemes respectively). However, our results show that full encryption consumes more CPU than partial encryption as shown in Fig. 3.

Based on the aforementioned results, we can identify an incentive to perform on-device full encryption when the available CPU and battery are sufficient and perform partial encryption when the available CPU and battery are less than the required budget. This rule-based decision making can be extended if more encryption parameters and performance metrics are investigated. In fact, the size of messages to be encrypted, their type, and their frequency should have an impact on the CPU utilization, power consumption, and execution time needed for their encryption. The available Memory and the quality of "WIFI/ mobile" connections are important performance metrics that should be studied.

Assuming such a decision-making process is available, the challenge arises when it comes to design an adaptable scheme that is capable of automatically switching from full to partial CP-ABE encryption according to the context. Such a scheme should have the ability to generate a secret key that can decrypt the data regardless of which encryption technique (full or partial) is used to encrypt the data. More specifically, using Bethencourt's scheme [10] to perform full encryption and Jin's et al. [19] to perform partial encryption require two secret keys for each client to decrypt the same file. The first secret key will be used

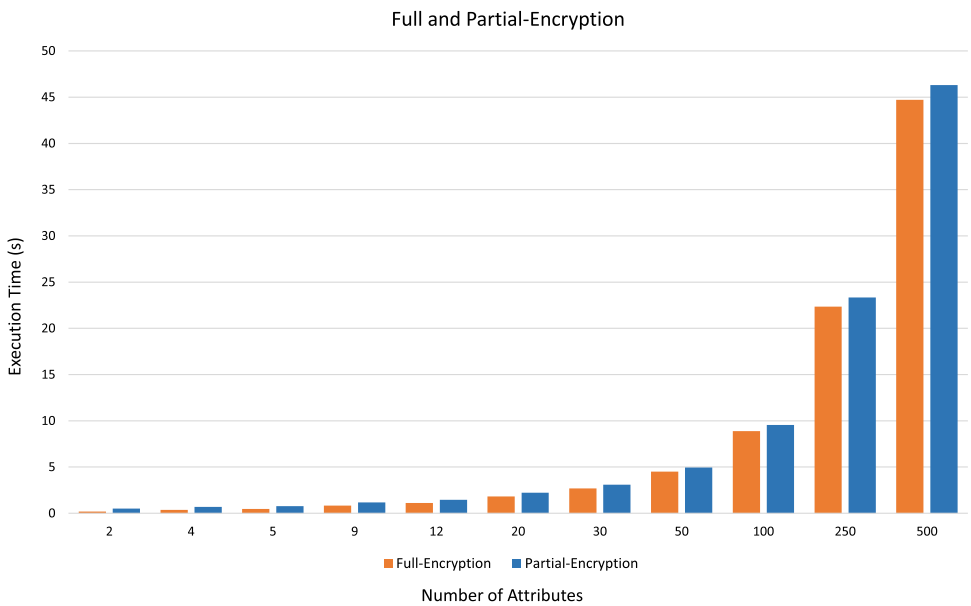

Fig. 2 Total time of full and partial encryption scheme

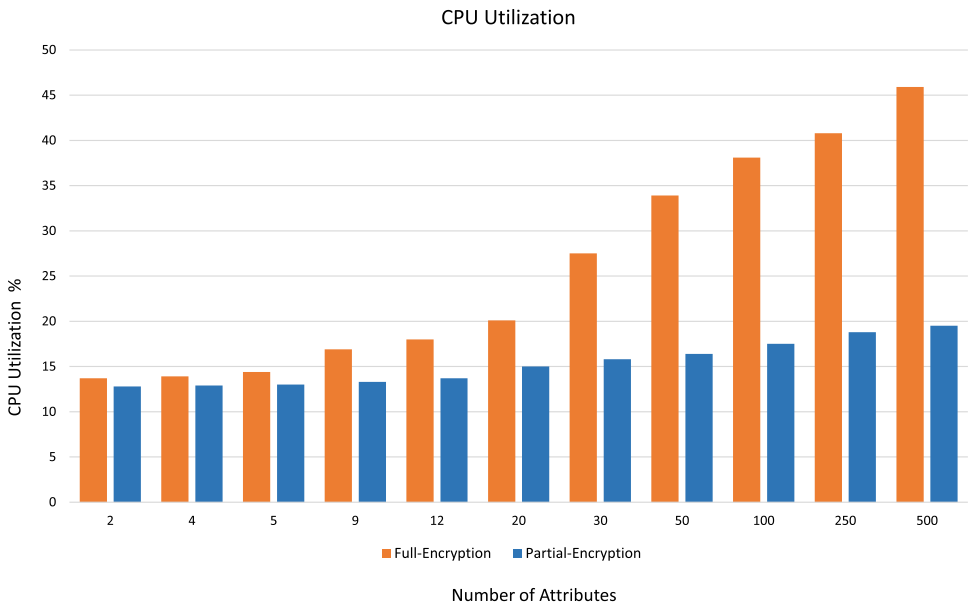

Fig. 3 CPU utilization of full and partial encryption scheme

when full encryption is applied while the second key will be used to decrypt data encrypted using a partial scheme. More precisely, in Jin's scheme, the data is first encrypted with a dummy attribute before being encrypted by the access control policy. This is not supported in Bethencourt's scheme [10].

5 Mathematical model

In this section, we will discuss the mathematical specification of an adaptive problem and we will find the formal definition of performing full and partial (local computation and offloading respectively) CP- ABE operations on a constrained device. Finally, we will discuss the optimization issue of in adaptive solution. Our variable notations are shown in Table 2.

5.1 Decision variable

In our article, the decision variable notation is x_{D_i}, where x_{D_i} is:

$$x_{D^i} = \begin{cases} 0 & \text{Full Encryption} \\ 1 & \text{Partial Encryption} \end{cases}$$

5.2 Assmptions

Let T^i_{ct} be the total time required to generate the ciphertext CT (the encrypted data generated by each device) for task i.

Table 2 Notations

Parameter	
$\mathbb{A}$	Access policy
$\mathbb{A}_{\mathbb{D}}$	Access policy with dummy attribute
τ	Real access Tree
τ_D	Access tree with dummy attribute
CT^i	Ciphertext of task i
CT^i_1	Partial Ciphertext of task i
f^i_{ct}	CP-ABE function that generate CT for task i
$f^i_{ct_l}$	CP-ABE function in local device to generate CT of task i
$f^i_{ct_r}$	CP-ABE function in remote device to generate CT of task i
U^i_l	CPU utilization of local device for task i
U^i_r	CPU utilization of proxy machine for task i
RAM^i_l	Memory utilization of local device for task i
RAM^i_r	Memory utilization of proxy machine for task i
P^i_l	Power consumption of local device for task i
P^i_r	Power consumption of proxy machine for task i
T^i_{ct}	Total time required to generate CT for task i
T^i_{fct}	Execution time required to perform CP-ABE
T^i_t	Transmission time for task i
T^i_l	Time required to perform CP-ABE locally
T^i_r	Time required to perform CP-ABE remotely
T^i_{prop}	Propagation delay for task i
lx	Connection type between constraint device and the proxy
T^i_{trans}	Transmission delay for task i
x^i_D	Decision parameter whether local or offload operation i

Assumption 1 Equation (6) shows that T^i_{ct} is dependent on $T^i_{f_{ct}}$ and T^i_t, where $T^i_{f_{ct}}$ is the time required to perform cryp-

tography operations to generate the CT for task i. T_t^i is the time required to either send the CT directly to the Cloud in case T_{ct}^i performed a full encryption or the time needed to send the CT_1 to the proxy machine and send CT to the Cloud.

Assumption 2 We find T_t^i only in case CP-ABE is performed partially since we assume that the CP-ABE performed (full or partial) will take the same time to upload CT to the Cloud. Hence, we calculate T_t^i only if CP-ABE performs a partial encryption because it is the time to send CT_1 from RP1 to RP2 before uploading the final CT to the Cloud.

$$T_{ct} = \sum_{i=1}^{I} (T_{f_{ct}}^i + T_t^i) \tag{6}$$

where $T_{f_{ct}}^i$ is the execution time required to perform CP-ABE, and T_t^i is the transmission time. $T_{f_{ct}}^i$ depends on the available resources in the constrainted device in terms of CPU and memory utilization as shown in Eq. (7).

$$\begin{aligned} T_{f_{ct}} = & \sum_{i=1}^{I} (1 - x_D^i)(f_{\mathrm{ct}I}{}^i \times U_l^i \times RAM_l^i \times P_l^i) \times t_l^i \\ & + \sum_{i=1}^{I} x_D^i (f_{\mathrm{ct}I}{}^i \times U_l^i \times RAM_l^i \times P_l^i \\ & + f_{\mathrm{ct}r}{}^{i_r} \times U_r^i \times RAM_r^i \times P_r^i) \times t_r^i \end{aligned} \tag{7}$$

where f_{ct}^i is CP-ABE function that generate CT for task i. U_l^i is the percentage of CPU utilization on a constraint device for task i and RAM_l^i is the percentage of memory utilization on a constraint device for task i. P_l^i is the power consumption on a constraint device for task i. U_r^i and RAM_r^i are the percentage of CPU, memory utilization in the proxy machine respectively. P_r^i is the power consumption in proxy machine. T_l^i and T_r^i are the time needed to perform CP-ABE operations (either full or partial encryption respectively). x_D is the decision variable taken to perform CP-ABE (either full or partial).

