Efficient Ciphertext Policy Attribute-Based Encryption through Outsourcing with Checkability for the Internet of Things

Chih-Hung Wang[1], Ying-Jung Hsu[2]

[1,2]Computer Science and Information Engineering, National Chiayi University

Abstract**:** *Internet of Things (IoT) has been more and more popular in the recent years, this paper presents a novel approach to outsourcing the computation of the encryption with checkability in the ciphertext policy attribute-based encryption (CP-ABE) structure for the IoT. Because of the computation of the CP-ABE, the resource-constrained device in the IoT may be difficult to complete the computation. Through the way of splitting secret and sending the pairing computation to the assistant nodes, the resource-constrained device can check the returning results by the assistant nodes and generate the ciphertext with only several multiply operations. In the proposed scheme, the resource-constrained device can correctly complete the encryption through outsourcing without releasing any secrets.*

Keywords: *ciphertext policy attribute-based encryption, Internet of Things, encryption outsourcing, resource-constrained device.*

1. Introduction

In the recent years, there are lots of paper about ABE (attribute-based encryption). ABE, a novel scheme about access control, can use the attributes of the user to encrypt or decrypt message. It is divided into two categories: CP-ABE[2] and KP-ABE[3]. The difference between the two methods is that the CP-ABE uses the access structure to encrypt message and the attribute set to be the key generation; the KP-ABE uses the attribute set to encrypt message and the access structure to be the key generation.

Considering the cost of the ABE computation, lots of issues about outsourcing or cooperation were proposed. The methods assist the resource-constrained device to run encryption or decryption under ABE scheme. In 2011, Waters et al. [7] present a CPA-secure CP-ABE outsourcing scheme. In 2014, Asim et al. [1] presented a complete method to outsource data; the proxy helps the user to handle data without pairing computation and helps the low-end mobile device running the encryption and decryption. Touati et al. [6] further developed a method of cooperative CP-ABE (C-CP-ABE) that can help the nodes in the IoT to be able to run encryption of the CP-ABE.

In addition to helping nodes to run encryption in the IoT environment, this paper also provide the property that the nodes can check the correctness of the outsourcing data. Apart from that, the proposed scheme combine the decryption outsourcing of the CP-ABE. For instance, imaging there are lots of embedded devices for monitoring the patient's physiological situation in the hospital all day long and the monitoring data may include the patient's privacy. Therefore, the embedded devices encrypt the data and transmit the ciphertext to the medical system, and then stores the data in the system database. The doctor who has been authorized can legally view the patient physiological situation through the doctor's computer or the personal cell phone.

2. Preliminary

Before explaining how the proposed method works, this section simply introduces some foundation background knowledge about cryptographic techniques and defines the notations used in this paper.

2.1. Bilinear Maps

Let e be a bilinear map $e: G \times G \rightarrow G_T$, where G and G_T are two multiplicative cyclic groups of prime order p, and g is a generator. The bilinear map e has the following properties:

- ***Bilinearity***: for all $u, v \in G$ and $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{a \cdot b}$.
- ***Nom-degeneracy***: $e(g, g) \neq 1$.

We define that G is a bilinear group if the group operation in G and the bilinear map e:G×G→G_T both compute efficiently.

2.2. CP-ABE Model

In the CP-ABE [2] model, the message is encrypted according to the access structure and the private key is identified with a set *S* of descriptive attributes. The message is allowed to be encrypted when it satisfy the access tree *AT*. The non-leaf node is the threshold gate and the leaf node is the attribute in the *AT*. Talking about *AT*, we have to mention the following functions:

Parent(x): return the parent of the node *x* in the access tree *AT*.

Att(x): return the attributes associate with the node *x* in the access tree *AT*.

Index(x): return the order of the node *x* in the access tree *AT*.

2.3. Outsourcing the decryption of CP-ABE

Outsourcing the decryption of CP-ABE model [4] mainly has five algorithms. In the following, we describe how the algorithms work respectively in detail:

- ***Setup(λ,U)***: It takes as input the security parameter λ and the universe description *U*. Outputs the master secrect key *MSK* and the public key *PK*.
- ***Encrypt(PK,M,T)***: Inputs the public key *PK*, the message *M* and a LSSS access structure *T*. Outputs the ciphertext *CT* with the description *T*.
- ***KeyGen(MSK,S)***: Inputs *MSK* and a set of attributes *S*. Outputs the transformation key *TK* and the private key *SK*.
- ***Transform(TK,CT)***: Takes as input the transformation key *TK* for a set of attributes *S* and the ciphertext *CT* for the LSSS access structure *T*. If *S* does not satisfy *T*, aborting algorithm; otherwise, outputs the partially decryption ciphertext *CT'*.
- ***Decrypt(SK,CT)***: Inputs the private key *SK* and the ciphertext *CT*. If *CT* is not the partially decryption ciphertext format, runs the ***Transform*** algorithm first. Otherwise, outputs the plaintext *PT*.

