Analysis of Security Access Control Systems in Fog Computing Environment

Junlin Zhang

School of Information Technology, Guangdong Technology College, Zhaoqing, Guangdong, 526100, China
E-mail: Junlin_Zhang2023@outlook.com

Received 08 February 2023; Accepted 08 April 2023;
Publication 12 August 2023

Abstract

Fog computing is a computing environment that can respond to user operational needs in real time. Aiming at the shortcomings of user privacy protection performance and structural performance, a method of completely hiding access structures is proposed under the framework of cloud and mist computing. The cuckoo filter is applied to the fog computing environment, and users are detected through fog nodes. If an attribute is detected to exist in the fully hidden access structure, the mapping function between the attribute and the access structure line number is returned. The research results show that with the increase of the number of attributes, the advantage of attribute confirmation time for fog servers is gradually obvious; The overall delay of fog computing is shorter, the Time To Live (TTL) is longer, the average delay is only 3 ms, and the delay is lower; The completely hidden access structure constructed by the cuckoo algorithm occupies only 1% of the total system

Journal of Cyber Security and Mobility, Vol. 12_5, 653–674.
doi: 10.13052/jcsm2245-1439.1252

steps, which can more effectively achieve user privacy protection without increasing overhead. The proposed scheme greatly reduces the amount of computation while fully protecting user privacy, and meets the needs of users for fast and secure access.

Keywords: Fog computing, cuckoo filter, secure access, control technology, structure hiding.

1 Introduction

After more than a decade of development, cloud computing technology has steadily advanced, but its shortcomings are also increasingly evident. The shortcomings of data transmission between cloud computing and users, such as high latency and low scalability, have become difficult to meet user needs. In order to solve the problems in cloud computing, scholars have proposed the integration of fog computing technology and cloud computing to form a new architecture that provides more high-speed and high-bandwidth network services [1–3]. Fog computing is a fusion of multiple technological trends, similar to cloud computing, and suitable for distributed computing and storage in Internet of Things scenarios. Fog computing is a network at the edge of users, which can respond to users' operational needs in real time, and thus solve the delay problem in cloud computing. In a fog computing environment, access control is a common technology for protecting data security [4–6]. Access control refers to the use of various technologies to restrict the access of foreign objects to subject data resources, among which attribute cryptography is widely used. When using attribute encryption in practice, in addition to the most basic user data security needs to be guaranteed, as parties continue to enhance collaboration during the operation process, it is also necessary to add access structures in real time [7–9]. In order to achieve more flexible and secure data access control in a fog computing environment, the research focuses on access control under cloud and fog architectures, and constructs a secure access structure hiding scheme. By confirming the integrity of data, new access structures are added and completely hidden, in order to achieve more timely data sharing and higher levels of privacy protection. Most of the existing research on access structures can only be partially hidden and cannot fully protect data and user privacy. The proposed scheme can not only fully protect user privacy, but also reduce unnecessary computation and improve various performance.

2 Related Work

At present, fog computing has been extensively used in various fields [10, 11]. Many studies have applied fog computing to improve the efficiency and security of computer operations. There is no shortage of Yousefpour A team's research on the computing performance of fog computing itself. They compare fog computing with edge computing and conduct performance analysis of derivative computing methods [12]. However, more teams have applied fog computing to improve the efficiency and security of computer computing. For example, the Tange K team and the Muttag A A A team have conducted corresponding research. Their common feature is that they have applied fog computing to the security protection of specific industries. The Tange K team has chosen the industrial industry, while the Muttag A A team has chosen the healthcare industry. Due to the fact that the generation of fog computing itself relies on network security protection technology, such applications tend to be more fundamental and expanded [13, 14]. While other researchers focus on the computational efficiency of fog computing itself, they will use fog computing to replace cloud computing with lower computational efficiency, reduce computational latency, and improve network operational stability. The Abdulqadir H R team, Lakzaei M team, and Rahman F H team have all conducted this type of research. The Abdulqadir H R team prefers to work with the Internet of Things multi-user architecture for efficient computing, while the Lakzaei M team and Rahman F H team prefer to work with virtual machine energy consumption and computing efficiency [15–17]. This research takes into account two research directions, based on the expanded application research of network security, and also analyzes the computing efficiency of fog computing during the model design process. It fully utilizes the computing characteristics and application strengths of fog computing itself to achieve efficient control of security access.

In the field of the Internet, where fog computing is often used, there are still many applied studies on Internet security. The most important Internet security issues are user access security and privacy security. The Ranaweera P team, Bhatt S team, and Qi J team have all conducted research in this area, because with the development of technology, user security issues often have updated characteristics. Ranaweera P team analyzed the access security problem in the multi access edge computing system and enhanced the security of the multi access edge computing system. Bhatt S team started with the user access verification of the Internet of Things to conduct access control research, while QiuJ team discussed the user privacy problem [18–20].