In addition to $T_{f_{ct}}$, Eq. (6) shows that T_{ct}^i also depends on T_t^i. T_t^i consists of two terms shown in Eq. (8).

$$T_t = \sum_{i=1}^{I} \frac{2x_D}{x_D + 1} (T_{prop}^i + T_{trans}^i) \tag{8}$$

where T_{prop}^i is the propagation delay and T_{trans}^i is the transmission delay. Propagation delay is the time of propagate data from the beginning link of RP1 to the target vehicle RP2 (lx). The propagation delay depends on the media that the data transfer is on. Transmission delay (T_{trans}^i) is a fixed value and depends on the length a packet. T_{trans}^i is the time needed to push out CT_1 from RP1 into the link between RP1 and RP2 (lx).

Problem Definition. *Consider constraint device j performs multi CP-ABE tasks* $f_{ctI}, I = \{i_1, i_2, ..i_n\}$. *Each CP-ABE operation needs* T_{ct}^i *to generate* CT^i. T_{ct}^i *depends on* U_l^i *CPU utilization and* RAM_l^i *(memory) utilization, and* P_l^i *(battery) power consumption,* $T_{prop_l}^i$, *and* $T_{trans_l}^i$ *in case* f_{ct}^i *is performed locally (full encryption). On the other hand,* T_{ct}^i *depends on* U_l^i *CPU utilization,* RAM_l^i *memory utilization,* P_l^i *power consumption,* $T_{prop_lx}^i$, $T_{prop_r}^i$, $T_{trans_l}^i$, *and* $T_{trans_px}^i$ *in case* f_{ct}^i *selects (partial encryption) or offloading option*. Accordingly, all of these factors should be considered when deciding if f_{ct}^i should perform full or partial (delegating to proxy machine) encryption. In this article, we aim to reduce T_{ct} for all I. In constraint devices, this is considered a challenge for the following reasons:

We discussed in Sect. 4 that the relation between resources and T_{ct} is an inverse relation. More precisely, more resources (U^i, RAM^i, P^i) take less time to generate CT. In addition to the resource, the total time (T_{ct}^i) depends on whether f_{ct} will be performed locally or will be offloaded. Therefore, minimizing T_{ct} by taking into consideration the factors that can minimize T_{ct} and guaranteeing that generating CT is visible with maximum available resources is our objective.

Lemma 1 *Optimizing the total time* (T_{ct}^i) *of performing CP-ABE operations in constrained devices is NP-Hard.*

Proof We reduce the 0–1 knapsack problem to reduce the total time of generating CT (Eq. 9) since the binary partition is made on a serial task graph. Therefore, as 0–1 knapsack problem is NP-hard [25], deciding if the scheme should perform full or partial encryption to achieve Eq. 9 is NP-hard. Considering a simple scenario of our problem. The constraint device will perform only one CP-ABE operation to generate CT^i. The total time of performing CP-ABE using full encryption (locally in a constraint device) is T_{ct}^i, whereas it is $T_{ct}^{'i}$ for the same task (i) in case i is performed using partial encryption (offloading technique). Due to limitation of resources in constraint devices, taking the decision (x_D^i) of whether task i should be performed using full or partial encryption is a NP-hard. T_{ct}^i is depends on the execution time (T_{fct}^i) and the transmission time T_t^i. T_{ct}^i depends on U_i, RAM_i, and P_i of the constrained device and the assistant device (proxy server). T_{ct}^i also depends on $\mathbb{A}$ and the size of data. We take these factors into consideration to select the appropriate technique (full or partial encryption) that can generate CT^i with a minimum total time $\left(T_{ct}^i = \min \sum_{t=0}^{I} T_{ct}^i\right)$. □

5.3 Optimization formulation

To achieve our goal discussed above (minimum total time), we discuss the optimization formula in this section.

$$T_{ct}^{I} = \min\left(\sum_{i=1}^{I} T_{f_{ct}}^{i} + T_{t}^{i}\right) \tag{9}$$

$$\begin{aligned} T_{f_{ct}}^{I} = \min\Bigg(& \sum_{i=1}^{I} (1 - x_D{}^{i})(f_{ct_I}{}^{i} \times U_I^i \times RAM_I^i \times P_I^i) \times T_I^i \\ & + \sum_{i=1}^{I} x_D^i (f_{\mathrm{ct}_I}{}^{i} \times U_I^i \times RAM_I^i \times P_I^i \\ & + f_{\mathrm{ct}_{r_r}}{}^{i} \times U_r^i \times RAM_r^i \times P_r^i) \times T_r^i \Bigg) \end{aligned} \tag{10}$$

$$T_t^i = \min\left(\sum_{i=1}^{I} \frac{2x_D{}^{i}}{x_D{}^{i} + 1}(T_{prop}^i + T_{trans}^i)\right) \tag{11}$$

S.t

$i \in I \quad (c1)$

$x_D^i = \{0, 1\} \quad (c2)$

$10\% \leq U_c^i < 35\% \quad (c3)$

$35\% \leq U_m^i < 55\% \quad (c4)$

$55\% \leq U_i dl^i \leq 100\% \quad (c5)$

$10\% \leq RAM_c^i < 35\% \quad (c6)$

$35\% \leq RAM_m^i < 55\% \quad (c7)$

$55\% \leq RAM_i dl^i \leq 100\% \quad (c8)$

$T_{prop}^i = \{wire, wireless\} \quad (c9)$

The c1 constraint ensures that the number of tasks is natural. Constraint c2 to ensures that the decision variable x_D^i is a binary value based on the decision taken. c3, c4, and c5 ensure the status of CPU (idle, meduim, critical) of the machine that will perform task i. c6, c7, and c8 ensure the status of memory (idle, medium, critical) of the machine that will perform task i. c9 ensures that the connection media between the constraint device and the proxy machine is either wire or wireless.

In order to achieve our optimization objective, we use machine learning that considers all factors that may increase the T_{ct}. Then it makes the decision whether the machine should perform CP-ABE locally or it should offload CP-ABE. We will discuss this machine learning technique in Sect. 8 .

6 Adapted CP-ABE

In this section, we will discuss how to switch from the scheme proposed by Bethencourt et al. [10] and the scheme proposed by Jin et al. [19] without involving two different private keys per user. Both schemes are secure as discussed in Sect. 2. We will start with defining the main use case scenario. We then compare the schemes before proposing adaptive CP-ABE encryption.

6.1 Main use case

Figure 4 shows three schemes. Scheme (A) represents full encryption [10] where the data is totally encrypted in the device (the gateway RP1 in Fig. 4). Scheme (B) is the partial encryption scheme proposed by Jin et al. in [19]. Scheme (C) is our proposed scheme. We call it the adaptive encryption scheme. The adaptive scheme selects the appropriate encryption (full or partial) to encrypt the data based on the context of the policy, file size, and the availability of resources on the device (RP1).

