International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT206448

A Secure and Fine-Grained Big Data Access Control Scheme for Cloud-Based Services

Nisha J William, Nisha O S
Department of Computer Science and Engineering, Lourdes Matha College of Science and Technology, Kuttichal, Kerala, India

Article Info
Volume 6, Issue 4
Page Number: 254-262
Publication Issue :
July-August-2020

Article History
Accepted : 20 July 2020
Published : 27 July 2020

ABSTRACT

Cloud computing is the delivery of computing services including servers, storage, databases, networking, software, analytics, and intelligence over the Internet. Nowadays, access control is one of the most critical problems with cloud computing. Ciphertext-Policy Attribute Based Encryption (CP-ABE) is a promising encryption technique that enables end-users to encrypt their data under the access policies defined over some attributes of data consumers and only allows data consumers whose attributes satisfy the access policies to decrypt the data. In CP-ABE, the access policy is attached to the ciphertext in plaintext form, which may also leak some private information about end-users. Existing methods only partially hide the attribute values in the access policies, while the attribute names are still unprotected. This paper proposes an efficient and fine-grained big data access control scheme with privacy-preserving policy. Specifically, it hides the whole attribute (rather than only its values) in the access policies. To assist data decryption, it designs an algorithm called Attribute Bloom Filter to evaluate whether an attribute is in the access policy and locate the exact position in the access policy if it is in the access policy. The paper also deals with offline attribute guessing attack. Security analysis and performance evaluation show that this scheme can preserve the privacy from any LSSS access policy without employing much overhead.

Keywords : Cloud Computing, Access Control, Attribute, LSSS Access Structure, Attribute Bloom Filter, Offline Attribute guessing attack

I. INTRODUCTION

In the era of big data, a huge amount of data can be generated quickly from various sources (e.g., smart phones, sensors, machines, social networks, etc.). Towards these big data, conventional computer systems are not competent to store and process these data. Due to the flexible and elastic computing resources, cloud computing is a natural fit for storing and processing big data. When outsourcing data into the cloud, end-users lose the physical control of their data. Moreover, cloud service providers are not fully-trusted by end-users, which make the access control more challenging. Thus, end-users may worry that the cloud server may make wrong access decision

intentionally or unintentionally, and disclose their data to some unauthorized users.

Existing access control methods all suffer a problem that: the access policy may leak privacy. This is because the access policy is associated with the encrypted data in plaintext form. From the plaintext of access policy, the adversaries may obtain some privacy information about the end-user. To prevent the privacy leakage from the access policy, a straightforward method is to hide the attributes in the access policy. However, when the attributes are hidden, not only the unauthorized users but also the authorized users cannot know which attributes are involved in the access policy, which makes the decryption a challenging problem.

This scheme aims to hide the whole attribute instead of only partially hiding the attribute values. The basic idea is to express the access policy in LSSS access structure (M,ρ) where M is a policy matrix and ρ matches each row M_i of the matrix M to an attribute, and hide the attributes by simply removing the attribute matching function ρ. Without the attribute matching function ρ, it is necessary to design an attribute localization algorithm to evaluate whether an attribute is in the access policy and if so find the correct position in the access policy. An algorithm Attribute Bloom Filter to locate the attributes to the anonymous access policy is further build, which can save a lot of storage overhead and computation cost especially for large attribute universe.

This system also deals with the offline attribute guessing attack which checks the guessing of "attribute strings". A new way of index generation mechanism is proposed, in which data user assigns data owner and generates index with the public key of data owner. Because of lacking data owner's secret key, fake index used to implement keyword guessing attack by adversary is easy to distinguish.

The contributions are summarized as follows:

i. This paper proposes a secure and fine-gained big data access control scheme for cloud based services, where the whole attributes are hidden in the access policy rather than only the values of the attributes.
ii. It also designs an algorithm called Attribute Bloom Filter to evaluate whether an attribute is in the access policy and locate the exact position in the access policy if it is in the access policy.
iii. It further deals with offline attribute guessing attack that checks the guessing of attribute string.

II. SYSTEM MODEL

We Consider The Big Data Access Control System, As Shown In Fig. 1. The System Consists Of Five Entities, Namely Cloud Servers, Attribute Authority, End-Users, And Data Consumers.

