IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 34, No. 3, June, 2020, pp. 233-241
Journal homepage: http://iieta.org/journals/ria

Adaptive Ciphertext Policy Attribute Based Encryption Scheme for Internet of Things Devices Using Decision Tree

Mohammad Bany Taha[1*], Hanan Suwi[2], Fawaz Khaswneh[3], Khaled Alzaareer[4]

[1] Ericsson, GAIA Department, 8275 Trans Canda Route, saint-Laurent, Montreal, Quebec, H4S 0B6, Canada
[2] Department of Software Engineering and IT, University of Quebec, Notre-Dame, Montreal, Quebec, H3C 1K3, Canada
[3] Ericsson, BDGS RDP Department, 8275 Trans Canda Route, Saint-Laurent, Montreal, Quebec, H4S 0B6, Canada
[4] Department of Electrical Engineering, University of Quebec, Notre-Dame, Montreal, Quebec, H3C 1K3, Canada

Corresponding Author Email: an35670@ens.etsmtl.ca

https://doi.org/10.18280/ria.340301

Received: 22 March 2020
Accepted: 5 May 2020

Keywords:
machine learning, IoT, CP-ABE, decision tree, offloading

ABSTRACT

The Internet of Things (IoT) has recently become a hot spot for researchers and its industrial importance is growing exponentially day after day. Statistics show that the number of IoT devices will reach fifty billion by 2020. IoT applications are backed through clouds where data is stored and processed by gigantic processing systems and accessed only by authorized users. In the present work, we outsource the encryption and decryption of ABE operations to proxy servers in order to reduce the computation overhead of encryption and decryption. However, in some cases performing full encryption is more worthy than partial encryption if IoT resources are fit to perform this encryption thus, our scheme allows to adaptively switch from partial encryption to full encryption based on the resources that available and the number of attributes in access tree. The decision maker component (manager) uses decision tree to select the appropriate scheme based on the context at encryption time. Finally, we evaluate the performance of our scheme with other scheme proposed for constrained devices.

1. INTRODUCTION

The fast development and growing of cloud computing help IoT applications to be more accessible; because most of IoT applications are based on cloud computing, that also helps IoT applications to rapidly grow. These factors encourage researchers to develop the cloud environment to be trustworthy for these applications [1-4]. IoT components can gather information remotely and store it on cloud computing. However, there are several challenges of data confidentiality in an IoT environment: Firstly, the confidentiality and privacy of IoT data may breach since this data is usually sensitive [5] and since cloud computing is honest but curious. Secondly, IoT devices are constrained devices and that leads to yet another challenge since there is no choice to keep this data on these devices, as it is constrained storage. Thirdly, providing the confidentiality of data means reserving many resources while IoT devices have limited resources. For all these challenges, providing the confidentiality and privacy of data in IoT devices has become a critical issue.

To fix the above mentioned problems, the data owner should either use an authenticated access control system that allows only authorized users to access the data, or encrypt the data before uploading it to the cloud. However, using an authenticated access control system is not completely secure because intruders could still access the data using malicious software [6]. Therefore, preserving the data by encrypting it prior to uploading it to the cloud is more suitable. However, encryption consumes high resources and that is considered a problem with IoT devices since they are constrained devices (CPU, RAM, storage, Battery). Symmetric encryption is considered lighter than asymmetric encryption [7]. Nevertheless, in symmetric encryption, the data owner should know the client who requests to read the data in order to send them the key and that is not suitable in an IoT scenario, since the data owner usually know nothing about the clients who want to decrypt the data. Additionally, in symmetric encryption, each ciphertext has the same key for encryption and decryption, therefore the clients will have the same keys, and this is considered as not secure. On the other hand, asymmetric encryption is more suitable than symmetric encryption in an IoT scenario, since the data owner will encrypt the data using a public key while the clients will use a private key to decrypt the data. This kind of encryption seems more suitable than symmetric encryption since the data encrypted with a public key is supposed to be available to everyone. However, in an IoT scenario, the data should be encrypted in context information or a set of attributes, since the data owner usually knows nothing about the client who wants to decrypt the data.

Attribute Based Encryption (ABE) was first proposed in 2006 by Goyal et al. The authors proposed a new form of asymmetric encryption called Key-Policy Attribute Based Encryption (KP-ABE). Later, in 2007, Bethencourt et al. proposed a new type of Attribute Based encryption called Ciphertext policy Attribute Based Encryption (CPABE). In this article, we will discuss only the schemes that are based on CP-ABE because it is more suitable for an IoT environment, since the data owner can specify who can decrypt the data. However, based on the aforementioned challenges discussed in the first paragraph of this section, ABE technique fixes the first two challenges but not the third one, because of the very

high computation cost of ABE schemes relatively to constrained devices. Several schemes have proposed to reduce the computation overhead of CP-ABE [8-11]. Most of these schemes optimize the overhead of CP-ABE operations but have several weaknesses, as per the following: firstly, some of the proposed schemes are not suitable for constrained devices because of performance and are not feasible to apply on these devices. Secondly, some proposed schemes are restricted to specific types of access policy, which makes them restricted for a limited range of applications.

The contribution of our work can be summrized as the following:

(1) We study the performance of two cryptography encryption schemes on IoT devices. The first scheme [12] is perform full encryption which consider hevey on IoT devices, whereas the second scheme is performed light encryption on IoT devices [12].

(2) We proposed adaptive scheme using machine learning that can switch from a scheme to another based on the context at encryption time. We proof that the switching increases the efficiency of the scheme.

(3) Finally, we validate our scheme by analyzing the performance of our scheme and comparing the results with other schemes.

2. RELATED WORK

In this section, we mainly focus on ABE schemes that proposed to reduce the computation cost of ABE by delegating the cryptography operations on constrained devices. On the other hand, we will discuss the existing works that proposed to generate constant ciphertext on IoT devices to reduce their storage overhead.