The main job of the gateway (RP1) in all three schemes is to encrypt data (either full or partial encryption) before uploading it to the Cloud. In full encryption (Fig. 4), RP1 performs all CP-ABE operations and then uploads the CT

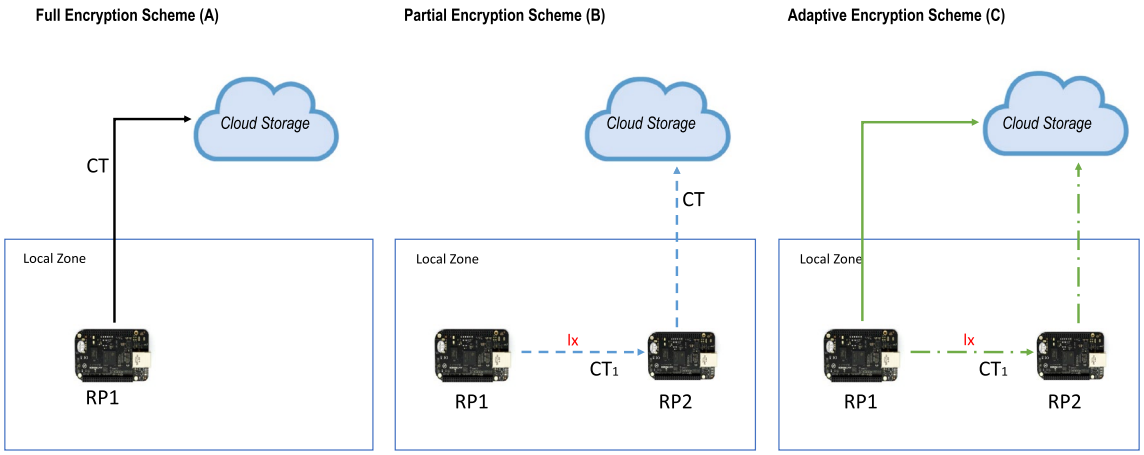

Fig. 4 Full (**a**), partial (**b**) and (**c**) adaptive Schemes

to the Cloud. In partial encryption, RP1 performs a few CP-ABE operations to generate intermediate CT1. Then, RP1 send CT1 and non-critical parameters to RP2 to perform the remain part of CP-ABE. Even though the scheme delegates most of the CP-ABE operations to RP2, the data cannot be revealed by attacker (such as Man in The Middle Attack) [19].

In this article, we assume that RP1 and RP2 are on the same network. They could communicate through LAN or WLAN. The performance results related to experimenting partial encryption show an important difference depending on whether RP1 and RP2 are connected through LAN or WLAN. This will be discussed later in Sect. 7.

6.2 Comparison of [10] and [19] encryption schemes

In a full encryption scheme, the data is encrypted completely in the gateway (RP1) (by encrypting the data with a policy and generating the final CT), then uploading it to the Cloud. Algorithm 1 shows the cryptography operations of full encryption that the gateway (RP1) performs before it uploads the CT to the Cloud. In Algorithm 1, the algorithm generates a key and security parameter in line 1 and 2 respectively. In line 3 of Algorithm 1, the sub-secret ϵ calculated for each node y in τ including leaf nodes. ϵ is a polynomial chosen for all nodes in the access tree from the top node (root node) until the last leaf node in a top down manner. The algorithm selects a random s that generated in line 2 of algorithm 1 and set $\epsilon_R = s$, where ϵ_R is the polynomial value of the root node. Then it moves down to the next node (y) and finds $\epsilon_y = q_{parent(y)}(index(y))$. It chooses d_y to define q_y where d_y is a degree of polynomial ϵ_y [10]. Lines 4 and 5 of Algorithm 1 show how to calculate the security parameters C, C′, C_y, and C_{yp}. Note that C_y, and C_{yp} in line 6 are calculated for each attribute (att(y)) in the access policy ($\mathbb{A}$).

Algorithm 1 Full Encryption in RP1

Input: PK, M, $\mathbb{A}$
Output: CT
1: Generate key (k) from group element GT
2: Select random element s from $\in$ ZR
3: Calculate secret shared ε_y , $\forall\ \tau$ in $\mathbb{A}$
4: Compute $C = h^s, C' = M \times e(g,g)^{\alpha s}$
5: Calculate $C_y = g^{\varepsilon_y}, C_{yp} = H(att(y))^{\varepsilon_y}, \forall \tau \in \mathbb{A}$
6: **return** $CT = \{C, C', \forall y \in Y : C_y, C_{yp}\}$

In a partial scheme (B of Fig. 4), data is encrypted with only the dummy attribute and most of the CP-ABE cryptography operations are delegated to another assistant device or a proxy server. Several schemes proposed this idea such as the scheme in [19]. Algorithm 2 shows the steps of partial encryption performed in the gateway, and the operations in step 1, 2, and 3 are the same as those performed in the full encryption scheme (Algorithm 1). However, in step 4 of Algorithm 2, the gateway only calculates the security parameters (C_{y_D} and C_{yp_D}) of τ_D where τ_D is an access tree with a dummy attribute as shown in Fig. 1.

Therefore, in partial encryption, the device takes less time and consumes less resources compared to encryption since the algorithm calculates C_{y_D} and C_{yp_D} only for one attribute (the dummy attribute) instead of calculating C_y and C_{yp} for all attributes τ in $\mathbb{A}$ as it is in full encryption. The assistant device completes the encryption process by performing most of the CP-ABE encryption operations (Algorithm 3). Indeed, it computes the security parameters C_y and C_{yp} for all attributes (τ_D) in the access tree $\mathbb{A}$ except the dummy attribute (C_{y_D}, C_{yp_D}). Algorithm 3 shows the steps of generating the second part of ciphertext. In Fig. 4, the assistant device is (RP2). In a full encryption scheme, there is no assistant device since it performs all CP-ABE operations on the user device (RP1).

Algorithm 2 Partial Encryption in RP1

Input: PK, M, $\mathbb{A}$
Output: CT_1
1: Generate key (k) from group element GT
2: Generate random element s from $\in$ ZR
3: Calculate secret shared ε , $\forall\ \tau$ in $\mathbb{A}$
4: Compute $C = h^s, C' = M \times e(g,g)^{\alpha s}$
5: Compute $C_{y_D} = g^{\varepsilon_y}, C_{yp_D} = H(att(y))^{\varepsilon_y}, \forall\ \tau_D \in \mathbb{A}$
6: Send CT_1 to RP2, where $CT_1 = \{C, C', C_{y_D}, C_{yp_D}\}$

Algorithm 3 CP-ABE Encryption in RP2

Input: PK, CT_1, $\mathbb{A}$
Output: CT
1: Compute $C_y = g^{\varepsilon_y}, C_{yp} = H(att(y))^{\varepsilon_y}, \forall\ \tau_D \in \mathbb{A}$
2: Compute $C_{yp} = H(att(y))^{\varepsilon_y}, \forall\ \tau_D \in \mathbb{A}$
3: Generate CT, where $CT = \{C, C', C_{y_D}, C_{yp_D}, \forall \tau_D \in \mathbb{A} : C_y, C_{yp}, \}$

6.3 Adaptive CP-ABE

We solve the second challenge that we have discussed in Sect. 4 by adding one attribute (dummy attribute) to any access policy in the full encryption algorithm. Thus, in our encryption algorithm (Algorithm 4), the client will be able to decrypt the data using one secret key regardless of whether full or partial encryption is performed on the gateway. Algorithm 4 shows the encryption algorithm in the constraint device (RP1). In our algorithm, the client is able to use his/her secret key to decrypt ciphertext whether the data is encrypted using a full or partial encryption scheme.

Adding one attribute to the access policy should increase the execution time and resource consumption. However, we found that the introduced overhead is negligible. In fact, we measured the execution time, CPU and memory utilization of our full encryption algorithm (Algorithm 4) with the full encryption algorithm (Algorithm 1) proposed in [10]. Figure 5 shows that the difference in execution time between the two algorithms is around 0.04 seconds. Moreover, the CPU and memory utilization of the two algorithms are almost equal.

Based on our experiments and discussion, offloading for CP-ABE operations is required in the constrained device to adapt the limitation of the resources problem. However, applying offloading technique on two different schemes ([10] and [19]) is not straightforward. Assume that the data owner uses the Bethencourt et al. scheme [10] to encrypt message M with policy $((B\}\}OR''C)\}\}AND''D)$ to generate CT′. And assuming the client want to decrypt CT′, and his/her SK is generated using Jin et al. [19], and Assume client's attributes is (B, D, and dummy attributes). Then, the client will not be able to decrypt CT′ for the following reason; The policy that CT′ blinds with is $\mathbb{A}$. Therefore, to decrypt CT′, all attributes in τ need to reconstruct the random values (line 2 of Algorithm 1) based on Eqs. 4 and 5 [10]. The client SK will have D_j and D'_j for all τ_D which will not satisfy $\mathbb{A}$, thus, random value (s) will not recover since $\tau \neq \tau_D$ [10]. Therefore, Eq. 4 will return $\perp$. In our Algorithm, we fixed this problem in line 13. We compute C_y and C_{yp} for all τ_D even if the decision is to perform full encryption. Therefore, the client's SK is able to decrypt CT because CT will have C_y and C_{yp} for all τ_D.