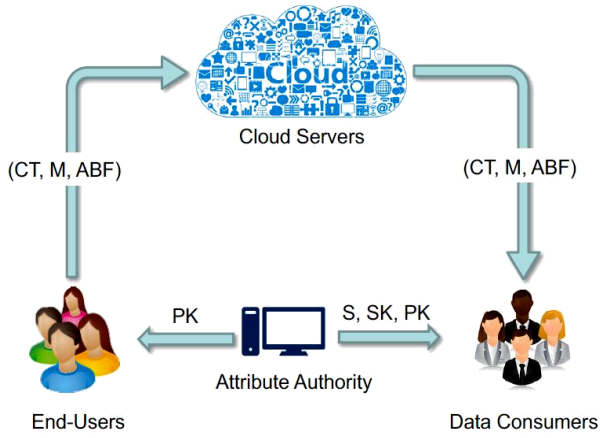

Fig. 1 System Model

- Cloud Servers: Cloud Servers Are Employed To Store, Share And Process Big Data In The System. The Cloud Servers Are Managed By Cloud Service Providers, Who Are Not In The Same Trust Domain As End-Users. Thus, Cloud Servers Cannot Be Trusted By End-Users To Enforce The Access Policy And Make Access Decisions.
- Attribute Authority: The Attribute Authority Manages All The Attributes In The System And Assigns Attributes Chosen From The Attribute Space To End-Users. It Is Also A Key Generation Centre, Where The Public Parameters Are Generated. It Also Grants Different Access Privileges To End-Users By Issuing Secret Keys According To Their Attributes. The Attribute

Authority Is Assumed To Be Fully Trusted In The System.

- End-User: End-Users Are The Data Owners/Producers Who Outsource Their Data Into The Cloud. They Also Would Like To Control The Access Of Their Data By Encrypting The Data With Cp-Abe. End-Users Are Assumed To Be Honest In The System.
- Data Consumers: Data Consumers Request The Data From Cloud Servers. Only When Their Attributes Can Satisfy The Access Policies Of The Data, Data Consumers Can Decrypt The Data. However, Data Consumers May Try To Collude Together To Access Some Data That Are Not Accessible Individually.

III. Proposed System

This big data access control scheme consists of the following algorithms: Setup, KeyGen, Encrypt, and Decrypt.

A. **Setup** $(1^{\lambda}) \rightarrow (PK,MSK)$

During the system setup phase, the attribute authority runs the Setup algorithm. Let U denote the attribute space in the system. Let G and G_T be cyclic multiplicative groups of prime order p, and $\hat{e}:G\times G \rightarrow G_T$ be a bilinear map. Let L_{att} be the maximum bit length of attributes in the system. Let L_{rownum} be the maximum bit length of the row numbers of access matrix. Let L_{ABF} be the size of bit array of the Attribute Bloom Filter. Let k be the number of hash functions associated with the ABF.

The attribute authority randomly chooses a generator $g \in G$, $\alpha,a \in Z^*_p$, and U = |U| random group elements $h_1,h_2,...,h_U \in G$. It also generates k hash functions $H_1(),H_2(),...,H_k()$ that maps an element to a position in the range of $[1,L_{ABF}]$. The public key is published as

$PK = < g, \hat{e}(g,g)^{\alpha}, g^a, L_{att}, L_{rownum}, L_{ABF}, h_1, h_2,\ldots, h_U, H_1(), H_2(),\ldots, K_k() >$

The master secret key is set as $MSK = g^{\alpha}$.

KeyGen $(PK,MSK,S) \rightarrow SK$

Each data consumer should register and authenticate to the attribute authority. If the data consumer is not legal, it aborts. Otherwise, the attribute authority will evaluate the role of the data consumer in the system and assign a set of attributes S chosen from the attribute space U to this data consumer. It computes

$K = g^{\alpha}g^{at}, L = g^t, \{K_x = h^t_x\}_{x\in S}$

where $t \in Z^*_p$ is chosen at random. Finally, the secret key is set as $SK = < K, L, \{K_x\}_{x\in S}, S >$

Encrypt $(PK,m,(M,\rho)) \rightarrow (CT,ABF)$

Before outsourcing data into the cloud, end-users encrypt the data by running the Encrypt algorithm. The data encryption algorithms contains: data encryption subroutine *Enc* and Attribute Bloom Filter building subroutine *ABFBuild*.