Zhou et al. proposed an efficient CP-ABE scheme that securely outsources most of encryption and decryption operations to the cloud [13]. As each access policy consists of a left sub-tree and a right subtree, 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. The Encryption Proxy Service EPS performs the encryption of the left sub-tree and generates CT_{EPS}. Finally, EPS generates the final ciphertext CT by combining CT_1 and CT_{EPS}. Decryption Proxy Service DPS performs most of the decryption operations by decrypting the left sub-tree of the access policy in order to generate CT_{out}. Client decrypts CT_{out} which consists of only the right sub-tree. As a result, the user and the client will perform the small part of the cryptography operations based on the number of attributes on the right sub-tree of the access tree. However, the final CT contains the attributes in the left sub-tree and the attributes in the right sub-tree which means that the root node must always be AND in the access policy. In case that the root node of the access policy is OR logic gate, then this scheme will not work properly. Moreover, if the number of attributes on the right side of the access policy is large then the scheme is not efficient to use in IoT devices.

Jin et al. improved Zhou et al.'s work by proposing a flexible, secure and lightweight scheme of ABE on mobile devices [13]. The restriction that arises in Zhou et al.'s scheme is fixed in Jin et al.'s scheme by adding a dummy attribute to the right sub-tree of the whole access policy. The scheme is based on CP-ABE. The main components of the scheme are Trust Authority (TA), Data Owner (DO), Encryption Proxy Server (EPS), Storage Service Provider (SSP), Decryption Proxy Server (DPS), and Data Requester (DR). Jin et al. scheme delegates most of the intensive ABE operations to Mobile Cloud Computing (MCC), and guarantees that neither EPS nor the cloud provider hosting the data can reveal that data. The authors suggest outsourcing most of ABE to the proxy server, thus reducing the execution time of encryption and decryption on user and client devices respectively. Jin et al. proposed a dummy attribute, namely (Att_{Dum}), which is owned by every user. The user first encrypts their data with the right sub-tree of the access policy that contains only the dummy attribute. Then, the user performs some non-critical operations to generate security parameters. Finally, the user sends the initial form of ciphertext, namely CT_{Dum} to EPS. EPS will generate the final form of the ciphertext CT by performing the CP-ABE on the real access policy at EPS to generate CT_{Acc} and finally combine it with CT_{Dum}. The final form of CT = $CT_{Acc} \wedge CT_{Dum}$. As such for decryption, DPS does most of the ABE decryption operations to generate the initial form of CT_1 and sends it to the client device. The client device does a small part of the decryption because DPS did most of these operations. Client decrypts CT_1, the decryption operation on the client side recovers the message encrypted with the dummy attribute (Att_{Dum}) that the right sub-tree of the access policy consists of. This scheme reduces computation cost of encryption and decryption operations. However, the authors do not provide any details to answer how to choose or generate a secure and unique dummy attribute for each user. Moreover, the ciphertext's size has increased with the number of attributes in access policy.

Wang et al. proposed a verified outsourcing ABE scheme for keygen, encryption and decryption, a scheme that successfully reduces the execution time. the CT becomes more complex and the size of it increases with the number of attributes in the access policy [14]. The user encrypts the data partially to generate EP_O and EP_L, EP_L never leaving the user device and EP_O send to EPS. EPS performs more operations that need more computation cost and sends the output parameters back to the user. However, the scheme consumes high communication cost on user and client machines.

Zhao et al. proposed a scheme [15] similar to Wang et al.'s works. Zhao et al. encrypt the message using symmetric encryption first. After that, DO uses ABE to encrypt random message with default attributes and sends CT_1 with C_{SE} to EPS. Same technique as in the study [14] is applied for the remaining part of the algorithms. Nguyen et al. proposed to outsource ABE scheme based on CP-ABE [16]. The scheme consists of Data Producer (DP), Data consumer (DC), Delegatee (DG) which is responsible of generating the final form of CT, and Trusted Key Distribution Centre (KDC). KDC is responsible of generating the SK, and also performs some of the cryptography operations to generate the security parameters and delegation key that are needed to form the final CT. The user performs only one exponentiation to generate the initial CT_1, but DG performs most of the expensive operations. The KDC is responsible of generating the delegation key and the user only specifies the access policy before passing it on to DG. 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 SK, then the client will be able to decrypt the data since there is no relation between C' (generated by user) and the access policy that the user sends to DG.

Storage overhead is also one of the challenges on IoT devices, since these devices also have limited storage resources, therefore several schemes are proposed to generate constant ciphertext. Cheung and Newport proposed hierarchical attributes scheme to reduce the size of ciphertext as well as the execution time of encryption and decryption [17]. The scheme is based on AND gate, the attributes have multiple values (i.e., positive, negative, and do not care), thus the ciphertext will be generated for each attribute based on the value of these attributes. To simplify their idea, assume that the C_i is computed for $attr_i$ then all C_j :j $\forall\tau$ will be part of the ciphertext, where C_i is a public parameter with the value of $attr_i$. In this case, the attributes are arranged into a hierarchical form that can improve the efficiency because Cheung and Newport use few groups of elements to find C_i. However, considering all C_j will be on the ciphertext, the ciphertext's size still depends on the number of attributes in the access policy. Moreover, if the user wants to encrypt his access policy with a new attribute, he must re-run setup algorithm again otherwise the client who kept the old SK (containing old attributes) will be able to decrypt the ciphertext that contains the new attributes because the old value of secret shared has not been changed. Emura et al proposed a scheme that improved Cheung et al.'s scheme. Their scheme is based on AND gate, generates constant ciphertext and reduces the cryptography operations on the user side and client side [18]. In Cheung et al's. scheme, the CT included all C_i based on the value of $attr_i$ then the number of CT is i, where i is [1,2,3, ...n]. Emura et al. found that the ciphertext should not have redundant values, as it is proposed in Cheung and Newport scheme, such as when the value of an attribute is not listed at the access policy $v_{i,j} \in /$ A where A is the access policy. Secondly, the access structure can be expressed by the summation of the master key. Emura *et al* calculate the encrypted value for all attributes in case it is an element in the access tree or not, and that will generate a constant ciphertext. Doshi and Jinwala [19] improved Emura et al's. works, since one of the drawbacks of Emura's scheme is that the attributes in secret key must be the same as the attributes in the access policy. Doshi and Jinwala proposed a CP-ABE scheme based on AND gate that provides constant ciphertext and where the attribute in ciphertext could be a subset of the attributes in the secret key [19].