Note that the key of this method is outsourcing the ciphertext. First, the ciphertext is transferred to the El Gamal format one, and then the user can run the ***Decrypt*** algorithm by computing simple multiplications and get the plaintext *PT*.

2.4. Batch exponentiations outsourcing with fixed base-variable

In this part, assume that the device *D* wants to outsource the exponentiation computation to the remote server RS, the followings would explain how to work:

- ***Initialize***: *D* wants to compute $g^{x_1}, g^{x_2}, \ldots, g^{x_i}$. In the first, *D* generates *n* random numbers $r_1, \ldots, r_n$, where *m* of which could be summed to x_j, where *m* is set close to (*n*-*i*)/2 generally, $j \in [1, i]$. Second, *D* sets $c_k = x_k - \sum_{j=1}^{m-1} base_j$ that guaranteed there must be the sum to be x_k, where $k \in [1, i]$, $base_j \in Z_p, j \in [1, m]$. Finally, *D* generates a new set $S=(s_1, \ldots, s_{i+m+n})$ with the random order, which composed of $r_j, c_k, base_l$, where $j \in [1, n], k \in [1, i]$, $l \in [1, m]$; and a list *L* to record the result.
- ***Send***: *D* sends *S* and *g* to the independent remote servers RS_1 and RS_2.
- ***Response***: RS_1 and RS_2 return the corresponding result $(g^{s_1}, \ldots, g^{s_{i+m+n}})$ one by one to *D*, and *D* would compare the result. If the comparison result is different, it means at least one of them is dishonest.
- ***Compute***: If RS_1 or RS_2 is dishonest, aborting protocol. Otherwise, for $j \in [1, i+m+n]$:

$$S_{base} = S_{base} \cdot s_j, \qquad S_k = S_k \qquad \textbf{if } s_j \in base_l$$

$S_{base} = S_{base}, \quad S_k = S_k \cdot s_j \quad$ ***if*** $s_j \in c_k$

$S_{base} = S_{base}, \quad S_k = S_k \quad$ ***otherwise***

At the end of the ***Compute***, *D* completes the computation: $g^{xj}=S_{base} \cdot S_j$, where $j \in [1, i]$.

2.5. Notations

The notations of the following table are used in this paper and their descriptions are explained.

TABLE I: Notation Table

Notation	Description	Notation	Description
D	Device	*PK*	Public key
P_i	Assistant node *i*	*CT*	Ciphertext
RS.	Remote server	*PT*	Plaintext
(*M*,*ρ*)	LSSS access structure	$r_1, \ldots, r_l$	Random number
M	A *l***n* matrix	$y_2, \ldots, y_b$	Random number
ρ	A function associated with the rows of *M* to attribute	M_i	The vector relative to the *i*-th row of the Matrix *M*
s	Encryption exponent	Q_i	The subset of λ_i, where *i=1 to l*
$F()$	a hash function map $\{0,1\}^*$ to G^a	R_i	The subset of r_i, where *i=1 to l*
$OutPack_i$	The subset of set S, Q_i, R_i, where *i=1 to l*	S	The subset of s
L_i	A list of the orders of data shares to P_i related to the computing operation and indicating which of them are random numbers		

3. Proposed Solution of Checkability

To run the CP-ABE scheme in the IoT environment, we have to outsource the costly computations to support the node execution successfully. The main idea of this paper is outsourcing all the exponentiation computations in the node encryption phase to the assistant nodes, such that the node can just compute the multiplications to complete the encryption algorithm.