Enc $(PK,m,(M,\rho)) \rightarrow CT$: The data encryption subroutine takes as inputs the public key PK, the message m and access structure (M,ρ). As shown in Fig 2, M is an $l\times n$ access matrix and the injective function ρ maps rows of M to attributes. The algorithm first chooses an encryption secret $s \in Z^*_p$ randomly and then selects a random vector $\vec{v} = (s,y_2,...,y_n)$, where $y_2,...,y_n$ are used to share the encryption secret s. For i=1,...,l, it calculates $\lambda_i = M_i \cdot \vec{v}$, where M_i is the vector corresponding to the i-th row of M. Then, it outputs the ciphertext as

$CT = < C=m\hat{e}(g,g)^{\alpha s}, C'=g^s, \{C_i=g^{a\lambda_i}h^{-s}_{\rho(i)}\}_{i=1,\ldots,l} >$

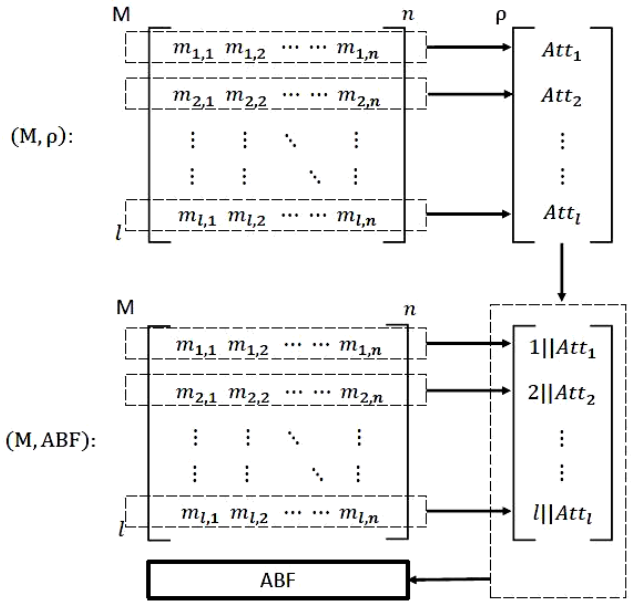

Fig. 2 The LSSS Access Policy and Attribute Bloom Filter

ABFBuild (M,ρ) → *ABF* : The ABF building subroutine takes as input the access policy (M,ρ). It outputs the Attribute Bloom Filter *ABF*.

Algorithm 1 ABFBuild

Input: An LSSS access policy (M,ρ), λ, L_{ABF}

Input: k hash functions $\{H_1(),\ldots,H_k()\}$

Output: *ABF*

1: Generate an element set S_e from the access policy (M,ρ)
2: *ABF* = new L_{ABF} element array of bit strings
3: **for** $i = 0$ to $L_{ABF} - 1$ **do**
4: *ABF*[i] = *NULL* // *Initialize the ABF with "NULL"*
5: **for** each element $e = i||att_e \in S_e$ **do**
6: *emptyPos* = -1, *finalShare* = x
7: **for** $i = 0$ to $k - 1$ **do**
8: $j = H_{i+1}(att_e)$ // *get the index of the position*
9: **if** *ABF*[j] == *NULL* **then**
10: **if** *emptyPos* == -1 **then**
11: // *reserve this position for the finalShare*
12: *emptyPos* = j
13: **else** // *generate a new share*
14: generate a random string $r_{j,e}$ with λ bits
15: *ABF*[j] = $r_{j,e}$
16: *finalShare* = *finalShare* ⊕ *ABF*[j]
17: **else** // *reuse an existing share*
18: *finalShare* = *finalShare* ⊕ *ABF*[j]
19: *ABF*[*emptyPos*] = *finalShare*
20: **for** $i = 0$ to $L_{ABF} - 1$ **do**
21: **if** *ABF*[i] == *NULL* **then**
22: // *fill the empty position with random strings*
23: generate a random string r_i with λ bits
24: *ABF*[i] = r_i

Search (*PP*,*SK*,SK_{uid}) → *SR*

Offline Index (*PP*) → *IX*: The offline index algorithm is run by End User, and in this stage the associated access structure and keyword is unknown. The input of this algorithm is public parameters, and the output is intermediate index IX.

End User does pre-computing when the associated keywords and access structure is unknown. It picks $s \in Z_p$, then computes $C = e(g_1,g_2)^{\alpha s}$, $C' = g^s{}_2$, $C'_0 = g_2^{sid_S}$. Let $K_{att} = U$. For attributes $i \in K_{att}$, chooses random $r_i \in Z_p$, computes $C'_i = h_{2,i}{}^{r_i}$, $D'_i = g_2^{r_i}$, and remove i from K_{att}. Output the Intermediate Index $IX = \{s, C, C', C'_0, \{C'_i, D'_i\}_{i \in U}\}$.