Based on aforementioned literature, we summarized the weaknesses and the gaps of the existing outsourcing scheme.

Zhou et al. proposed an efficient scheme [12] to securely outsource most of ABE operations but it is restricted to specific type of access policy (the root access policy must be AND). The scheme proposed in fixed the flexibility problem that arises in the study [12], using a dummy attribute so that any access policy could add a dummy attribute, and that allowed a wide range of application to use this scheme. Wang et al.' [14] scheme outsourced the key-gen, encryption and decryption, which means that we have to trust several components at this scheme, such as proxy encryption and decryption. Nguyen et al. [16] reduces the number of exponentiations on user device by allowing a key generator to generate some of the security parameters and the delegation key. Schemes in the researches [17-19] discussed how to reduce the storage overhead by generating a constant ciphertext. Table 1 shows the number of exponentiations for most schemes we discussed in this section. In Table 1, Nguyen et al.' scheme [16] needs only one exponentiation. The scheme performance at the user's device [12] is based on the right sub-tree, therefore if we assume the right subtree consists of only one attribute, then the number of exponentiations will be 3, which is the same as the scheme proposed in the study [12]. Most of the listed schemes in Table 1 need only one pairing for decryption. Scheme [16] does not support outsource decryption. Scheme [12] needs 2 $|\tau|$ + 1, where $|\tau|$ is the length of attributes in access policy.

Table 1. Exponentiations and pairing on several schemes

Scheme	Number of Exponentiation DO	Number of Pairing DR
[11]	n + 2	2|T| + 1
[12]	3	1
[13]	4	1
[14]	4	1
[19]	N/A	1
[15]	1	N/A

Based on that, we can summarize our scheme requirements as per the following:

- Correctness: The scheme should allow only the authorized user to access the data.
- scalability: The scheme should be scalable for a wide range of applications and should not be restricted.
- Feasibility: The scheme should be usable on constrained device.
- Security: The scheme should be secure, only the user machine and the Trust Authority machine being the trusted on the scheme.

3. PRELIMINARIES

In this section, we further outline some technical terminologies that we use in this article.

3.1 Bilinear map

Our scheme and most of CP-ABE scheme are based on bilinear map. Assume G0, G1 are two multiplicative group of prime order p. Then map e: G0 * G0 → G1 is a bilinear map. Bilinear map has three properties:

- Bilinear: $e(g^a,g^b)= e(g,g)^{a,b}$
- Non-degenerate:e(g,g) 6=1
- Computable: $e(g^a,g^b)= e(g,g)^{a,b}= e(g^b,g^a)$

3.2 CP-ABE

CP-ABE is a form of public key encryption. The data encrypt with access policy, where the secret key blind with number of attributes. The original form of CP-ABE consists of four algorithms as the following:

- Setup (λ)→(PK, MSK). the algorithm generates the public key (PK) and the master key (MK).
- KeyGen(PK, ω, MK)→ SK. SK, where SK is secret key ,and ω is the client's attributes list.
- Encryption (PK, M, A)→CT. Where CT is the ciphertext and A is User's access policy.
- Decryption(CT,SK)→ M. where M is the message.

3.3 Access tree

Access tree used to describe the access policy. Access tree consists of set of nodes. The top node called the root node where inner nodes are either 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 3 shows sample of access tree ((X AND Y) OR (C AND Z)) with the components of CT. The important thing about the tree is the fact that each node has secret share and all these shares are required to reconstruct the root share that the other shared of nodes tree derive from and no one has any information about this share except the client who has secret key [20, 21].

4. THE PROPOSED SCHEME

In this section we will discuss our framework scheme. We design a scheme can applied in several IoT scenarios such as smart home, healthcare, or smart city. In IoT scenarios, each scenario has different requirements such as the time of requesting to encrypt the data, also each data has different access policy to encrypt with. Moreover, the availability of IoT resources are changes from time to time based on tasks that the IoT devices are perform. Based on aforementioned, the information context of attributes and the resources availability should take in consideration for selecting the way that the data should encrypt with. We design a decision maker component we call it manager Figure 10 show the components of manager, we will further discuss the manager components in section 6.

Before we start discussing our scheme we will explain some proposed CP-ABE scheme for IoT devices. Figure 1 shows three scenarios of IoT framework. In scenario 1, IoT device can completely encrypt the data by performing full encryption. The data encrypted with real access policy and generate the final CT then upload it to the cloud. There are several schemes proposed for scenario 1 such as BSW scheme [22, 23]. Scenario 2 in Figure1 the data encrypted partially. In partial encryption IoT devices encrypts the data with only dummy attribute and delegate most of CP-ABE cryptography operations to EPS server. Several schemes proposed this idea such as the scheme proposed in [12, 24, 25]. The details of full and partial encryption will discuss later in this section. Scenario 3 in Figure 1 shows our scheme. To take advantage of two scheme we found that in some cases under specific situations IoT device can perform full encryption since outsourcing CP-ABE operations are not worthy always because the policy context and the resources of IoT devices are rapidly changed. In scenario 3 our scheme performing full encryption if the resources in IoT device are underloaded and the number of attributes in access policy is not large. Otherwise, the manager will decide to perform partial encryption.

In scenario 1, the data is fully encrypted in IoT device and the final ciphertext is uploaded to the cloud. In full encryption IoT device need only to communicate with cloud once to send the final CT and this reducing the communication overhead. However, performing all CP-ABE operations on the same IoT device will increase the computation cost on this device. Therefore, using the second scenario will reduce the computation cost of CP-ABE operations on IoT device by delegation most of CP-ABE computation cost to EPS. However, the connection between EPS and IoT device must be available always. Moreover, in some scenarios EPS's ciphertext should return to IoT device then IoT device detriment the final form of ciphertext before upload it to the cloud and that will increase the communication overhead on IoT device and need more time for overall encryption time. In some IoT scenarios IoT devices can perform key generation in addition to encryption task. In some environments the number of clients is not large which means they can generate the secret keys inside IoT device. However, this will increase the computation on IoT device in addition to encryption computation cost.