3.1. Network Model

In the encryption phase, the IoT environment has lots of devices connected to each other with such as wireless sensor networks (WSNs) , the embedded computing nodes are expected to be a resource-constrained devices, the unconstrained devices can cooperatively complete the computations, and a server is needed to storage and analyze the data which these embedded devices monitor. In the following, we classify the nodes into three classes [6]:

- **Resource-constrained devices**: the device has the ability of weak computations and low storages (such as the sensor, embedded computing device.) The resource-constrained devices are assumed to be difficult to implement the CP-ABE pairing computation, and never deliver sensitive data.
- **Unconstrained devices**: the device has stronger computation and storage ability than the resource-constrained devices, and it is also called "assistant node". The unconstrained devices is assumed to be dishonest (may return the error computation result to save the resource or other target.)
- **Remote server**: the device has the strong computation ability and large amount storage, and it can cooperatively compute the encryption algorithm and storage ciphertext without learning any sensitive data.

In the decryption phase, the user terminal requests the ciphertext from the remote server and there is a proxy to assist the decryption without release any sensitive data.

3.2. Assumption

The assumptions in the paper are listed below.

- There are at least two unconstrained devices to be the neighborhood of any resource-constrained devices.
- It always shares the pairwise keys between device A and device B when A want to send message to B. The keys have to be generated before, like specific bootstrapping phase or initialization phase.

- It is assumed that the remote server is trustworthy and is responsible for handling the outsourcing encryption and checking the correctness of the data from the assistant nodes.

3.3. How it works

The encryption algorithm is performed by the resource-constrain device *D*, and the inputs are the public key *PK*, plaintext *PT* and the LSSS access structure (M,ρ), where *M* is a $l \times n$ matrix and ρ is a function associate with the rows of *M* to attribute. Then device D outsources the computation to *N* assistant nodes P_a (where $a = 1 \sim N$), then output the ciphertext *CT* along with a description of LSSS access structure (M,ρ) to the remote server *RS*.

We modify the encryption phase from the method of CP-ABE [4], the other phase algorithms (Setup, KeyGen, Transform and Decrypt) are the same as the original ones in the CP-ABE (see Fig 1). In the following, we divide the encryption algorithm into several sub-phases to describe algorithm in detail:

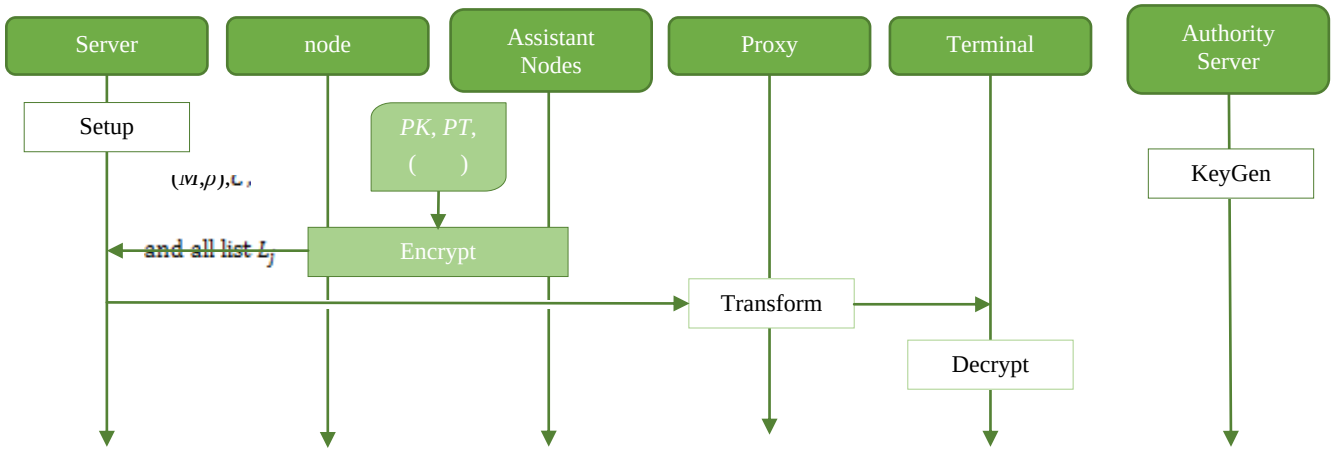

Fig 1: The outsourcing process in IoT environment

- ***Phase 1***: The resource constrained device D has to encrypt the Plaintext PT. Device D chooses *m* neighbours (where *m* is bigger than 2) unconstrained devices to be the assistant nodes to outsource some exponentiation operations in the encrypt phase.
- ***Phase 2***: The device *D* defines LSSS access structure (M,ρ) according to the attribute of the *PT*. It randomly chooses the numbers $s, y_2, \ldots, y_b \in Z_p^b$ to be a new vector $\vec{v}$ (where *s* is the encryption exponent, and these values will be used for the encryption) and the random number $r_1, r_2, \ldots, r_l \in Z_p$. The device *D* calculates $\lambda_i = \vec{v} \cdot M_i$, where $1 \leqq i \leqq l$ and M_i is the vector related to the *i*-th row of the Matrix *M*.
- ***Phase 3***: The device *D* splits *s* by the algorithm ***Split*** to generate the set $S = \{s_1, s_2, \ldots, s_x\}$ and a function L_0 where $\sum_{L_0(s_i) \geqq 0} s_i = s$; ***Split*** λ_i, r_i for *i=1 to l*, to generate the set Q_i, R_i and list L_i to record the order. It sets the security degree as *T* by the user and picks out subset from the sets S, Q_i, R_i into *N* parts: $\{OutPack_1, \ldots, OutPack_N\}$.
- ***Phase 4***: For each assistant node P_a, the device *D* sends the $OutPack_a$ and the F(*ρ*(*j*)) where *j=1 ~ l*, through the secure channel that means these data are encrypted by the share key K_{DP_a}.
- ***Phase 5***: Each assistant node P_a calculates the $e(g,g)^{\alpha s_i}$, g^{s_i} and the $g^{\alpha \lambda_j}$, g^{r_j}, $F(\rho(j))^{-r_j}$ where $s_i \in OutPack_a$, *j=1 to l*, and $F()$ is a hash function: $\{0,1\}^*$ to G^a; then returns the result of the $e(g,g)^{\alpha s_i}$ where $s_i \in OutPack_a$ to *D* by the secure channel encrypted by the share key K_{DP_i}. P_a sends the remaining result to the remote server *RS*.
- ***Phase 6***: After receiving the result from the assistant nodes, the device *D* running ***CheckResult*** algorithm with the list. If the algorithm returns error, *D* stops execution. Otherwise, it computes $\vec{C} = PT$

$\prod_{L_0(s_i)\geqq 0} e(g,g)^{\alpha s_i}$, and sends (M,ρ), $\tilde{C}$, and all list L_j to the remote server RS by the encrypted channel.

- ***Phase 7***: After receiving all the results from the assistant nodes and the device *D*, the remote server *RS* runs ***CheckResult*** algorithm with the list from the device *D*. If the algorithm returns error, *RS* outputs the error message and break the algorithm; otherwise, *RS* computes the ciphertext *CT*:

  $\tilde{C},\ C = g^s = \prod_{L_0(s_i)\geqq 0} g^{s_i}$; for $i=1$ to l, $(C_i = g^{\alpha\lambda_i} * F(\rho(i))^{-r_i}, D_i = g^{r_i})$.

  RS stores the ciphertext as the *CT* and the access structure (M,ρ).

ALGORITHM I: *Split*

Algorithm I *Split*: Separate the input numbers $q_1, q_2, \ldots, q_t$ into x parts $p_1, p_2, \ldots, p_x$, and generate a function *L* to indicate whether p_i is a random number.

Input: The set $H=(q_1, q_2, \ldots, q_t)$ and the number *x* to separate the set *H* into a new set *S* with *x* elements.

Output: A set of number *S* and the function *L*.

1: Generate two set of random numbers $rand_j$, $base_j$, where $j \in [1, (x-t)/2]$.

2: Generate $c_k = q_k - \sum_{j=1}^{(x-t)/2} base_j, k \in [1,t]$.

3: Combine $rand_j$, $base_j$, c_k where $j \in [1,(x-t)/2], k \in [1,t]$. Generate a set *S'*:

$$S' = \{rand_1, \ldots, rand_{(x-t)/2}, base_1, \ldots, base_{(x-t)/2}, c_1, \ldots, c_t\}$$

4: Shuffle the set *S'* to be a new set *S* and generate a function *L*:

$L(p_i) > 0$, if $p_i \in \{c_1, \ldots, c_t\}$; the value of $L(p_i)$ represents the sequence number of p_i in this set

$L(p_i) = 0$, if $p_i \in \{base_1, \ldots, base_{(x-t)/2}\}$

$L(p_i) < 0$, if $p_i \in \{rand_1, \ldots, rand_{(x-t)/2}\}$

5: Return the set *S* and the function *L*.