Offline Trapdoor (*PP*,*SK*,SK_{uid}) → *IT* : Offline trapdoor algorithm is run by Data Consumer, and in this stage the associated keyword and the owner of file set to be searched is unknown. The input of this algorithm is public parameters, user's secret key and its private key, and the output is intermediate trapdoor IT.

Data Consumer does pre-computing when the search task is unknown. It computes $K' = K_x{}^{\frac{1}{uid}}$, $L_0' = g_1{}^{\frac{1}{uid}}$, $L_1' = L_1{}^{\frac{1}{uid}}$, and saves it as Intermediate Trapdoor.

Search (*Index,Trapdoor*) → *SR* : Search algorithm is run by Cloud Services Provider. It takes the Index and Trapdoor as input and outputs the search result. Cloud service provider computes $\{w_i \in Z_p\}_{i \in I}$ satisfies $\Sigma_{i \in I} w_i \lambda_i = s$, and let $I \in \{1,2,\ldots,l\}$ be defined as $I = \{I : \rho(i) \in S\}$, then computes $T_1 = \Pi_{i \in I} [e(C_i, L'_i) e(D_i, K'_{\rho(i)})]^{wx}$, $T_2 = (K', C)$, $T_3 = C_e(C_0, L'_0)$, and test whether $T_2 = T_1 T_3$.

Decrypt (*M,ABF,PK,SK,CT*) → m
The decryption algorithm consists of two subroutines: *ABFQuery* and *Dec*.

ABFQuery (*S,ABF,PK*) → ρ' : The ABF query algorithm takes as inputs the attribute set S, the Attribute Bloom Filter ABF and the public key PK. It outputs a reconstructed attribute mapping ρ' = {(*rownum, att*)}s, which shows the corresponding row number in the access matrix M for all the attributes *att* ∈ *S*.

Algorithm 2 ABFQuery

Input: An Attribute Bloom Filter *ABF*, a set of attributes *S*
Input: *k* hash functions $\{H_1(),\ldots,H_k()\}$
Input: Maximum attribute string length L_{att}
Input: Maximum row number string length L_{rownum}
Output: $\rho' = \{(rownum,att)\}_{att \in S}$
1: **for** each *att* ∈ *S* **do**
2: $ReStr = \{0\}^{\lambda}$ // *initialize the reconstructed string*
3: **for** *i*= 0 to *k* − 1 **do**
4: $j = H_{i+1}(att)$ //*get the index of the position*
5: $ReStr = ReStr \oplus ABF[j]$
6: $att_eStr = LSB_{Latt}(ReStr)$
7: // *get* L_{att} *least significant bits*
8: $att_e = RmLeadingZeroBits(att_eStr)$
9: // *remove all the leading zero bits*
10: **if** $att_e == att$ **then**
11: $rownumStr = MSB_{Lrownum}(ReStr)$
12: // *get* L_{rownum} *most significant bits*
13: $rownum = RmLeadingZeroBits(rownumStr)$
14: // *remove all the leading zero bits*
15: Add (*rownum,att*) into ρ'

Dec (*SK,CT,*(M, ρ')) → *m* or ⊥ : The data decryption algorithm takes as inputs the secret key *SK*, the ciphertext *CT* as well as the access matrix M and the reconstructed attribute mapping ρ'. If the attributes can satisfy the access policy, it outputs the message m. Otherwise, it outputs ⊥.

IV. Experiment and Result

The big data access control system can be divided into 5 modules:

1. End User Application
2. End User Encryption
3. Cloud Service Provider
4. Data Consumer Request
5. Retrieve Data

A. *End User Application*
First of all end user will register into the cloud server as shown in fig. 3. Then he can login into the system and select data they need to share as in fig. 4. Authorization permission to access data will be given to some of the consumers so that authorized users can only access the data. Details of data that need to be shared in the system will be known to the end user.

B. *End User Encryption*
Data will be transferred into the Cloud Server only after it is encrypted and converted into non plaintext form. The key for encryption purpose will be provided by the Attribute Authority as in fig. 5. Attribute Authority will receive request from end user for encryption key and it will generate a key.

Using the key received from Attribute Authority end user can encrypt the data which is shown in fig. 6.

Fig. 3 End user Login

Fig. 4 Select data

Fig. 5 Generate key and send to End user

Fig. 6 Encrypt data

C. *Cloud Server*

End user will transfer encrypted data into the Cloud Server. Data Consumers will login into the system and requests for data from the cloud. Requested data will be granted to authorized users or data consumers. Here cloud server is acting as a storage place.