Based on aforementioned, full encryption in some situation is more worthy to use than partial encryption because it is more secure since all CP-ABE operations are performed in trust part. However, we found that the execution time becomes unacceptable in some case when the number of attributes in access policy become more. Moreover, the CPU utilization and RAM usage become infeasible to apply in IoT device.

We measure the execution time of encryption operations in IoT device (Beaglebone black) of scenario one and two. Figure 2 shows the execution time of encryption operations on Beaglebone device for BSW [22] and Dummy [12] schemes. BSW scheme perform full encryption as explained in scenario 1 whereas Dummy-scheme represents scenario 2. as shown 2 the execution time of encryption operation is very close in BSW-scheme and Dummy-scheme when the number of attributes is less than five attributes. However, the difference in execution time of encryption between BSW-Scheme and Dummy-scheme increase when the number of attributes in access policy becomes more than five attributes. Based on that, we can perform full encryption when the number of attributes is less than five attributes and partial encryption when the number of attributes is more than five attributes. Designing scheme that allow to switch from full to partial encryption require to use same scheme architecture. More precisely, using BSW scheme to perform full encryption and Dummy scheme to perform partial encryption required two secret keys for each client to decrypt the data, one suitable for full scheme if the data encrypted using full encryption and other key to decrypt the data if the data encrypted with Dummy-scheme, the reason is in dummy architecture the data encrypted with dummy attribute which it is not in BSW scheme which means the secret key of BSW's scheme cannot decrypt the data encrypted with dummy attribute. In this case if IoT device will perform full encryption then it has to add only one dummy attribute to the real attributes. The execution time of encryption operations in BSW and Dummy-scheme shown in Figure 2. The difference between BSW and Dummy scheme in term of execution time for encryption algorithms in less than 1 second instead, we use only one scheme for full and partial encryption and one scheme to generate secret key for the client instead having two keys (one for BSW scheme and second one for Dummy scheme). Moreover, adaptive scheme will allow to perform full encryption if the number attributes is less than five in access policy and the resources of IoT devices are underloaded.

We also measure the difference of execution time of generating secret keys and the size of secrete keys in IoT device (Beaglebone black) for BSW and our scheme as shown in Figure 3 and 4 respectively. As shown in Figure 3 our need 0.06 second more to generate the secret key than BSW scheme and that is not really difference specially when the number of attributes is more than fifteen. Figure 4 shows the secret key size of BSW and our scheme, the differences are always constant around 665 bytes regardless of the number of attributes in secret key.

