A File Encryption System Based on Attribute Based Encryption

Jie Tong
School of Information Engineering
Wuhan University of Technology
Wuhan, P. R. China
E-mail: tongj0501@qq.com

Yihong Long*
School of Information Engineering
Wuhan University of Technology
Wuhan, P. R. China
E-mail: longyihong@sina.com

Quan Liu
School of Information Engineering
Wuhan University of Technology
Wuhan, P. R. China
E-mail: quanliu@whut.edu.cn

2021 17th International Conference on Computational Intelligence and Security (CIS) | 978-1-6654-9489-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/CIS54983.2021.00100

***Abstract*—Attribute Based Encryption (ABE) is a kind of public-key cryptosystem that is commonly used to protect file data security and achieve the fine-grained sharing of files. However, the existing attribute-based encryption algorithms are very complex and involve a large amount of additional cryptographic data, which is not convenient to implement. To solve this problem, an attribute-based data encryption scheme is proposed in this paper to achieve secure storage and access control of files. This system is convenient to implement and reliable in operation. In this file encryption system, the attribute private key is divided into a server-side secret share and a user-side secret share. When a user is authorized to access a file, the client and the server decrypt the encrypted file by collaborative computation using the user-side secret shares and server-side secret shares of the user's attribute private keys.**

Keywords-Attribute Based Encryption; attribute private key; file encryption;

I. INTRODUCTION

At present, a large amount of sensitive information exists in the form of files, and the leakage of confidential files has become a major hidden risk for electronic data security. There have been many years of research on electronic file storage, and there have been many successful applications. However, single electronic file encryption cannot satisfy the fine-grained decryption control requirements for encrypted electronic files. The latest scheme combines file encryption and decryption with an access control policy to achieve fine-grained access control of encrypted electronic files.

Attribute Based Encryption (ABE) is a kind of public key cryptosystem that is an extension of Identity Based Encryption (IBE) [1]. Compared to IBE, the ABE replaces identity with attributes, where both the user's private key and the ciphertext are associated with a set of attributes, and the encryption party can specify the group of recipients by setting an access policy consisting of attributes so that the ciphertext can be decrypted only when the user satisfies the access policy. In 2005, Sahai and Waters proposed the concept of ABE and constructed a concrete scheme that uses a threshold access structure where the receiver can decrypt the message encrypted by the publisher only if the number of identical attributes between them reaches a threshold value set by the system [2]. According to different application situations, Attribute Based Encryption can be divided into two categories: Key-Policy Attribute Based Encryption (KP-ABE) and Ciphertext-Policy Attribute Based Encryption (CP-ABE). In KP-ABE, the decrypt keys are associated with access control policies, and the ciphertexts are associated with attribute sets [3-5]. The CP-ABE scheme is opposite to the KP-ABE scheme. Its ciphertexts correspond to the access control policies and the decrypt keys correspond to the attribute sets, which can be successfully decrypted only if the attributes in the attribute set satisfy the access control policy [6-8]. Therefore, the CP-ABE scheme is commonly used for encrypted storage of files and fine-grained access control.

An obvious problem of the existing CP-ABE schemes is that the algorithm is complex, involving a large amount of additional cryptographic data in private key generation, encryption, and decryption, which is difficult to implement for general engineers. Most of the application situations of attribute-based encryption are carried out in a client/server model. For such application scenarios, we can combine attribute-based encryption and server-side access control to solve the problem instead of letting the attribute-based encryption algorithm solve all the problems.

In this work, we provide an attribute-based encryption scheme to solve this problem. In this system, the attribute private keys are divided into server-side secret shares and user-side secret shares, which are kept by the server and the user respectively. In the scheme, threshold attribute set and non-threshold attribute set can be used to encrypt data. The encryption process uses an encryption algorithm based on bilinear mapping. In this paper, SM9 [9-10] is used to encrypt the data. When decrypting data, the user and server respectively use the user-side secret shares and server-side secret shares of the user's private keys to decrypt the encrypted data through collaborative calculation. Based on this scheme, combined with server-side access control, we can achieve the requirements for encrypted file storage and secure sharing.

978-1-6654-9489-2/21/$31.00 ©2021 IEEE
DOI 10.1109/CIS54983.2021.00100

II. PROPOSED SCHEME

A. Overview of Our Scheme

In this paper, an attribute-based encryption scheme is proposed, in which ciphertext is associated with one or more sets of attributes. Attribute sets can be divided into non-threshold sets and threshold attribute sets. In the former case, a file can be decrypted when the user's attributes satisfy any of the attribute sets in the decryption control data, and in the latter case, a file can be decrypted when the user's attributes satisfy the threshold of the threshold attribute set in the decryption control data. In addition, a threshold attribute set can also be regarded as an attribute that encrypts files together with other attributes. The overall framework of the system is described in Fig. 1.