We find the security parameters that need to encrypt the message M as shown in line 2 and 3 of Algorithm 4. In line 4 to line 6, the algorithm calculates ϵ for each node in τ_D. After this, our algorithm checks the resources in the device that run in (RP1) and the resources in the proxy (RP2). This is shown in line 7 of Algorithm 4 where U_l and U_r are the CPU utilization in the local device (RP1) and proxy machine (RP2) respectively. RAM_l and RAM_r are the memory utilization in the local device (RP1) and proxy devices (RP2) respectively. *lx* in line 7 of Algorithm 4 is the media type of connection between RP1 and RP2. These variables in addition to the policy length that the data will encrypt with ($\mathbb{A}$) impact the decision of whether the algorithm will perform the encryption locally or by offloading the encryption. Figure 14 shows the decision tree that we use for performing CP-ABE operations locally or by offloading to the assistant device. We will further discuss the machine learning technique that we used in Sect. 8. Line 8 of Algorithm 4 shows that the algorithm takes the decision (x_D) whether it will perform the encryption locally or by offloading the task to the proxy. If the decision based on the context at the time of encryption is to perform full encryption (locally where $x_D = 0$), the Algorithm will perform the lines from 11 to 15. The Algorithm Computes C and C′ first, and C_y and C_{yp} for all τ_D. Finally, the algorithm uploads the CT to the Cloud in line 15 of the algorithm.

If the decision ($x_D = 1$) is to perform partial encryption, then the algorithm performs the steps from line 17–20 in the constrained device (RP1). The first two steps are the

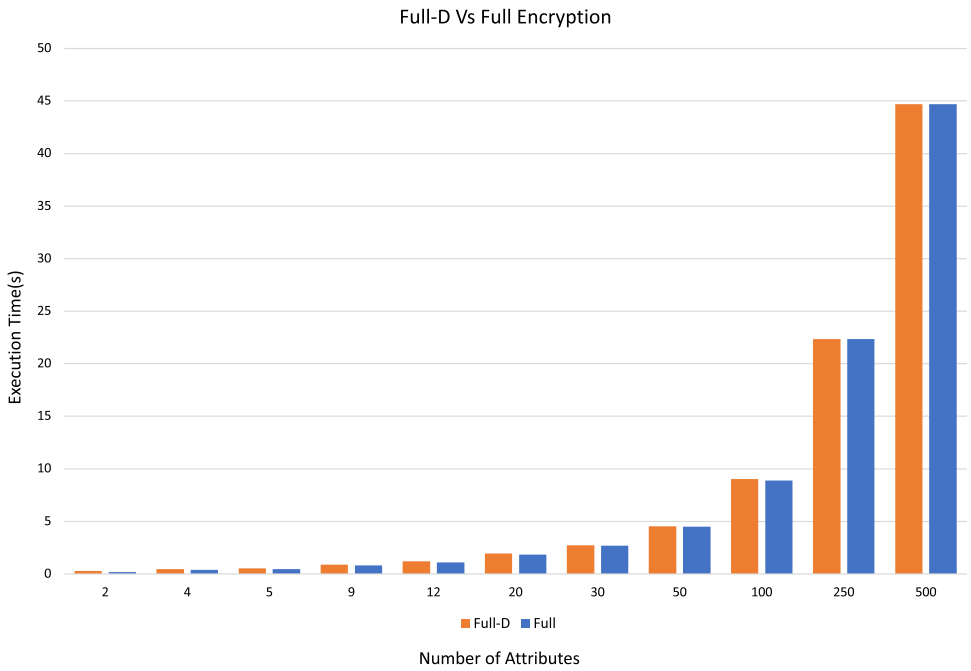

Fig. 5 Comparison between full-D and full encryption scheme

same as in full encryption where the algorithm computes C and C′ (line 17). However, unlike full encryption, the algorithm computes only C_{y_D} and C_{yp_D} for the dummy attribute (line 18). The remaining components will be computed in assistant device (RP2) and this will reduce the computation cost on the constraint device and the time complexity of the algorithm.

Algorithm 4 Our Algorithm (RP1)

Input: PK, M, $\mathbb{A}$
Output: CT || CT1
1: initialization:
2: Generate key (k) from group element GT
3: Select random element s from $\in$ ZR
4: **for each** $Node_i$ in τ_D **do**
5: Calculate secret shared ε , $\forall$ τ_D in $\mathbb{A}_{\mathbb{D}}$
6: **end for**
7: Find U_l, U_r, RAM_l, RAM_r,lx
8: Find x_D based on machine learning
9: End initialization
10: **if** $(x_D = 0)$ **then**
11: Compute $C = h^s$, C' = M $\times$ $e(g,g)^{\alpha s}$
12: **for each** $Node_y$ in τ_D **do**
13: Compute $C_y = g^{\varepsilon_y}$, $C_{yp} = H(att(y))^{\varepsilon_y}$
14: **end for**
15: Upload CT, where $CT = \{C, C', \forall \tau_D \in \mathbb{A} : C_y, C_{yp}, \}$
16: **else**
17: Compute $C = h^s$, C' = M $\times$ $e(g,g)^{\alpha s}$
18: Compute $C_{y_D} = g^{\varepsilon_{y_D}}$, $C_{yp_D} = H(att(D))^{\varepsilon_{yD}}$
19: Send CT_1 to (RP2). Where $CT_1 = \{C, C', C_{y_D}, C_{yp_D}, \}$
20: **end if**

Algorithm 5 Our Algorithm (RP2)

Input: PK, $\mathbb{A}$, CT_1
Output: CT
1: **for all** (att(y) in τ_D **do**
2: Compute $C_y = g^{\varepsilon_y}$, $C_{yp} = H(att(y))^{\varepsilon_y}$,$\forall$ $\tau_D \in \mathbb{A}$
3: Compute $C_{yp} = H(att(y))^{\varepsilon_y}$,$\forall$ $\tau_D \in \mathbb{A}$
4: **end for**
5: Generate CT, where $CT = \{C, C', C_{y_D}, C_{yp_D}, \forall \tau_D \in \mathbb{A} : C_y, C_{yp}, \}$
6: Upload CT to the Cloud

Algorithm 5 shows the steps that RP2 performs to generate the final CT of CP-ABE after it receives the CT_1 from RP1. RP2 computes $C_y$and C_{yp} respectively for all attributes in τ_D (Algorithm 5, line 2 and 3) except C_{y_D} and C_{yp_D} since these two components are computed in RP1. The algorithm combines CT_1 with the components that were computed to generate a final CT as shown in line 5. Finally, RP2 uploads CT to the Cloud.

Lemma 2 *The time complexity of our algorithm (Algorithm 4) in the data owner device* (*RP*1) *is* $\mathcal{O}(n)$.

Proof The time complexity of our algorithm (4) in the constraint device depends on the length of the access tree τ_D. In line 2 and 3, the Algorithm 4 generates a random number where the time complexity in these two operations is $\mathcal{O}(1)$. The algorithm in line 4 to line 6 calculates the polynomial ϵ for each node in τ_D. The time complexity for this part of the algorithm is $\mathcal{O}(n+1)$. The time complexity for "if condition" (line 10) is $\mathcal{O}(1)$. In case the number of nodes in τ_D is n, then the time complexity of the algorithm is $\mathcal{O}(n+1)$ to compute C_y and C_{yp} components for all nodes in τ_D. The time complexity for /'for'/ loop in the algorithm (line 18) in case the decision is $x_D = 1$is $\mathcal{O}(1)$, because the algorithm will compute C_{y_D} and C_{yp_D} for only the dummy attribute. The time complexity of the algorithm is $\mathcal{O}(n)$

□

7 Experimental results and analysis

In this section, we discuss the performance of a full-D and partial CP-ABE encryption in different scenarios. Table 3 shows the hardware specifications of the gateway (RP1) and the assistant device (RP2) that have been used in our experiments. We use the Charm framework to run several CP-ABE schemes [26]. We also use the TOP tool to measure the CPU and memory utilization, and we use USB power meter to measure the power consumption. As we discussed earlier, the execution time and CPU utilization of encryption algorithm increases whenever the number of attributes in the access policy increases. In addition to the number of attributes, we found that the size of the file also influences the execution time and CPU utilization in constrained devices. Figures 6 and 7 show the execution time of full-D and partial encryption applied on two files, one of size 1-byte and the other of size 500-MB. Each figure shows that large files need more time to perform encryption than small files. Also, large files consume more resources than small files as shown in Figs. 8 and 9. In terms of energy consumption, the results are proportional to those of CPU usage as it is expected, since the energy consumption for partial and full encryption is the one consumed their usage of the CPU. Figures 10 and 11 show the power consumption of partial and full-D encryption schemes respectively.