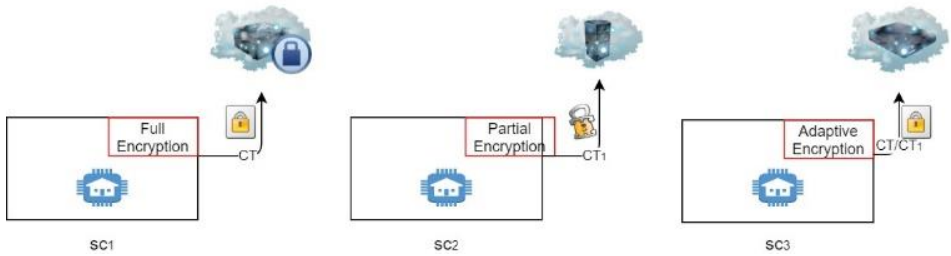

Figure 1. sc_1, sc_2, and sc_3 IoT framework

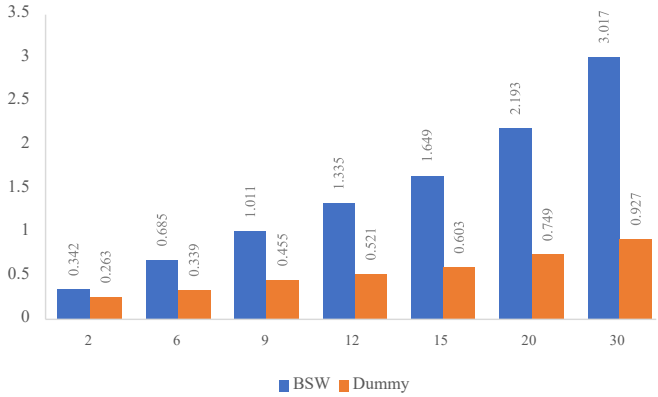

Figure 2. Execution time of encryption at scenario 1 and scenario 2

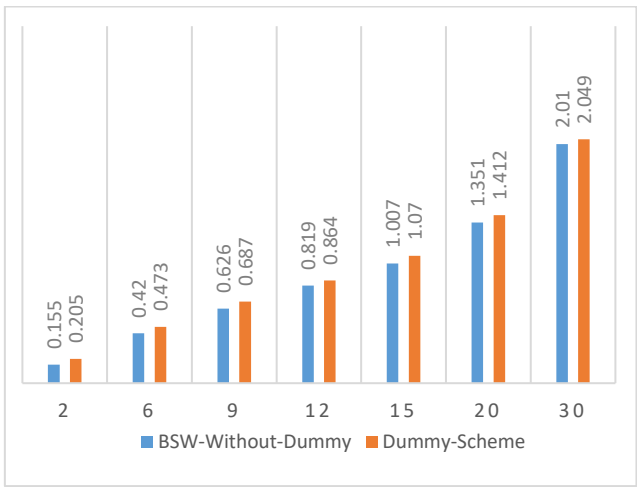

Figure 3. Execution time of key generation for BSW and partial scheme

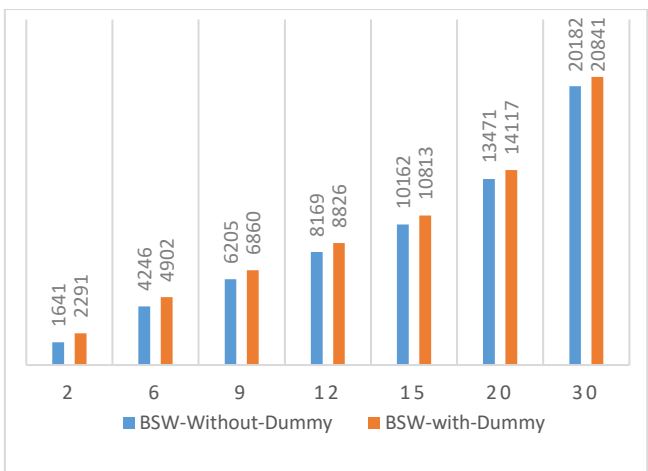

Figure 4. Secret key size for BSW and partial scheme

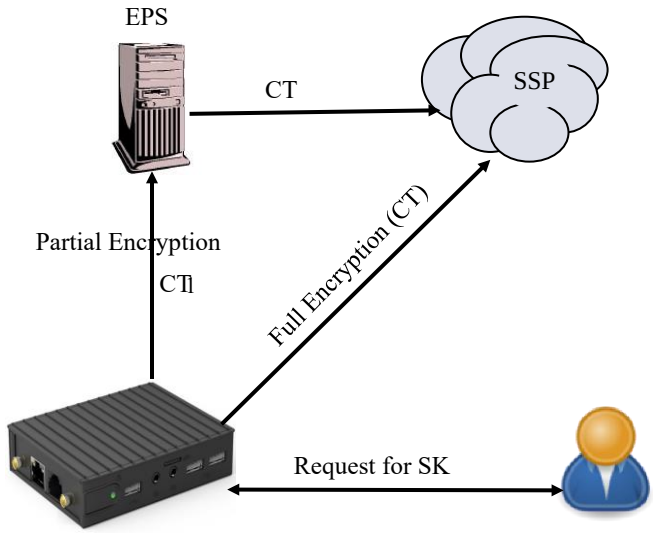

Figure 5. Proposed framework

Our scheme shown in Figure 5. In Figure 5, DO is the data owner (user), Client is the person who want to decrypt the data, EPS is Encryption Proxy Server. SSP is Storage Service Provider.

The scheme based on CP-ABE, the DO generate PK by running Algorithm 1. Data Owner runs Algorithm either 3 or 4 to encrypt the data and send it to either EPS or SSP. The manager component in IoT device will decide to perform full or partial encryption based on the access policy and the resources available in the machine.

If the manager decides to perform full encryption then Algorithm 3 runs in DO's machine and upload CT to SSP without needing to send the data to EPS. Otherwise, in case the manager chooses to perform partial encryption then, DO machine will generate CT_1 and send it to EPS. Table 2 show the notations we use in this article.

Table 2. Notations

Notation	Description
$\mathbb{A}$	Access policy
|T|	Attributes list in Access Policy
ω	User's attributes
K	Symmetric key
$\models$	Satisfy
CT1	Initial Ciphertext
CT	Final ciphertext
Ct, Ci, C	Ciphertext components
EPS	Encryption Proxy Provider
SSP	Storage Service Provider
DPS	Decryption Proxy Service
User	The encryptor
Client	The decryptor

4.1 Setup

Algorithm 1 shows the setup algorithm steps, this Algorithm runs at IoT device to generate PK and MK.

Algorithm 1 Setup Algorithm
Input: Security Parameters λ
Output: PK, MK
1: Generate g, gp from bilinear group G_1 and G_2
2: Select random elements $\alpha, \beta \in Z_R$
3: Calculate PK = G0, g, h= g^{β}, f= $g^{arg(1/\beta)}$, $e(g,g)^{\alpha}$, MK= (β, gα)
4: return PK, MK

4.2 Key generation

Algorithm 2 runs in IoT device to generate the SK.

Algorithm 2 Key Generator Algorithm
Input: PK, MK, ω
Output: SK
Initialisation:
1: Select random element r
2: Calculate D = $g_r^{\alpha-\beta}$
3: calculate D_j and D_{jp} for each attribute in ω
LOOP Process
4: for j in ω do
5: Select random element rj
6: Calculate D_j, D_{jp}
7: **end** for
8: return *SK*

4.3 User encryption

The user can either perform full or Dummy encryption as the following:

Algorithm 3 User Full Encryption Algorithm
Input: PK, M, A
Output: CT
1: Select random element s from $\in$ ZR
2: Generate key (k) from group element GT
3: Calculate secret shared$^{\epsilon}$, $\forall$ j in $\mathbb{A}$
4: Compute $C_0 = g^s$, $C_t = M \times e(g,g)^{\alpha s}$
5: calculate Cy = pk^{ϵ}, Cyp = $H(i)^{\epsilon}$,$\forall$ j in $\mathbb{A}$
6: return CT

4.3.1 Full encryption
As shows in algorithm 3, the user's data (M) encrypted with access policy A. The full encryption will calculate the all security parameters need to generate CT as shown in Algorithm 3.

Algorithm 4 User Dummy Encryption Algorithm
Input: PK, M, A
Output: C_t, C_t, C,
1: Generate key (k) from group element GT
2: select random element s from $\in$ ZR
3: Calculate secret shared$^{\epsilon}$, $\forall$ j in $\mathbb{A}$
4: Compute C = g^s, $C_t = M \times e(g,g)^{\alpha s}$
5: return CT_1

4.3.2 Partial encryption
As shows in algorithm 4 the user's data (M) encrypted with access policy A. In order to reduce the computation cost at DO machine we delegate most of the cryptography operations to EPS machine. In Algorithm 4 we only need two exponentiations to generate CT_1. Meanwhile, we ensure that our scheme is secure even we delegate most of the cryptography operations to EPS. Moreover, we reduce the communication cost between DO machine and SSP by allowing EPS to combine CT_{Dum} and CT_{EPS} to generate and upload the final CT.

The secret shared is used in the access policy, Figure 6 shows how generated for each node (s1,2,3…)•ϵ is re-divide into sub-secret for each node in the access tree in a way that all sub-secret required to reconstruct the original value of s.