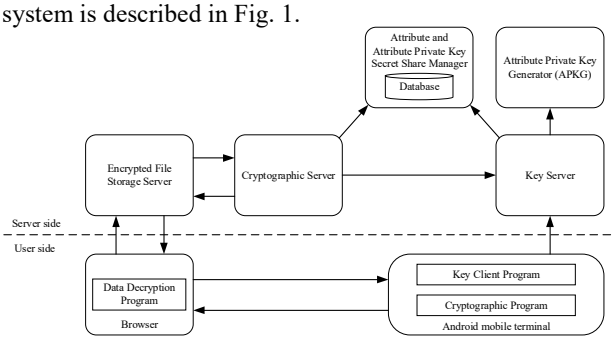

Figure 1. The framework of the whole system

The system is based on C/S architecture and consists of seven main parts as follows.

(1) Attribute Private Key Generator (APKG): generate the attribute private keys corresponding to the user's attributes.

(2) Key Server: the system component of the server that interacts with the user-side key client program when the user requests a secret share of the attribute private key. The secret share of the attribute private key is the secret that is related to the attribute private key and can be restored by the server side and the user side respectively.

(3) Attribute and Attribute Private Key Secret Share Manager: manage and store the user's attributes and the secret shares of the attribute private keys.

(4) Android mobile terminal: the device includes key client program and cryptographic program, achieves cryptographic calculations, and stores the shares of the attribute private key on the user side.

(5) Browser: the browser uploads files, interacts with the server, and calls the Android mobile terminal for decryption operations.

(6) Encrypted File Storage Server: files are encrypted and stored with access control in the server.

(7) Cryptographic Server: the Cryptographic Server interacts with the user to decrypt the file with the Encrypted File Storage Server as an intermediary.

B. Details of Attribute Based Encryption Scheme

1) Attribute Private Key and Attribute Private Key Secret Share Generation and Management

ABE involves a bilinear mapping: e: $G_1 \times G_2 \rightarrow G_T$, where G_1, G_2 are additive groups and G_T is a multiplicative group. The order of the group G_1, G_2, G_T is a prime integer n. The generator of group G_1 is P_1, and the generator of group G_2 is P_2.

APKG secretly holds an integer master key s in [1, n-1], calculates the master public key $P_{pub}=sP_2$, and publishes the master public key P_{pub}.

When a user requests a secret share of the private key corresponding to attribute A (i.e., attribute private key) d_A, the user submits the request to obtain a secret share of the private key of attribute A to the Key Server. The Key Server verifies through the user database of the Attribute and Attribute Private Key Secret Share Manager that the user applying for the attribute private key secret share is the claimed user and has the corresponding attribute A. After the verification passes, the application submitted by the user is submitted to the APKG, and if the verification does not pass, the Key Server rejects the user's application. The key service system generates the secret share of the attribute private key d_A for attribute A as follows:

(1) APKG maps the identifier of attribute A to the horizontal coordinate x_A of an element in group G_1, and then gets two vertical coordinates by the elliptic curve equation corresponding to group G_1, and takes one of the vertical coordinates y_A, then $G_A = (x_A, y_A)$. Not every x_A has a solution y_A. For the case of no solution, perform multiple hash transformations on x_A until there is a solution y_A [1]. G_A is the public key corresponding to attribute A (i.e., the attribute public key).

(2) Calculate the private key corresponding to attribute A (i.e., the attribute private key) $d_A=sG_A$. Then d_A is returned to Key Server and cached in memory.

(3) Key Server chooses a random integer t_Z in [1, n-1] and computes $d_t=(t_Z)^{-1}d_A$, where $(t_Z)^{-1}$ is the multiplicative inverse of t_Z modulo n (i.e., $(t_Z\ (t_Z)^{-1}) \bmod n=1$).

(4) Take t_Z as d_{Au} and d_t as d_{As}, then $d_A=d_{Au}d_{As}$. d_{As} is the secret share of the server-side attribute private key d_A, d_{Au} is the user-side secret share of the attribute private key d_A.

Key Server stores d_{As} in the user database and returns d_{Au} to the user. The user stores d_{Au} in the key database of the Android mobile terminal.