ALGORITHM II: *PickOut*

Algorithm II *PickOut*: Divide set *S* into *p* subset; the security degree T output $Q_1, Q_2, \ldots, Q_u$

Input: Two values, the set *S* and the number *p* which is going to separate set *S* into *p* subset.(*p* is the number of assistant nodes)

Output: subset $Q_1, Q_2, \ldots, Q_u$ and the record function $L_1, L_2, \ldots, L_u$.

1: Set |S| is the number of the element of the set *S*, $w = \frac{|S|}{p}$.

2: Set the number of the subset element $\#sub = \frac{|S| * T}{p}$

3: //Generate subset Q_u:

4: **for** *temp1*= 1 to *u* **do**

5: //ensure that all answers would be calculated more than *T* times and no two different assistant nodes would compute the same subset.

6: **for** *temp2*= 1 to *#sub* **do**

7: Pick the {(*temp2*-1)+w}-th of the set *S* to be the element of the set Q_{temp1}.

8: Record in the List L_{temp1}

9: **end for**

10: Shuffle the set Q_{temp1} and record the original order in the List L_{temp1}

11: **end for**

12: **return** subset $Q_1, Q_2, \ldots, Q_u$ with the record function $L_1, L_2, \ldots, L_u$

ALGORITHM III: *CheckResult*

Algorithm III *CheckResult*: When receiving the computation results, check the correctness by comparing the answers with the same calculation operation but from different assistant nodes.

Input: Several sets $S_1, S_2, \ldots, S_i$, the function s $L_1, L_2, \ldots, L_i$

Output: return error or true.

1: **for** the inputs from different assistant nodes but with same calculation operation **do**
2: **if** the computation results are different **then**
3: **return** error
4: **end if**
5: **end for**
6: **return** true