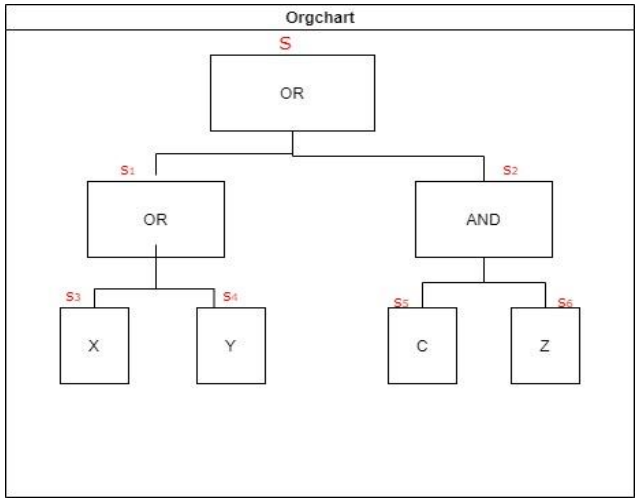

Figure 6. Construction of S

4.4 Encryption proxy service

This stage performs at EPS if the manager components decide to perform partial encryption. Algorithm 5 shows the main steps that are performed in order to generate the final ciphertext CT. The user must pass CT_1 to EPS to perform the second part of encryption. τ in Algorithm 5is the list of attribute in A.

EPS should calculate C_y and C_{yp} for each attribute in A because calculating C_y and C_{yp} perform two exponentiations for each attribute and this cause high computation overhead as we will see in section 5.

Algorithm 5 EncryptionOut
Input: PK,ϵ , $\mathbb{A}$
Output: C_y, C_{yp}
LOOP Process
1: for j in ω do
2: calculate Cy = pk,
3: Cyp = $H(i)^{\epsilon}$,$\forall$ j in $\mathbb{A}$
4: end for
5: Generate the final form of CT C,C_t, C_y, C_{yp} }, τ,$\mathbb{A}$
6: return CT

5. IMPLEMENTATION AND PERFORMANCE EVALUATION

In this section we further outline the performance and evaluation of our scheme, we also compare our scheme's results with [12]'s scheme results. We test our scheme on

Beaglebone black, Beaglebone is an IoT device, the specification of Beaglebone black shows in Table 3. In order to evaluate the efficiency of our and Jin et al.'s schemes we measure the execution time of encryption, decryption operations in DO and DR machines. We also found the CT size in DO machine in both schemes. Moreover, we test both schemes in symmetric curve (Super Singular) 512-bit and asymmetric curve type A ($y^2 = x^3 + x$) MNT 159, 201,224. Finally, we evaluate the computation overhead of two schemes by measured the memory usage of these schemes on DO and DR using python script.

Table 3. Beaglebone black specs

	Description
Processor	ARM 1GB Cortex A8
memory	256 MB
Flash	4GB MicroSD
Clock Speed	1000 MHz
OS	Linux beaglebone 4.4.9-ti-r25

5.1 Execution time

Definition 1: The execution time of encryption operation on user's machine is effected by the number of attributes in access policy in full and partial encryption.

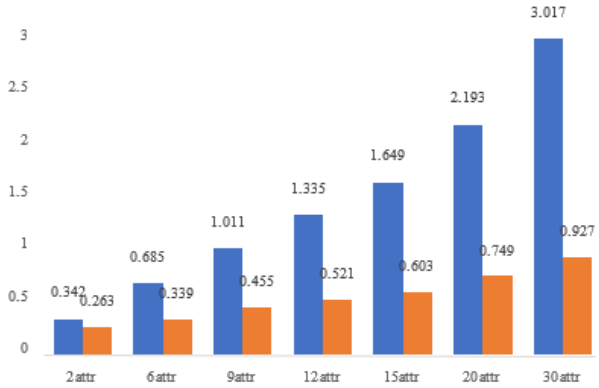

Figure 7. Execution time of full and partial encryption at user machine

Figure 7 shows the execution time of full and partial encryption. The execution time of partial and full encryption is very closed when the number of attributes less than five attributes. However, the differences in time become more when the number of attributes increase. The execution time of partial encryption is less than of full encryption because in partial encryption the data encrypt with only dummy attribute. However, in partial encryption the user machine must calculate the shares values for all attributes in access policy and that would explain the slightly increase of encryption time in partial encryption. In full encryption Algorithm (3) user machine should performs more cryptography operations than the algorithm of partial encryption (4) and that explain why the execution time of full encryption is more than the execution time of partial encryption.

5.2 CPU utilization

Definition 2: CPU utilization on partial encryption algorithm is less than the CPU utilization on Full encryption algorithm.

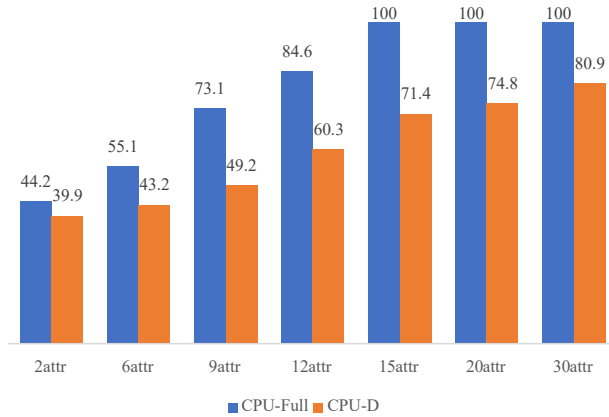

Figure 8. CPU utilization of full and partial encryption

Figure 8 shows CPU utilization of full and partial encryption on user machine. the percentage of CPU utilization is increased when the number of attributes is increase in access policy. For example, in Full encryption the CPU utilization is acceptable when the number of attributes is less than twelve. However, it become not acceptable when the number of attributes is more that twelve. Whereas, the CPU utilization is acceptable for thirty attributes if we use partial encryption.

5.3 Memory utilization

Defintion 3: Memory usage of partial encryption algorithm is less than the memory usage of full encryption algorithm. Figure 9 shows the Memory usage in full and partial encryption on user machine. Full encryption consumes more memory usage than partial encryption. In term of memory usage, full and partial encryption are acceptable when the number of attributes in access policy is thirty or less.