User's attributes include attributes limited by the time validity period and attributes that are not limited by the time validity period. Attribute limited by the time validity period is composed of the attribute identifier and the time validity period. The time validity period is composed of the start time and the end time. For attributes limited by the time validity period, the user applies for the secret share of the private key corresponding to the attribute limited by the new time validity period within the specified time before the expiration of the time validity period of the attribute. The user and Attribute and Attribute Private Key Secret Share Manager store the secret shares for the new attribute private keys on the user side and the server side respectively.

2) Encryption

The operation of data encryption is implemented independently by the data encryption party. The encryption process uses the key derivation function KDF(), message

authentication code function MAC(K_2, Z), and SM4 block cipher algorithm in SM9. Using SM4 to encrypt the plaintext m with the key K_1 is denoted as Enc(K_1, m). K_1_len is the bit length of the key K_1 in the block cipher algorithm, K_2_len is the bit length of the key K_2 in the function MAC(K_2, Z). The data encryption party uses the attribute set S=(A_1,..., A_m) to encrypt data M with SM9 as follows:

(1) Map the attributes A_1,..., A_m to the group G_1 respectively in the same way as the APKG, and the attribute public keys corresponding to attributes A_1,..., A_m are G_{A1},..., G_{Am}.

(2) Choose a random integer r in [1, n-1].

(3) Calculate $w=e(G_{A1}+\ldots+G_{Am}, P_{pub})^r$, then convert the data type of w to a bit string.

(4) Calculate $C_1=rP_{pub}$, then convert the data type of C_1 to a bit string.

(5) Splice the attribute name $A_1||\ldots||Am$ as the identifier IDs of the attribute set S.

(6) Calculate the integer klen=K_1_len+K_2_len, and then calculate K=KDF($C_1||w||$IDs, klen). Let K_1 be the leftmost K_1_len bits of K, and K_2 be the remaining K_2_len bits, if K_1 is an all-0 bit string, return (2).

(7) Calculate C_2=Enc(K_1, M).

(8) Calculate C_3=MAC(K_2, C_2).

(9) Output ciphertext $C=C_1||C_3||C_2$.

In addition, the data encryption party can also use the threshold attribute set S=(A_1,..., A_m| t, m) to encrypt data, where $m\geq 2$, $m\geq t\geq 1$ (i.e., if the user has any t of attributes A_1,..., A_m, then he/she can decrypt the encrypted data). The encryption operation is as follows:

(1) Choose a random integer z in [1, n-1].

(2) Map z to an element G_s in group G_1 in the same way as the APKG (the mapping method can also be different). Gs is the public key corresponding to the threshold attribute set S=(A_1,..., A_m| t, m).

(3) According to the above method of using the public key corresponding to the non-threshold attribute set to encrypt data, take G_s as the public key corresponding to a non-threshold attribute set, and use G_s to encrypt the data (i.e., calculate $w=e(G_s, P_{pub})^r$) and output ciphertext C.

(4) Take n as the modulo of the modulo operation, based on the Shamir threshold secret sharing scheme [11], z is divided into m integer secret shares z_1,..., z_m in [0, n-1] according to the (t, m) threshold secret sharing.

(5) z_1,..., z_m are encrypted as enc_z_1,..., enc_z_m by the public keys of attributes A_1,..., A_m according to the above method of using the public key corresponding to the non-threshold attribute set to encrypt data where an attribute can be regarded as a non-threshold attribute set containing only one attribute.

(6) The final encrypted data is {C, enc_z_1,..., enc_z_m}.

When the threshold attribute set is used together with other attributes to encrypt data (e.g., one or more threshold attribute sets together with other attributes constitute a threshold or non-threshold attribute set), each threshold attribute set is regarded as an attribute, the public key corresponding to the threshold attribute set is used as the public key corresponding to the attribute to encrypt data.

3) Decryption

When the user decrypts the data encrypted with the attribute set S=(A_1,..., A_m) (i.e., when the encrypted data is decrypted with $d_{A1}+\ldots+d_{Am}$ as the private key), where d_{A1},..., d_{Am} are the attribute private keys corresponding to the attributes A_1,..., A_m. Cryptographic Server on the server side checks whether there are the secret shares d_{A1s},..., d_{Ams} of the attribute private keys requested to decrypt the data in the user database. If not, the decryption operation fails. If there is, Cryptographic Server uses d_{A1s},..., d_{Ams}, while the user side uses the secret shares d_{A1u},..., d_{Amu} of the attribute private keys stored on the user side to decrypt the data through interaction without revealing their respective secrets.