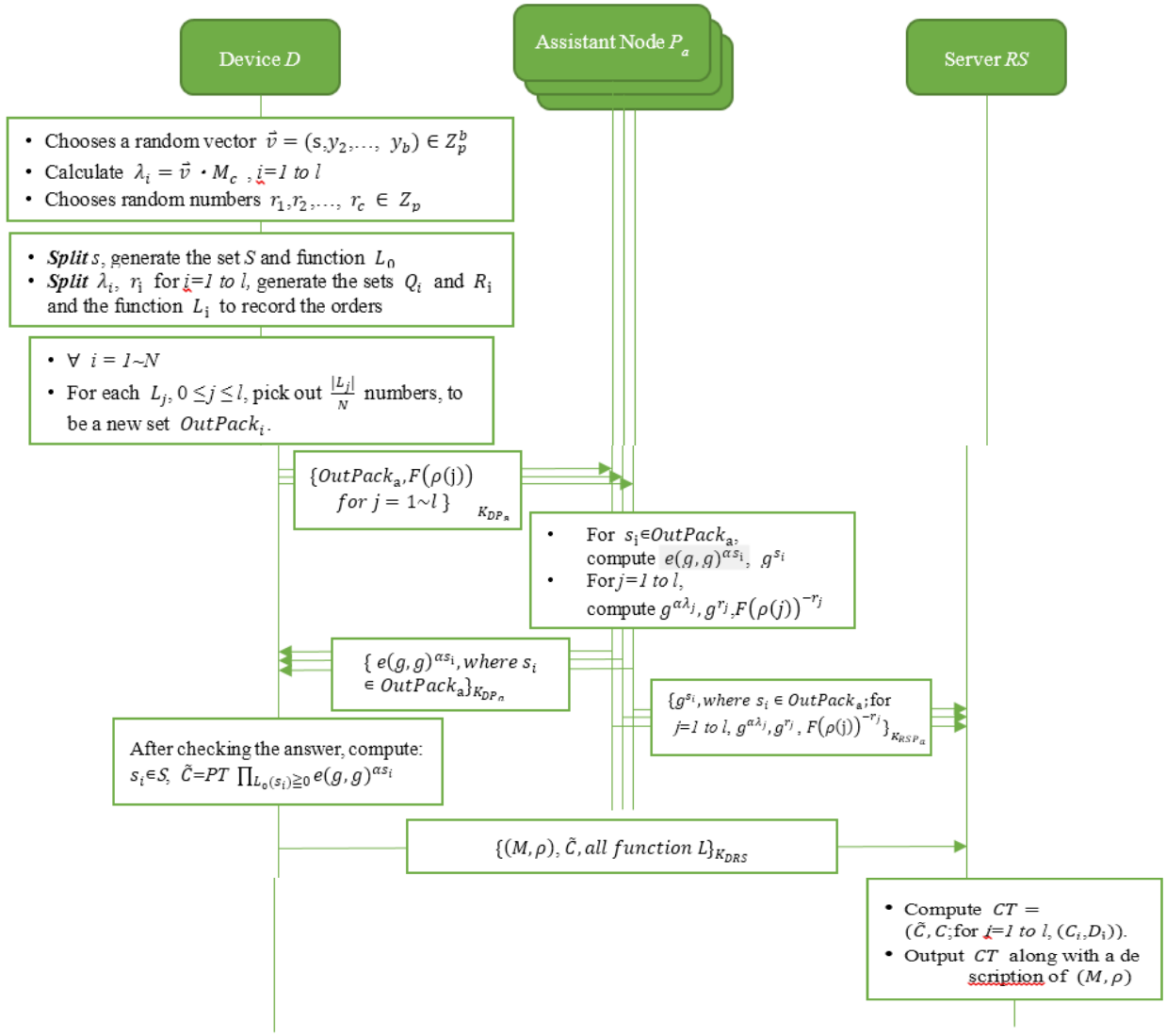

Fig 2: The diagram of the proposed protocol

4. Discussion

In this section we discuss the security and the performance analyses.

4.1. Security Analysis

4.1.1. Security degree

The CP-ABE is secure when the number of collusion nodes is less then *T*, the parameter inputted by the user. In this paper, we request that *T* must be larger than 2.

4.1.2. Computational accuracy

For example, when the user set $T \geq 2$,it means the computation would be check two times (or more) every two computation parts (with the same computation operation). In the condition of no collusion node exists, the probability of cheating by a malicious node without being detected is negligible (near to zero) (we ignore the probability of two nodes returning the same wrong answer without collusion).

4.2. Performance Analysis

4.2.1. Computation cost

Because of outsourcing the exponentiation computation to the assistant nodes, the device D has no exponentiation, but it has to compare the outsourcing computation result $|S| \times T$ times and do the multiplication |S| times.

4.2.2. Communication cost

The communication times of the device *D* is (the number of the assistant node) $\times 2+1$.

5. Conclusions and Future Works

This paper presented a new approach to outsource the encryptions within the IoT environment. The advantage of the new approach is that it has the ability to check the outsourcing results without heavy computational cost. The decryption part can follow the methods of [4, 7] to achieve both of encryption and decryption outsourcing in the CP-ABE for IoT environment.

In the future, it would be interesting to find out the best solutions of some parameters like security degree *T*, the number of split part, etc. by the simulation experiments to replace the heuristic inputs by the users.

6. Acknowledgements

This research is supported by Ministry of Science and Technology, Taiwan under the grants: MOST 105-2221-E-415-015 and MOST 106-2221-E-415-003-MY2.

7. References

[1] M. Asim, M. Petkovic, and T. Ignatenko. "Attribute-based Encryption with encryption and decryption outsourcing," *12th Australian Information Security Management Conference* (Perth, Australia, December 1-3, 2014), ECU Security Research Institute, pp. 1-28, 2014.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," 2007 IEEE Symposium on Security and Privacy (SP'07), IEEE Press, 2007.

https://doi.org/10.1109/SP.2007.11

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters. "Attribute-based encryption for fine-grained access control of encrypted data, " In Proc. the 13th ACM conference on Computer and communications security, ACM, 2006.

https://doi.org/10.1145/1180405.1180418

[4] M. Green, S. Hohenberger, and B. Waters. "Outsourcing the decryption of ABE ciphertexts," in Proc. the 20th *USENIX conference on Security, 2011.*

[5] X. Ma, J. Li, and F. Zhang, "Efficient and secure batch exponentiations outsourcing in cloud computing," 2012 4th International Conference on Intelligent Networking and Collaborative Systems (INCoS), IEEE Press, 2012.

https://doi.org/10.1109/iNCoS.2012.31

[6] L. Touati, Y. Challal, and A. Bouabdallah, "C-CP-ABE: Cooperative ciphertext policy attribute-based encryption for the Internet of Things," 2014 International Conference on Advanced Networking Distributed Systems and Applications (INDS), IEEE Press, 2014.

https://doi.org/10.1109/INDS.2014.19

[7] B. Waters, "Ciphertext-policy attribute-Based encryption: An expressive, efficient, and provably secure realization," Public Key Cryptography, vol. 6571, pp. 53-70, 2011.

https://doi.org/10.1007/978-3-642-19379-8_4