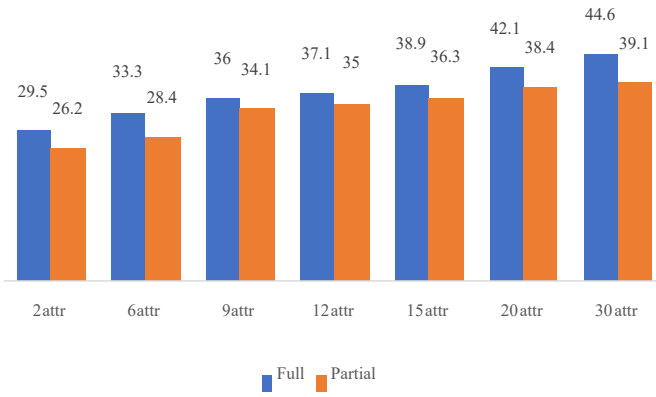

Figure 9. Memory utilization of full and partial encryption

6. ADAPTIVE CP-ABE

In this section we discuss our Adaptive CP-ABE scheme. The main goal of our scheme is to reduce the computation cost of CP-ABE operations, in section 4 we discussed two ways to encrypt the data using of CP-ABE (Full and Partial encryption). Full encryption will perform all cryptography operations for encryption at user machine and upload CT to the cloud, thus will consume high resources on user machine. However, in full encryption the user machine does not need proxy server to perform some cryptography operations and that need less communication overhead. Figure 11 shows the fluctuation of

execution time for CP-ABE encryption for file size 100 KB. For Partial and full encryption schemes [12, 22] respectively. It is clear that the efficiency of the scheme depends on the context of the resources (CPU, Memory) in addition to the factors that we discussed above.

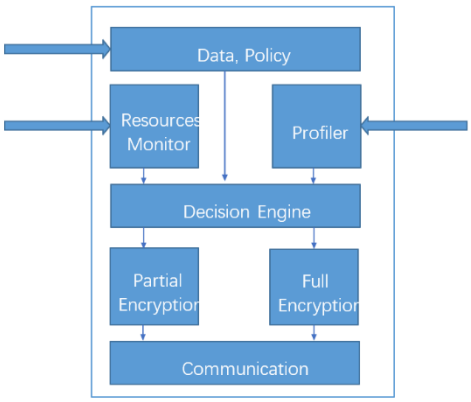

Figure 10. Manager components

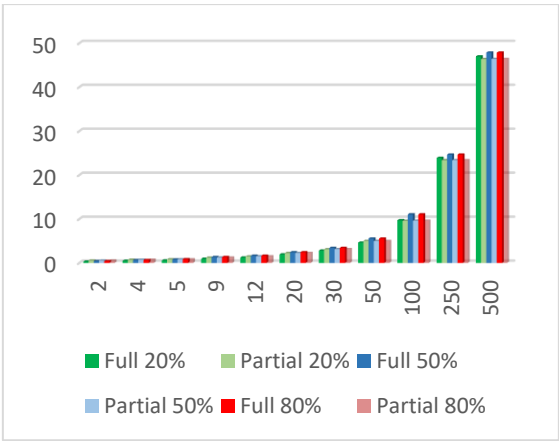

Figure 11. Encryption time for schemes [12] and [22] on file size 100 KB

In our scheme we design a manager component that can specify which encryption that user machine should perform (either full or partial encryption). Figure 10 shows the manager components. In Figure 10 the data is the user's message, the access policy is the access policy that the user wants to encrypt her data with. The profiler and Resource Monitor are too frequently check the device resource status. Manager component has decision engine to select which and where data should be encrypted. Finally, the communication part is to connect the manager component with IoT device.

The objective of manager component is to decide which kind of encryption should perform and where. As we discussed early, in some cases partial encryption is not always worthy to perform and since the context and the length of access policy rapidly changed. Whereas, partial encryption is worthy if the length of access policy is large and the IoT resources are overloaded.

6.1 Decision tree

We use decision tree as it shows high accuracy than other techniques. We compare the accuracy of decision tree with Sequential minimal optimization (SMO) and Naive Bayes. The accuracy for decision tree was 94.3% whereas it 85% and 92.7 for SMO and N-Bayes respectively.

The machine learning is work based on the dataset. Our dataset consists of file size, status of resources, connection type, and the length of the access policy. The label column of the dataset is the decision (full or partial). The selection is based on which one shows better results (less time). As shows in Figure 12, the tree checks the number of attributes in the access policy first, then based on the number of attributes its checks the number of attributes again or the CPU status. Then the data size or number of attributes again based on the status as shows in the figure.

7. CONCLUSION

In this article we discuss several schemes proposed to provide CPABE for constrained devices. All of these schemes either provide full encryption to encrypt the data before upload it to the cloud or providing partial encryption before upload it to the cloud and then complete the remind part of encryption on cloud or trusted server. In our scheme we design adaptive CP-ABE technique that allow to constrained device to either perform full encryption or partial encryption based on the available resources of constrained device and based on the context and the number of attributes in access policy. We found that performing full encryption is more beneficial than partial encryption in some cases whereas performing partial encryption is worthy in another cases.

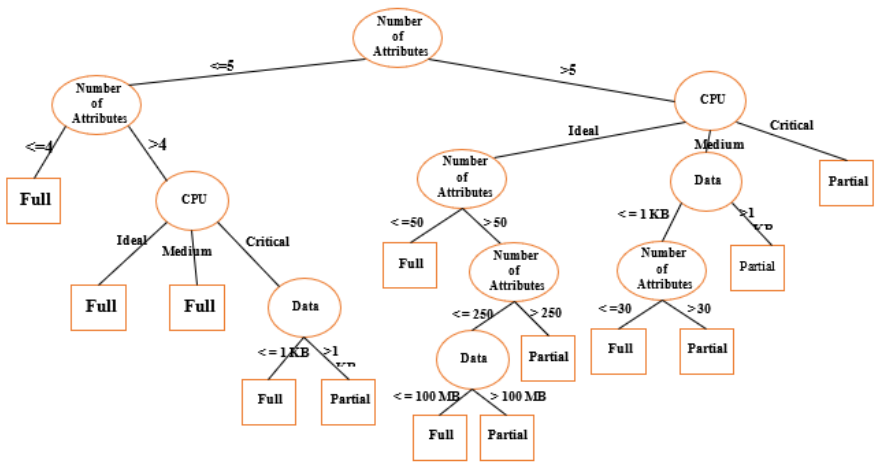

Figure 12. Decision tree

REFERENCES

[1] Li, S.C., Li, D.X. (2017). Securing the Internet of Things. Syngress.