In this experiments, we assume that all files are stored locally in the gateway. This will help to get accurate results. Moreover, to conduct a fair comparison between full and partial encryption, we investigate several scenarios applied on full and partial encryption. As shown in Table 4, each scenario consists of several properties. For example, in S1, the file size is 1-byte, the number of attributes in the access policy is nine, the status of resources (*Avg* − *Res*) is medium (around 50% of the resources is available), and

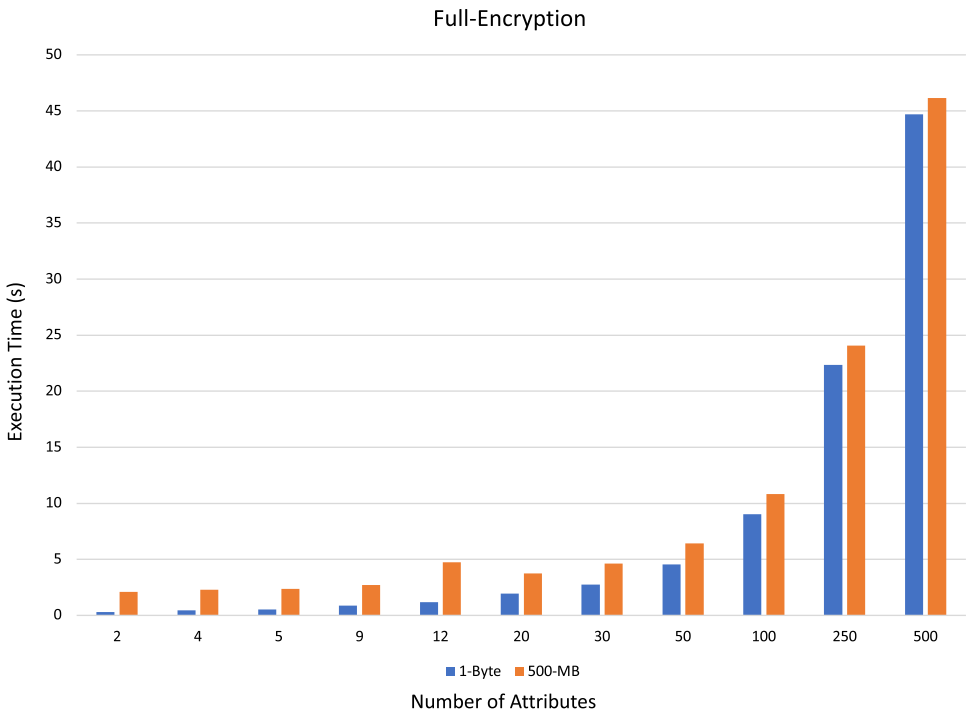

Fig. 6 Overall execution time of full-D encryption for the file size 1 Byte and 500 MB

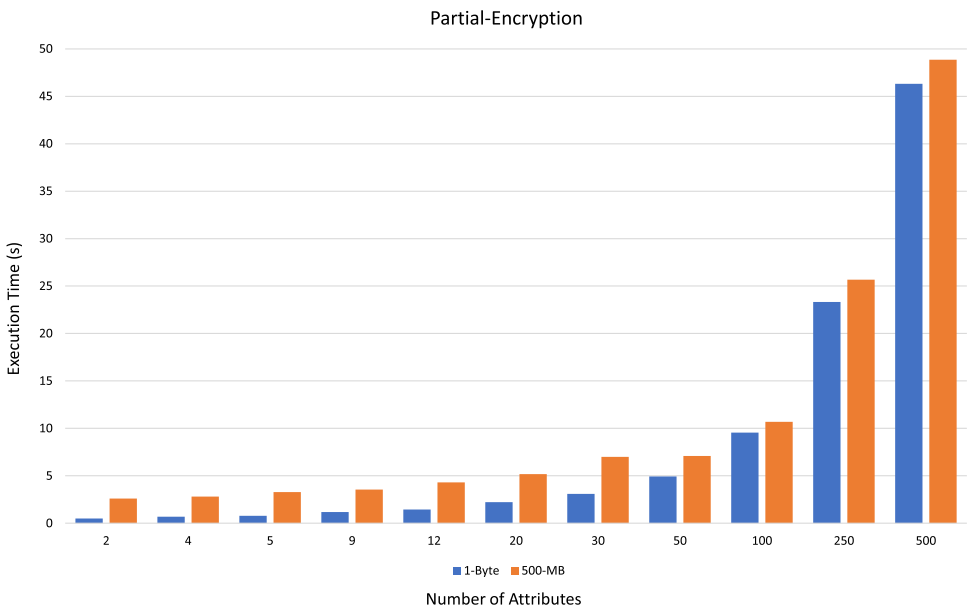

Fig. 7 Overall execution time of partial encryption for the file size 1 Byte and 500 MB

the connection between RP1 and RP2 is LAN. In Table 4, there are three Avg-Res statuses (Idle, Medium, and Critical). Idle status means only 20% of RP1 (CPU, memory, and battery) is reserved which means 80% of the CPU is available. Medium status means 50% of RP1 $Avg - Res$ is available. Finally, critical status means 20% of the $Avg - Res$ is available.

Figure 12 shows the total time of full and partial CP-ABE encryption in scenarios S1 to S15. We can observe interesting facts. In S1 and S2, there are two different file sizes, 1-byte in S1 and 1-KB in S2. Both files are encrypted with the same number of attributes (9) as shown in Table 4. In S1, the connection between RP1 and RP2 is LAN while it is WLAN in S2. The results in Fig. 12 show that full encryption is faster than partial encryption in S1 while partial encryption is faster than full encryption in S2. The fact that the file size in S2 is bigger than the file size in S1 can explain these results.

When we compare S3 and S4, we can see that the two scenarios have the same file size and these files are

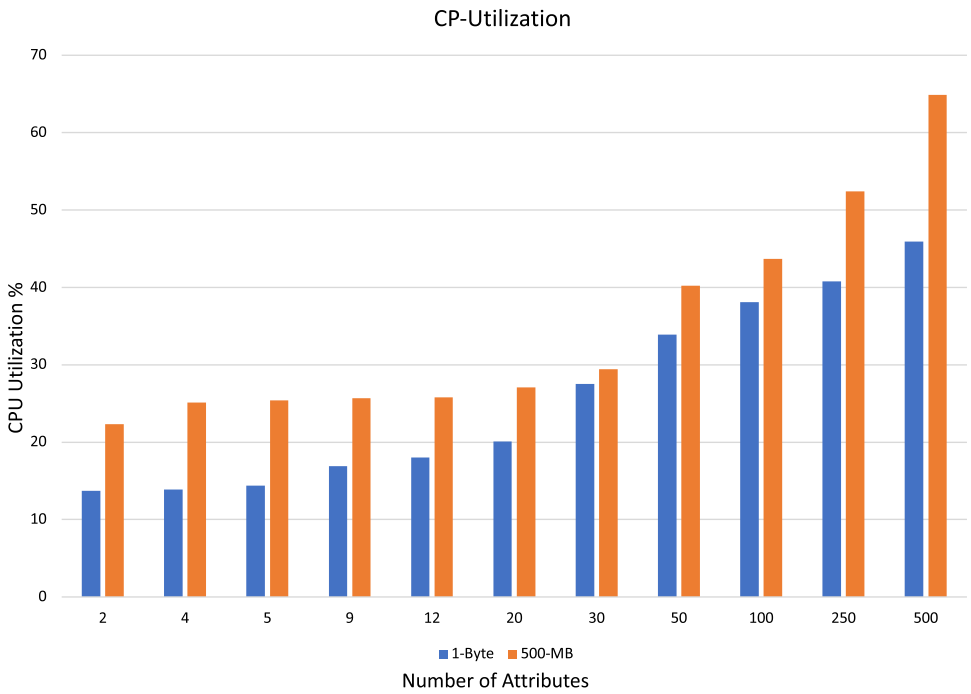

Fig. 8 CPU utilization of full-D encryption algorithm for the file size 1 Byte and 500 MB