D. *Data Consumer Request*

Data consumer has to register and login into the cloud. He can send requests for data that are available in cloud. The cloud server will send the request to cloud service provider. Requests for the data accesses will be processed by cloud service provider.

E. *Retrieve Data*

Cloud service provider receives request for data from the cloud consumer. It will search for relevant data to the query from the cloud. The data available in the cloud will be transferred to the consumer. After receiving data, the consumer will request for key from attribute authority to decrypt the data. Attribute authority will generate key as shown in fig. 7 and will send it to the data consumer. The data consumer can then decrypt the data using received key which is shown in fig. 8.

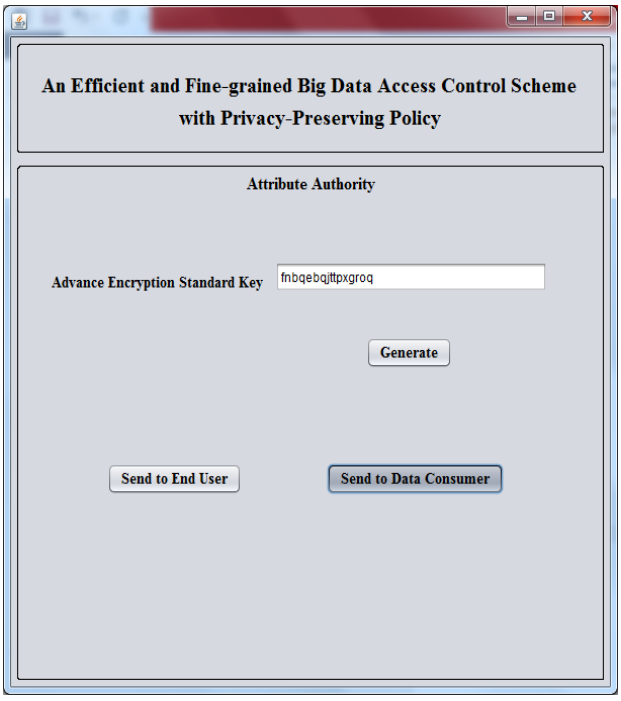

Fig. 7 Generate key and send to Data consumer

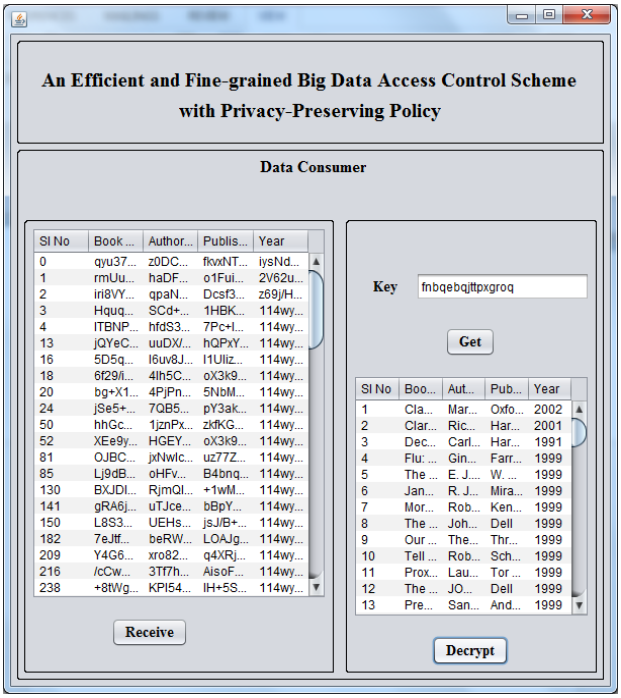

Fig. 8 Decrypt data

V. CONCLUSION

This work has proposed an efficient and fine-grained data access control scheme for big data, where the access policy will not leak any privacy information. Different from the existing methods which only partially hide the attribute values in the access policies, this method can hide the whole attribute (rather than only its values) in the access policies. However, this may lead to great challenges and difficulties for legal data consumers to decrypt data. To cope with this problem, we have designed an attribute localization algorithm called Attribute Bloom Filter to evaluate whether an attribute is present in the access policy. The system also deals with offline attribute guessing attack which is guessing of "attribute strings" by continually querying the ABF. The scheme is selectively secure against chosen plaintext attacks.

VI. REFERENCES

[1]. Kan Yang, Qi Han, Hui Li, Kan Zheng, Zhou Su and Xuemin Shen, "An Efficient and Fine-grained Big Data Access Control Scheme with Privacy-preserving Policy", IEEE Internet of Things, 2018.