[2] Fuad, A., Rahman, A., Daud, M., Mohamad, Z. (2016). Securing sensor to cloud ecosystem using internet of things (IoT) security framework. In Proceedings of the International Conference on Internet of things and Cloud Computing, Okinawa, Japan, pp. 1-5. https://doi.org/10.1145/2896387.2906198

[3] Zhou, J., Cao, Z.F., Dong, X.L., Vasilakos, A. (2017). Security and privacy for cloud-based IoT: Challenges. IEEE Communications Magazine, 55(1): 26-33. https://doi.org/10.1109/MCOM.2017.1600363CM

[4] Taha, M.M.B., Chaisiri, S., Ko, R.K.L. (2015). Trusted tamper-evident data provenance. 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, pp. 646-653. https://doi.org/10.1109/Trustcom.2015.430

[5] Ukil, A., Bandyopadhyay, S., Pal, A. (2014). IoT-privacy: To be private or not to be private. In Computer Communications Workshops (INFOCOM WKSHPS), Toronto, Canada, pp. 123-124. https://doi,org/10.1109/INFCOMW.2014.6849186

[6] Ali, M., Dhamotharan, R., Khan, E., Khan, S.U., Vasilakos, A.V., Li, K., Zomaya, A.Y. (2017). SeDaSC: Secure data sharing in clouds. IEEE Systems Journal, 11(2): 395-404. https://doi.org/10.1109/JSYST.2014.2379646

[7] Sasi, S.B., 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 Journal of Engineering, 4(3): 1-4. https://doi.org/10.9790/3021-04330104

[8] Yao, X.X., Chen, Z., Tian, Y. (2015). A lightweight attribute-based encryption scheme for the internet of things. Future Generation Computer Systems, 49: 104-112. https://doi.org/10.1016/j.future.2014.10.010

[9] Morales-Sandoval, M., Vega-Castillo, A.K., 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. Information Security Journal: A Global Perspective, 23(1-2): 22-31. https://doi.org/10.1080/19393555.2014.891282

[10] Thatmann, D., Zickau, S., Forster, A. (2015). Applying attribute-based encryption on publish subscribe messaging patterns for the internet of things. In International Conference on Data Science and Data Intensive Systems, pp. 556-563. https://doi.org/10.1109/DSDIS.2015.52

[11] Borgh, J., Ngai, E., Ohlman, B., Malik, A.M. (2016). Attribute-based encryption in systems with resource constrained devices in an information centric networking context. 2017 Global Internet of Things Summit (GIoTS), Geneva, pp. 1-6. https://doi.org/10.1109/GIOTS.2017.8016277

[12] Jin, Y., Tian, C., He, H., Wang, F. (2015). A secure and lightweight data access control scheme for mobile cloud computing. 2015 IEEE Fifth International Conference on Big Data and Cloud Computing, Dalian, pp. 172-179. https://doi.org/10.1109/BDCloud.2015.57

[13] Zhou, Z.B., Huang, D.J. (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), Las Vegas, USA, pp. 37-45.

[14] Wang, H., He, D.B., Shen, J., Zheng, Z.H., Zhao, C., Zhao, M.H. (2016). Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing. Soft Computing, 21: 7325-7335. https://doi.org/10.1007/s00500-016-2271-2

[15] Zhao, Z.Y., Wang, J.H. (2017). Verifiable outsourced ciphertext-policy attribute-based encryption for mobile cloud computing. KSII Transactions on Internet & Information Systems, 11(6): 3254-3272. https://doi.org/10.3837/tiis.2017.06.024

[16] Nguyen, K.T., Oualha, N., Laurent, M. (2017). Securely outsourcing the ciphertext-policy attribute-based encryption. World Wide Web, 21: 169-183. https://doi.org/10.1007/s11280-017-0473-x

[17] Cheung, L., Newport, C. (2007). Provably secure ciphertext policy ABE. In Proceedings of the 14th ACM conference on Computer and Communications Security, Auckland, New Zealand, pp. 456-465. https://doi.org/10.1145/1315245.1315302

[18] Emura, K., Miyaji, A., Omote, K., Nomura, A., Soshi, M. (2010). A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. International Journal of Applied Cryptography, 2(1): 46-59.

[19] Doshi, N., Jinwala D. (2011). Constant ciphertext length in CP-ABE. 2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011), Allahabad, India, pp. 451-456. https://doi.org/10.1109/ICCCT.2011.6075139

[20] Green, M., Hohenberger, S., Waters, B. (2011). Outsourcing the decryption of ABE ciphertexts. 20th USENIX conference on Security, p. 34.

[21] Somesh, S., Singh, N. (2018). Role based security for cloud based data with data reliability. Int J Res Eng IT Soc Sci, 7: 23-28.

[22] Bethencourt, J., Sahai, A., Waters, B. (2007). Ciphertext-policy attribute-based encryption. In 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, USA, pp. 321- 334. https://doi.org/10.1109/SP.2007.11

[23] Taha, M.M.B., Talhi, C., Ould-Slimane, H. (2019). Performance evaluation of CP-ABE schemes under constrained devices. Procedia Computer Science, 155: 425-432. https://doi.org/10.1016/j.procs.2019.08.05

[24] Taha, M.M.B., Talhi, C., Ould-Slimane, H., Alrabaee, S. (2020). TD-PSO: Task distribution approach based on particle swarm optimization for vehicular ad hoc network. Transactions on Emerging Telecommunications Technologies. https://doi.org/10.1002/ett.3860

[25] Taha, M.M.B., Talhi, C., Ould-Slimanec, H. (2019). A cluster of CP-ABE microservices for VANET. Procedia Computer Science, 155: 441-448. https://doi.org/10.1016/j.procs.2019.08.061