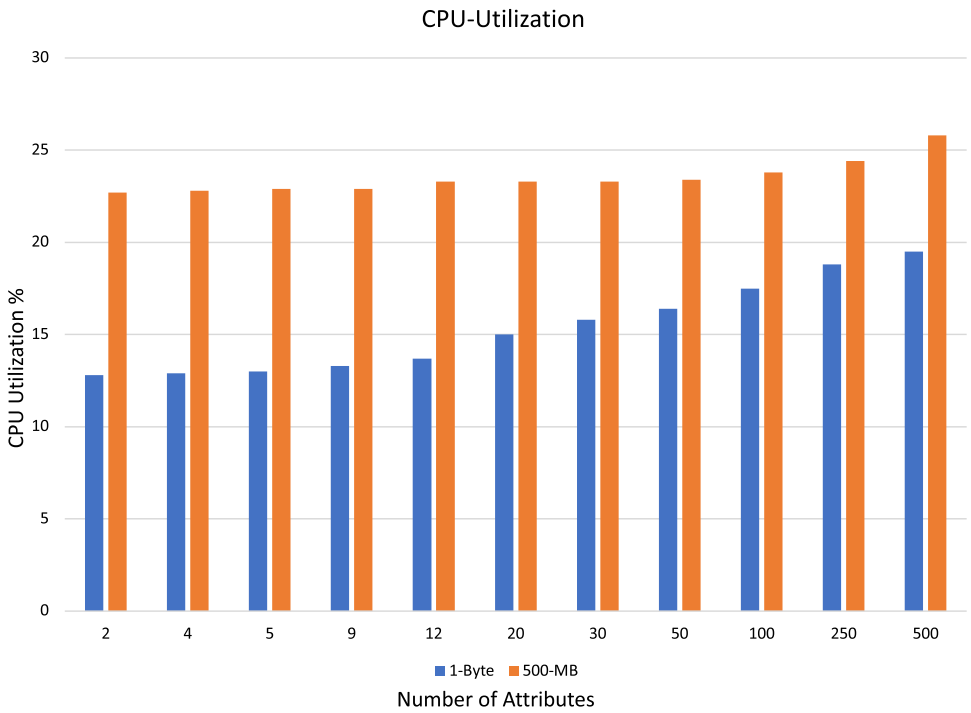

Fig. 9 CPU utilization of partial encryption algorithm for the file size 1 Byte and 500 MB

encrypted with access policies that have the same number of attributes. Figure 12 shows that in S3, full encryption is faster than partial encryption whereas in S4, partial encryption is faster than full encryption. The connection between RP1 and RP2 in S3 is WLAN which is slower than LAN. This makes full encryption faster than partial encryption. However, when the connection is LAN in S4, partial encryption is faster than full encryption.

S5, S6, and S7 are three scenarios with the same file size (1 KB) and number of attributes in the access policy (50). The difference between these three scenarios is in terms of available *Avg* – *Res* in RP1 and the connection type between RP1 and RP2. Our experiment shows that full encryption is faster than partial encryption in S5 whereas partial encryption is faster than full encryption in S6 and S7. This shows that full encryption becomes slow when

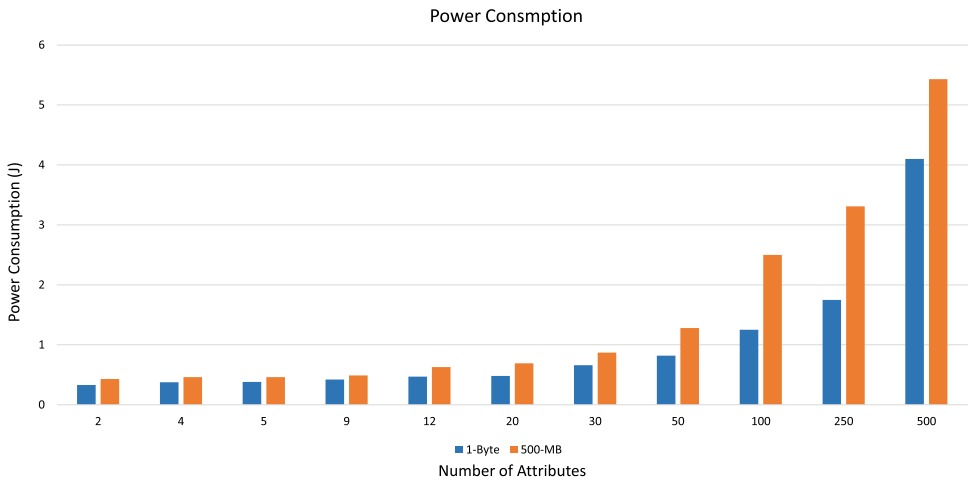

Fig. 10 Power consumption of full-D encryption algorithm for the file size 1 Byte and 500 MB

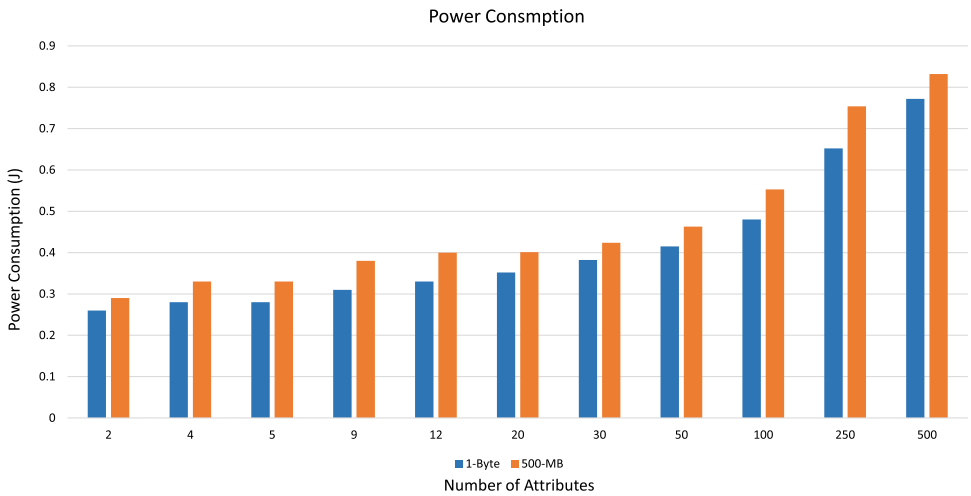

Fig. 11 Power consumption of partial encryption algorithm for the file size 1 Byte and 500 MB

the length of the access policy is large and the available $Avg - Res$ % is small. This also explains why full encryption is faster than partial encryption in S6 and S7.

In S8, S11, S13, and S15, full encryption is faster than partial encryption when there are enough resources and the number of attributes in the access policy are small. However, partial encryption is faster when there are not enough resources in constrained devices and the

Table 3 Technical specifications of the gateway and trusted nodeß

	*RP*1	*RP*2
CPU	Quad-Coretex A53 1.2GHz	Quad-Coretex A53@ 1.2GHz
Memory	1 GB SDRAM	1 GB SDRAM
Storage	MicroSD 4 GB	MicroSD 4 GB
Network	10/100 Mbps	10/100 Mbps

Table 4 Several scenarios

Scenario	$MSG - size$	$\lvert\mathbb{A}_{\mathbb{D}}\rvert$	$CPU - RP1$	Connection
S1	1-Byte	9	Medium	LAN
S2	1-KB	9	Medium	WLAN
S3	1-KB	9	Critical	WLAN
S4	1-KB	9	Critical	LAN
S5	1-KB	50	Idle	LAN
S6	1-KB	50	Medium	LAN
S7	1-KB	50	Medium	WLAN
S8	10-MB	12	Idle	LAN
S9	10-MB	12	Medium	LAN
S10	100-KB	5	Critical	LAN
S11	100-KB	5	Medium	LAN
S12	100-KB	5	Critical	WLAN
S13	1-MB	5	Medium	LAN
S14	1-Byte	20	Critical	WLAN
S15	1-MB	20	Idle	LAN