Assume the ciphertext is $C=C_1||C_3||C_2$. The decryption process uses the key derivation function KDF(), message authentication code function MAC(K_2, Z), and SM4 block cipher algorithm in SM9. Using SM4 to decrypt the ciphertext c with the key K_1 is denoted as Dec(K_1, c). K_1_len is the bit length of the key K_1 in the block cipher algorithm, K_2_len is the bit length of the key K_2 in the function MAC(K_2, Z). The decryption operation between the Cryptographic Server and the user side is as follows:

(1) Cryptographic Server takes out $C_1=rP_{pub}$ from C.

(2) Cryptographic Server calculates $w_{s1}=e(d_{A1s},C_1)$, $w_{s2}=e(d_{A2s},C_1)$,…, $w_{sm}=e(d_{Ams},C_1)$, then send w_{s1}, w_{s2},…, w_{sm} to the user side.

(3) The user-side Android mobile terminal calculates $w_{u1}=w_{s1}\hat{}d_{A1u}$,…, $w_{um}=w_{sm}\hat{}d_{Amu}$, where ^ denotes a power operation, the element before ^ is the base and the number after ^ is the exponent, then send w_{u1},…, w_{um} to the Cryptographic Server.

(4) Cryptographic Server calculates the multiplication of w_{u1},…, w_{um}, and $w'=w_{u1}\ldots w_{um}=e(d_{A1}+\ldots+d_{Am},C_1)$, then convert the data type of w' to a bit string.

(5) Splice the attribute name $A_1||\ldots||Am$ as the identifier IDs of the attribute set S.

(6) Calculate the integer klen=K_1_len+K_2_len, and then calculate K'=KDF($C_1||w'||$IDs, klen). Let K_1' be the leftmost K_1_len bits of K', and K_2' be the remaining K_2_len bits. If K_1' is an all-0 bit string, report an error and exit.

(7) Calculate M'=Dec(K_1', C_2).

(8) Calculate u=MAC(K_2', C_2), take out C_3 from C, if $u\neq C_3$, report an error and exit.

(9) Output plaintext M'.

When the data decryption party decrypts the data {C, enc_z_1,..., enc_z_m} encrypted with the threshold attribute set S=(A_1,..., A_m| t, m), Cryptographic Server first checks to confirm that the user has at least t parts of the m attributes A_1,..., A_m, if there is, continue to decrypt, otherwise the data decryption will fail. The decryption process with threshold attribute set as follows:

(1) Take out enc_z_1,..., enc_z_m from the encrypted data.

(2) T parts of z_1,..., z_m are decrypted by the secret shares of the attribute private keys of the t attributes in the attributes A_1,..., A_m through interaction according to the above method of using the private key corresponding to the non-threshold attribute set to decrypt data where an attribute can be regarded as a non-threshold attribute set containing only one attribute.

(3) Reconstruct z by Lagrange interpolation with t secret shares of z.

(4) The Key Server and APKG map z to an element G_s in group G_1, and calculate $d_s=sG_s$.

(5) d_s is regarded as a non-threshold attribute set or the private key corresponding to an attribute to decrypt the encrypted data (i.e., $w'=e(d_s, C_1)$ needs to be calculated).

If the data is encrypted by one or more threshold attribute sets together with other attributes, each threshold attribute set corresponds to a private key, the private key corresponding to the threshold attribute set is regarded as the attribute private key for decrypting the encrypted data.

The attribute sets including the ordinary attribute sets and threshold attribute sets are chosen according to the security requirements and policies.

4) File Encryption and Access Control

In this system, files are encrypted and stored in a designated folder. The system administrator can set an access control policy for the files in a specific directory, and the policy is stored in the corresponding directory in the form of an XML file. The access control policy for the files consists of a file decryption policy and an access control policy of the Encrypted File Storage Server.

For the file decryption policy, we use the logical relationship of "AND" and "OR" to form different access rules (e.g., (A OR B OR C) AND (D OR E)). For the access control policy of the Encrypted File Storage Server, we use the logical relationship of "AND" and "NOT" to form different access rules (e.g., F AND G AND (NOT H) AND (NOT I)).

After the user uploads the file from the browser, Encrypted File Storage Server encrypts and stores the file as follows:

(1) Generate a random symmetric key k_e and use SM4 symmetric key cryptographic algorithm to encrypt the file.