[2]. Qi Li, Youliang Tian, Yinghui Zhang, Limin Shen and Jinjing Guo, "Efficient Privacy-Preserving Access Control of Mobile Multimedia Data in Cloud Computing", Sep 2019.

[3]. Yujiao Song, HaoWang, XiaochaoWei and LeiWu, "Efficient Attribute-Based Encryption with Privacy-Preserving Key Generation and Its Application in Industrial Cloud", March 2019.

[4]. Yang Ming and Tingting Zhang, "Efficient Privacy-Preserving Access Control Scheme in Electronic Health Records System", Oct 2018.

[5]. Shangping Wang, Keke Guo and Yaling Zhang, "Traceable ciphertext-policy attribute-based encryption scheme with attribute level user revocation for cloud storage", Sep 2018.

[6]. P. Jayasree and V. Saravanan, "High Secure and dynamic Access Control Scheme for Big Data Storage in Cloud Environment", July 2018.

[7]. Harsha Bhat, Yudhish N,Yadunandan R and Shreyas LS, "A Secure and Verifiable Access Control Scheme for Big Data Storage in Clouds", May 2018.

[8]. Kai Fan, Qiong Tian, Junxiong Wang, Hui Li and Yintang Yang, "Privacy Protection Based Access Control Scheme in Cloud-Based Services", Jan 2017.

[9]. K. Yang, Z. Liu, X. Jia, and X. S. Shen, "Time-domain attribute-based access control for cloud-based video content sharing: A cryptographic approach," IEEE Trans. on Multimedia (to appear), February 2016.

[10]. K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, and W. Xiang, "Big data-driven optimization for mobile networks toward 5g," IEEE Network, vol. 30, no. 1, pp. 44–51, 2016.

[11]. Z. Su, Q. Xu, and Q. Qi, "Big data in mobile social networks: a qoeoriented framework," IEEE Network, vol. 30, no. 1, pp. 52–57, 2016.

[12]. H. Li, D. Liu, K. Alharbi, S. Zhang, and X. Lin, "Enabling fine-grained access control with efficient attribute revocation and policy updating in smart grid," KSII Transactions on Internet and Information Systems (TIIS), vol. 9, no. 4, pp. 1404–1423, 2015.

[13]. H. Li, D. Liu, Y. Dai, and T. H. Luan, "Engineering searchable encryption of mobile cloud networks: when qoe meets qop," IEEE Wireless Communications, vol. 22, no. 4, pp. 74–80, 2015.

[14]. H. Li, Y. Yang, T. Luan, X. Liang, L. Zhou, and X. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," IEEE Trans. on Dependable and Secure Computing DOI: 10.1109/TDSC.2015.2406704], 2015.

[15]. K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 12, pp. 3461–3470, Dec 2015.

[16]. R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," IEEE Network, vol. 28, no. 4, pp. 46–50, 2014.

[17]. K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 7, pp. 1735–1744, July 2014.

[18]. L. Lei, Z. Zhong, K. Zheng, J. Chen, and H. Meng, "Challenges on wireless heterogeneous networks for mobile cloud computing," IEEE Wireless Communications, vol. 20, no. 3, pp. 34–44, 2013.

[19]. J. Hur, “Attribute-based secure data sharing with hidden policies in smart grid,” IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 11, pp. 2171–2180, 2013.

[20]. C. Dong, L. Chen, and Z. Wen, “When private set intersection meets big data: an efficient and scalable protocol,” in Proc. of CCS’13. ACM, 2013, pp. 789–800.

[21]. J. Lai, R. H. Deng, and Y. Li, “Expressive cp-abe with partially hidden access structures,” in Proc. of ASIACCS’12. ACM, 2012, pp. 18–19.

[22]. B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” in Proc. of PKC’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 53–70.

[23]. J. Lai, R. H. Deng, and Y. Li, “Fully secure cipertext-policy hiding cpabe,” in Information Security Practice and Experience. Springer, 2011, pp. 24–39.

[24]. P. Mell and T. Grance, “The NIST definition of cloud computing” Recommendations of the National Institute of Standards and Technology- Special Publication 800-145], 2011.

Cite this article as :

Nisha J William, Nisha O S, "A Secure and Fine-Grained Big Data Access Control Scheme for Cloud-Based Services", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 6 Issue 4, pp. 254-262, July-August 2020. Available at
doi : https://doi.org/10.32628/CSEIT206448
Journal URL : http://ijsrcseit.com/CSEIT206448