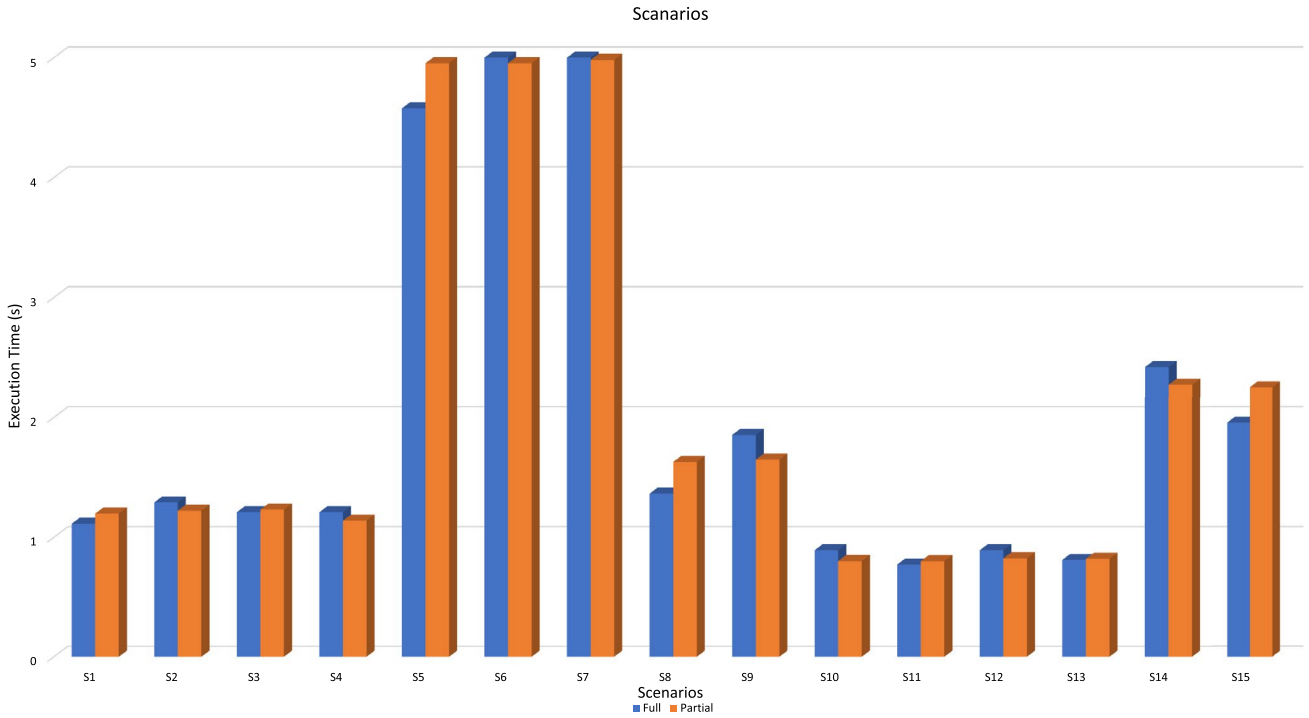

Fig. 12 Total time of full and partial CP-ABE encryption on different scenarios

number of attributes in the access policy is big such as in S9, S10, S12, and S14.

Based on the results presented in this section, we found that the total time of CP-ABE encryption algorithms depends on several factors in addition to the complexity of the access policy. In some cases, performing full encryption is faster than partial if there are enough resources available in constrained devices. Moreover, file size is the main factor for choosing between full and partial encryption. Finally, the connection between the constrained devices plays an important role in this decision since LAN is faster than WLAN.

8 Full or partial encryption decision x_D^i

As discussed in Sect. 7, the total time of the CP-ABE encryption algorithm depends on various factors such as length of access policy, file size, available resources (CPU, Memory, and power consumption), and network connection between devices. Hence, designing an adaptive scheme that is able to select the appropriate encryption technique (whether full or partial) at the time of encryption is mandatory. Therefore, a machine learning technique is needed to allow automatic selection of the encryption algorithm based on the context related to the data to be encrypted, the available resources on the device and the network connection characteristics. It is worth mentioning that these factors change over time.

There are several machine learning algorithms investigated in related work [27]. In this work, we utilize machine learning to select the appropriate encryption technique based on the execution context. Indeed, we have experimented with several machine learning algorithms using the Weka tool [28]. Hereafter, we recall some of the main metrics commonly investigated when we study machine learning algorithms.

The accuracy of a machine learning algorithm indicates the correctly classified elements [29]. Accuracy is defined by the following equation:

$$Accuracy = \frac{(TN + TP)}{(TP + FP + FN + TP)}$$

where TP, TN, FP, and FN are explained as follows. Each true predictive value that is equal to the actual value is considered as True Positive (TP). When a true predictive value is not equal to the actual value, it is called False Positive (FP). If the negative predictive value is equal to the actual value, then this value is called True Negative (TN). Finally, if the negative predictive value is not equal to the actual value, we call it False Negative (FN).

The exactness and quality of machine learning can be measured by the precision [30] metric defined by the following equation:

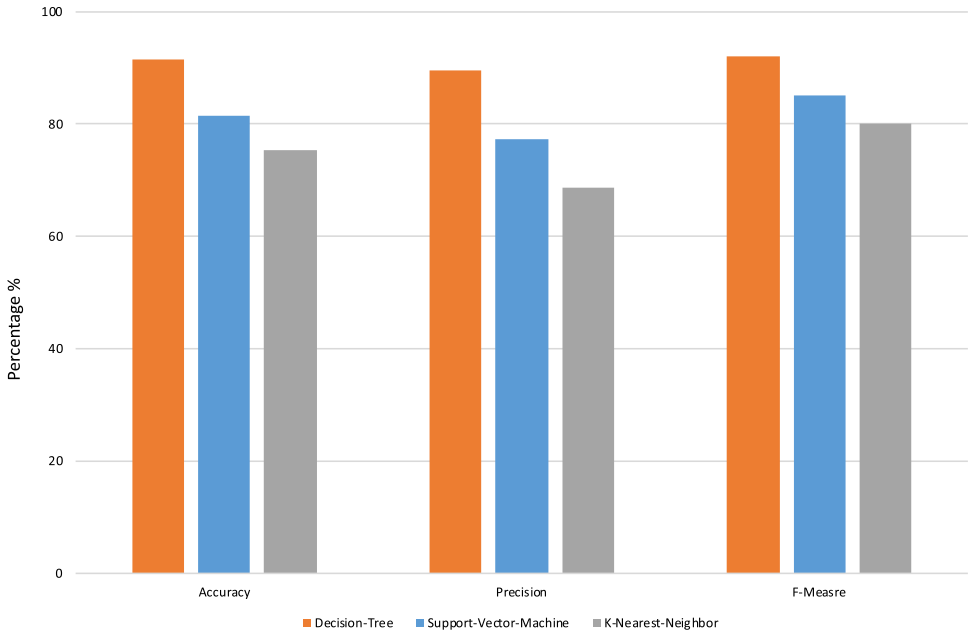

Fig. 13 Comparison between three different machine learning algorithms

$$Precision = \frac{TP}{TP + FP}$$

The completeness or quantity of the algorithm is measured by the recall metric defined by the following equation:

$$Recall = \frac{TP}{TP + FN}$$

Finally, the weight harmonic of precision and recall called the F-measure [29] metric is defined by the following equation:

$$F - measure = \frac{2 * precision * recall}{precision + recall}$$

In order to select the best-fit machine learning for our work, we built a dataset and we experimented with three machine learning algorithms: decision tree, Support-Vector-Machine, and K-Nearest-Neighbor. We have compared the results of the three algorithms in terms of accuracy, precision, and F-measure as shown in Fig. 13.

To build our dataset, we selected six files having the following sizes: 1byte, 1 KB, 10 KB, 100 KB, 1 MB, and 10 MB. For each file, we measured the CPU utilization and total time while varying the number of attributes in the access policy (from 2 to 500), the available CPU (Idle, Medium, Critical), and the connection type (LAN or wireless). For each of the tested configurations, we have identified the most appropriate scheme (either full or partial).

Based on our dataset, we found that the accuracy of the decision tree algorithm is 91.48. We use the decision tree algorithm to take the decision (x_D) whether Algorithm 4 should perform full or partial CP-ABE encryption. On the other hand, since we carried out the training phase of decision tree (DT) in capable device, we evaluate the overhead of decision tree model by running the pickle file deployed on raspberry pi to generate classification scores. Our results show that CPU, memory, and power consumption of the DT model are 13.5 MiB, 4.6 MiB, and 2.12 *J* respectively.

Referring back to Fig. 13, the accuracy of the decision tree algorithm is 91.48 whereas it is 81.5152 and 75.5251 in the Support-Vector-Machine and K-Nearest-Neighbor respectively. Based on these results, we have identified the decision tree as the best algorithm for our decision taking problem.

Figure 14 shows our decision tree. The tree compares first the number of attributes in access policy to five. If the number is less than or equal to four, it recommends performing full encryption. Otherwise, the decision tree checks the available $Avg - Res$ in the gateway device. $Avg - Res$ is the avarage resource (CPU, memory, battery) of IoT device. Hereby, the decision tree facilities the automation of the selection process to whether to perform the encryption or to check the file size. If the number of attributes in the access policy is more than five, then the tree checks the available $Avg - Res$ in the gateway device. Based on the available CPU, either it checks the file size or the number of attributes or it performs encryption.