(2) Read the file decryption policy from the policy configuration file in the designated directory and parse it into p different sets of attributes $S_1,\ldots, S_p$, $p\geq 1$ (e.g., (A OR B OR C) AND (D OR E) can be parsed into attribute sets: S_1=(A,D), S_2=(A,E), S_3= (B,D), S_4= (B,E), S_5= (C,D), S_6= (C,E)).

(3) The symmetric key is encrypted by p attribute sets $S_1,\ldots, S_p$ into p ciphertext data $k_{enc\_1},\ldots, k_{enc\_p}$ according to the aforementioned attribute based encryption scheme.

(4) Combine the p attribute sets $S_1,\ldots, S_p$ and the corresponding p copies of ciphertext data $k_{enc\_1},\ldots, k_{enc\_p}$ into file decryption control data.

(5) Attach file decryption control data to the header of the encrypted file. Then store the encrypted file in the designated directory.

When the user accesses the Encrypted File Storage Server through a browser and requests an encrypted file. The Encrypted File Storage Server reads its access control policy, checks whether the user's attributes satisfy the policy, if not, the decryption fails, and if it does, it submits the encrypted file to Cryptographic Server to request decryption of the file. The Cryptographic Server takes out the file decryption control data from the encrypted file, and then checks whether the user's attributes satisfy any of the p sets of attribute sets $S_1,\ldots, S_p$ in the file decryption control data from the attribute information of the user in the database. If not, the decryption fails, and if it does, the Cryptographic Server uses the secret share of the server side secret shares of the private keys corresponding to the user's attributes in the attribute set which the user satisfies to interact with the user side through the Encrypted File Storage Server to decrypt the random symmetric key k_e. Finally, the encrypted file is decrypted by k_e and returned to the Encrypted File Storage Server, and the Encrypted File Storage Server returns the plaintext of the file to the user. The relationship between server side and user side is shown in Fig. 2.

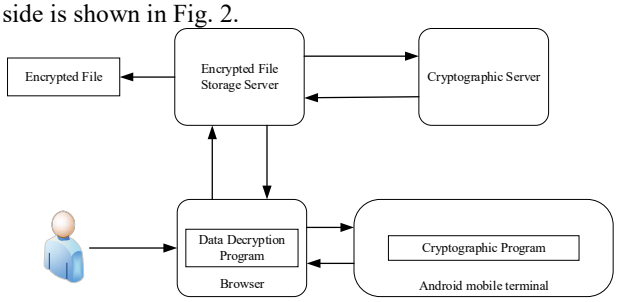

Figure 2. The relationship among server side and user side during decryption

The Android mobile terminal performs the pairing operation as a cryptographic component and returns the calculation result to the Cryptographic Server through the browser.

III. ANALYSIS OF THE SYSTEM

In this scheme, the attribute private keys are divided into the server-side attribute private key secret shares and the user-side attribute private key secret shares. In the decryption, the server side and the user side need to participate in collaborative computing, and the attribute private keys will not be reconstructed in this process, thus reducing the risk of attribute private key leakage and ensuring the security of attribute private key usage. In addition, since users will not be authorized to obtain the complete attribute private key, users cannot carry out collusive attacks through their attributes.

In a single attribute-based encryption scheme, the attributes in the policy can only be set as the logical relationship of "AND" and "OR". We combine the attribute-based encryption with the attribute-based access control implemented by the Encrypted File Storage Server to achieve the rule setting of "AND", "OR" and "NOT" between the attributes and enhance the access control of the system.

Compared with the traditional CP-ABE, for a user attribute set S, if the CP-ABE scheme is adopted, for each attribute $A\in S$, it computes $D_A=(g^r)\cdot(H(A)^{r_A})$, $D_A'=g^{r_A}$, where g is a generator of the multiplicative group in CP-ABE, r is a random integer corresponding to S, r_A is a random integer corresponding to attribute A, and H() is a hash function that maps attributes to the multiplicative group. If the scheme in this paper is adopted, it only needs to compute $d_A=sG_A$. It can be concluded that for each attribute, the amount of cryptographic data for CP-ABE is twice the

number of the scheme in this paper. With the increase of attributes, the gap between the amount of cryptographic data of the user's private key in the two schemes will be greater. If one or more attributes of a user change, it is not necessary to update the entire user's private keys, but only the corresponding attribute private keys. In encryption and decryption, we can use public key cryptographic algorithms such as SM9 or IBE instead of relying on complex access control structures. It can be concluded that the scheme proposed in this paper involves less additional cryptographic data and is more convenient to implement.

IV. THE FUNCTIONAL TEST

For a designated file directory, the system administrator sets the access control policy corresponding to the files in that directory: the decryption control policy of the file is "(sexName: male OR sexName: female) AND 2 of [major: cs, province: Hubei, role: undergraduate]", where "2 of [major: cs, province: Hubei, role: undergraduate]" means a threshold attribute set with a threshold value of 2, and the control policy of Encrypted File Storage Server is "school: whut AND (NOT role: teacher)". A file test.txt is used to upload to this directory, it is encrypted as shown in Fig. 3. When a user U_1 requests for test.txt and his attributes satisfy the access control policy, he can get the decrypted file and open it with NOTEPAD.EXE as shown in Fig. 4. When a user U_2 requests test.txt and his attributes do not satisfy the access control policy, he will not be authorized to obtain the file as shown in Fig. 5.

Figure 3. The encrypted file

Figure 4. File obtained by the user U_1

Figure 5. File opened by the user U_2

V. CONCLUSION

In this paper, a file encryption system based on ABE is proposed. In this scheme, the threshold attribute set and the non-threshold attribute set can be used to set access control policy and encrypt data. Compared with the existing CP-ABE, the ABE in this scheme is more convenient in both private key generation, encryption, and decryption, and easy to implement attribute revocation and private key update. The system combines Attribute Based Encryption with access control to achieve flexible access control for files. The successful implementation and testing of the system show that it can effectively satisfy the needs of file security protection and fine-grained access control.

ACKNOWLEDGMENT

The authors would like to thank the supports from the National Key R&D Program of China, Grant No. 2020YFB1710804.

REFERENCES

[1] D. Boneh, M. K. Franklin. "Identity-Based Encryption from the Weil Pairing,"Advances in Cryptology – Proc. Crypto 2001, LNCS 2139, Springer-Verlag, 2001, pp. 213-229.

[2] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," Advances in Cryptology – EUROCRYPT 2005, LNCS 3494, Springer-Verlag, 2005, pp. 457-473.

[3] Nuttapong Attrapadung, Benot Libert, and Elie de Panafieu, "Expressive Key-Policy Attribute-Based Encryption with Constant-Size Ciphertexts," Advances in Public-Key Cryptography – PKC 2011, LNCS 6571, Springer-Verlag, 2011, pp. 99-108.

[4] Hohenberger S and Waters B, "Attribute-Based Encryption with Fast Decryption," Advances in Public-Key Cryptography – PKC 2013, LNCS 7778, Springer-Verlag, 2013, pp. 162-179.

[5] Y. Rahulamathavan, S. Veluru, J. Han, F. Li, M. Rajarajan, and R. Lu, "User Collusion Avoidance Scheme for Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption," IEEE Transactions on Computers, Vol.65(9), Sept. 2016, pp. 2939-2946, doi: 10.1109/TC.2015.2510646.

[6] Phuong Viet Xuan Tran, Thuc Nguyen Dinh, and A. Miyaji, "Efficient Ciphertext-Policy ABE with constant ciphertext length," 2012 7th International Conference on Computing and Convergence Technology (ICCCT), IEEE Press, Dec. 2012, pp. 543-549.

[7] U. C. Yadav, "Ciphertext-policy attribute-based encryption with hiding access structure," 2015 IEEE International Advance Computing Conference (IACC), IEEE Press, Jun. 2015, pp. 6-10.

[8] P. P. Kumar, P. S. Kumar and P. J. A. Alphonse, "An Efficient Ciphertext Policy-Attribute Based Encryption for Big Data Access Control in Cloud Computing," 2017 Ninth International Conference on Advanced Computing (ICoAC), IEEE Press, Dec. 2017, pp. 114-120.

[9] C. Tian, L. Wang, and M. Li, "Design and Implementation of SM9 Identity Based Cryptograph Algorithm," 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA), IEEE Press, Sept. 2020, pp. 96-100.

[10] Ping Zhen, Yinzi Tu, Bingbing Xia, Jie Gan, and Xiaoke Tang, "Research on the Miller Loop Optimization of SM9 Bilinear Pairings," Proc. 17th IEEE International Conference on Communication Technology (ICCT 2017), IEEE Press, Oct. 2017, pp. 179-185.

[11] Q. Li and Y. Zhou, "Research and application based on A. Shamir's (t, n) threshold secret sharing scheme," 2012 7th International Conference on Computer Science & Education (ICCSE), IEEE Press, Jul. 2012, pp. 671-674.