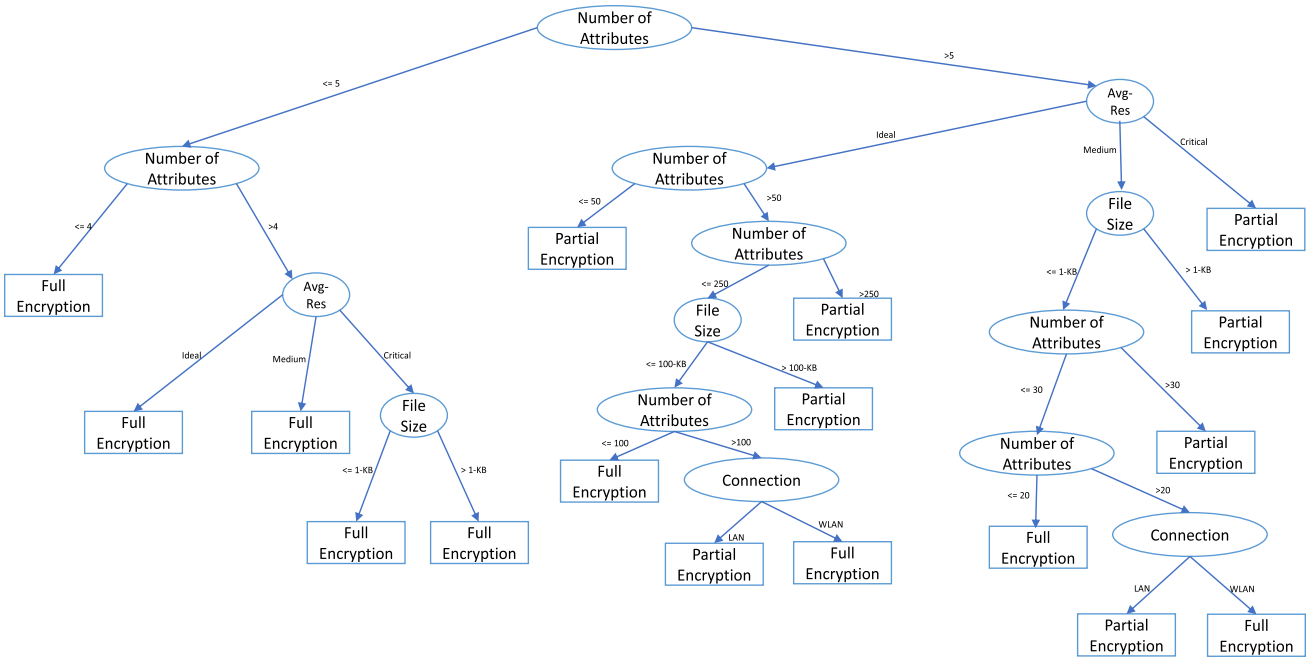

Fig. 14 Decision tree

The usage of a machine learning algorithm is motivated by the fact that it can smartly choose the appropriate encryption (whether full or partial encryption). The percentage of accuracy is important to select the appropriate machine learning algorithm [29].

9 Conclusion

In this article, we proposed a selective approach that allows choosing between performing all the CP-ABE encryption steps (full encryption) on constrained devices or performing only partial encryption and offloading the heaviest tasks to a remote device or proxy. We started the article by proposing an encryption scheme that allows switching between full and partial encryption without requiring different decryption keys. We provided experimental results that motivate the need for dynamic selection of the appropriate encryption scheme. Then we experimented with various machine learning algorithms over a dataset covering various configurations of parameters including available CPU, type of network connection, file size, and number of attributes.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Cuka M, Elmazi D, Bylykbashi K, Spaho E, Ikeda M, Barolli L (2019) Implementation and performance evaluation of two fuzzy-based systems for selection of iot devices in opportunistic networks. J Ambient Intell Hum Comput 10(2):519–529
2. Zanella A, Bui N, Castellani A, Vangelista L, Zorzi M (2014) Internet of things for smart cities. IEEE Internet Things J 1(1):22–32
3. Taha MB, Talhi C, Ould-Slimane H (2019) Performance evaluation of cp-abe schemes under constrained devices. Procedia Comput Sci 155:425–432
4. Ali M, Dhamotharan R, Khan E, Khan SU, Vasilakos AV, Li K, Zomaya AY (2017) Sedasc: secure data sharing in clouds. IEEE Syst J 11(2):395–404
5. Taha MMB, Chaisiri S, Ko RKL (2015) Trusted tamper-evident data provenance. In: Trustcom/BigDataSE/ISPA, 2015 IEEE, vol 1, pp 646–653. IEEE
6. Liu M, Wu Y, Xue R, Zhang R (2019) Verifiable outsourcing computation for modular exponentiation from shareable functions. Cluster Comput, 1–13
7. Miloslavskaya N, Tolstoy A (2019) Internet of things: information security challenges and solutions. Cluster Comput 22:103–119
8. Zhou L, Varadharajan V, Hitchens M (2015) Generic constructions for role-based encryption. Int J Inf Secur 14(5):417–430
9. Sasi SB, Dixon D, Wilson J, No P (2014) A general comparison of symmetric and asymmetric cryptosystems for WSNs and an overview of location based encryption technique for improving security. IOSR J Eng 4(3):1
10. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy (SP'07), pp 321–334. IEEE
11. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on computer and communications security, pp 89–98. ACM

12. Chowdhury R, Ould-Slimane H, Talhi C, Cheriet M (2017) Attribute-based encryption for preserving smart home data privacy. In: International conference on smart homes and health telematics, pp 185–197. Springer
13. Yao X, Chen Z, Tian Y (2015) A lightweight attribute-based encryption scheme for the internet of things. Future Gener Comput Syst 49:104–112
14. Morales-Sandoval M, Vega-Castillo AK, Diaz-Perez A (2014) A secure scheme for storage, retrieval, and sharing of digital documents in cloud computing using attribute-based encryption on mobile devices. Inf Secur J Global Perspect 23(1–2):22–31
15. Taha MB, Talhi C, Ould-Slimanec H (2019) A cluster of cp-abe microservices for vanet. Procedia Comput Sci 155:441–448
16. Borgh J, Ngai E, Ohlman B, Malik AM (2017) Employing attribute-based encryption in systems with resource constrained devices in an information-centric networking context. In: 2017 global internet of things summit (GIoTS). IEEE, pp 1–6
17. Bany Taha M, Talhi C, Ould-Slimane H, Alrabaee S (2020) TD-PSO: task distribution approach based on particle swarm optimization for vehicular ad hoc network. Trans Emerg Telecommun Technol, e3860
18. Zhou Z, Huang D (2012) Efficient and secure data storage operations for mobile cloud computing. In: Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm), pp 37–45. IEEE
19. Jin Y, Tian C, He H, Wang F (2015) A secure and lightweight data access control scheme for mobile cloud computing. In: 2015 IEEE fifth international conference on big data and cloud computing (BDCloud), pp 172–179. IEEE
20. Wang H, He D, Shen J, Zheng Z, Zhao C, Zhao M (2016) Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing. Soft Comput 21:7325–7335
21. Zhiyuan Z, Jianhua W (2017) Verifiable outsourced ciphertext-policy attribute-based encryption for mobile cloud computing. KSII Trans Internet Inf Syst 11(6):3254–3272
22. Touati L, Challal Y, Bouabdallah A (2014) C-cp-abe: cooperative ciphertext policy attribute-based encryption for the internet of things. In: 2014 international conference on advanced networking distributed systems and applications (INDS), pp 64–69. IEEE
23. Nguyen KT, Oualha N, Laurent M (2017) Securely outsourcing the ciphertext-policy attribute-based encryption. World Wide Web 21:169–183
24. Singh S, Singh N (2018) Role based security for cloud based data with data reliability. Int J Res Eng IT Soc Sci 07(ISSN: 2250-0588):23–28
25. Karp RM (1972) Reducibility among combinatorial problems. In: Complexity of computer computations. Springer, Boston, MA, pp 85–103
26. Charm (2011). https://github.com/JHUISI/charm
27. Cui L, Yang S, Chen F, Ming Z, Lu N, Qin J (2018) A survey on application of machine learning for internet of things. Int J Mach Learn Cybern 9:1399–1417
28. Mark H, Eibe F, Geoffrey H, Bernhard P, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD Explor Newslett 11(1):10–18
29. Peng Y, Kou G, Wang G, Wang H, Ko FIS (2009) Empirical evaluation of classifiers for software risk management. Int J Inf Technol Decis Mak 8(04):749–767
30. Patil TR, Sherekar SS (2013) Performance analysis of naive bayes and j48 classification algorithm for data classification. Int J Comput Sci Appl 6(2):256–261