In addition, the Gope P team has also conducted a discussion on user rights security, but this discussion is more targeted, mainly focusing on the security of industrial sensors and industrial system user protocols [21]. It can be seen that the current research on Internet security issues mainly focuses on user security. This study also starts from the perspective of user security, but uses fog computing, a new computing method, to explore the applicability of security. At the same time, it utilizes the computing characteristics of fog computing itself, and uses security access control technology in the fog computing environment to establish a security access structure concealment scheme, which improves system performance while protecting user privacy.

3 Facing the Security Access Control Technology in the Fog Computing Environment

A completely hidden access structure scheme for fog computing environments is proposed, using cuckoo filters to hide and detect attributes. The access structure represented by a linear secret sharing matrix is completely hidden and stored in the fog node, saving storage space for the cloud server, and querying whether user attributes exist in the hidden access structure in the fog node. The method proposed in the study can fully protect the privacy of users, reduce the amount of computing, and make computing more rapid and convenient.

3.1 Relevant Basic Technologies for Secure Access in the Fog Computing Environment

Fog computing is a result of the fusion of various technological tendencies. It is suitable for use in distributed computing and storage IoT scenarios. A large number of wired or wireless devices are distributed near each node of the user. These devices have network services interoperability and scalability. With the development of computer technology, data encryption has officially entered a new stage of modern cryptography. Encryption refers to transforming information into data that cannot be understood by attackers, and decryption refers to the restoration of encrypted information to understandable information by a specific user through some special means [22]. The basic flow of cryptography is shown in Figure 1.

As shown in Figure 1, the data sender obtains the encryption key at the key generation place, encrypts the information through a certain encryption algorithm to form ciphertext, and the data receiver acquires the decryption

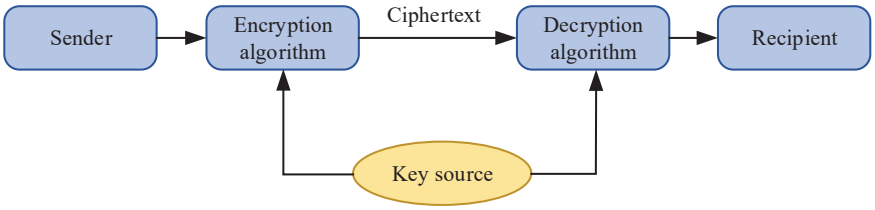


Figure 1 Basic process of cryptography.

key at the key generation place, and uses a certain decryption algorithm to decipher the ciphertext to get the original message. The hash function is a commonly used tool in cryptography, which converts the input data into a specific length output through an algorithm. The length of the input value of the hash function is random, but the output value is of a fixed length. Any input value can get a unique output value. The input value can be called the preimage of the output value, and the input value cannot be counted by the output value. MurMur hash algorithm is the most commonly used in recent years. It is more suitable for retrieval operations, with fast speed and a good hash effect. Its core steps are shown in formula (1).

$$\begin{cases} x^* = m \\ x = rotate\_left(x, r) \end{cases} \tag{1}$$

In formula (1), m is the optimal constant value obtained through a large quantity of experiments, and r is the number of bits to be set. When the input values have a strong regularity, the output value distribution characteristics of the Murmur hash algorithm perform well. Divide the secret D into n fragments, if and only when the collection amount of the secret share value reaches the k share, the complete secret value can be D reconstructed. If the collection amount of the secret share value is less than k the complete secret value, the complete secret value will not be reconstructed. This is the case of (k, n) secret sharing algorithm. In solving practical problems, the value of k is generally set as the number of secret shares, and the secret sharing algorithm is realized by XOR operation. Select $n - 1$a random value $r_1, r_2, \ldots, r_{n-1}$ for calculation, r_n and reconstruct according to the calculated r_n pair as shown in formula (2).D

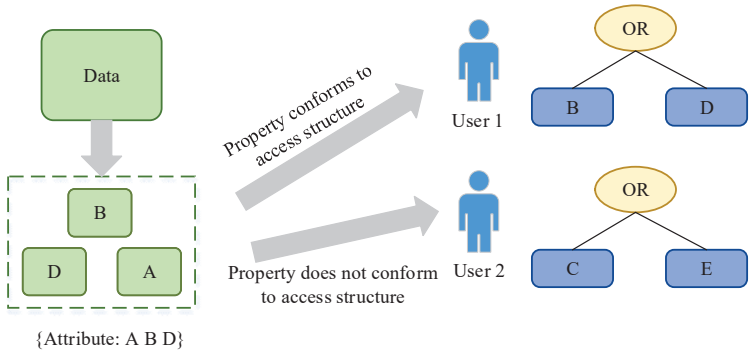
$$\begin{cases} r_n = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus D \\ D = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus r_n \end{cases} \tag{2}$$

The concept of secret sharing is realized by using the Lagrangian interpolation method. In this approach, the secret is shared with the keepers, and y the shared secret can be reconstructed $x \leq y$ only when there is one set of keepers, which needs to be satisfied x. When the user specifies the access structure, the access structure can be transformed into a computable matrix through Linear Secret Sharing Schemes (LSSS).

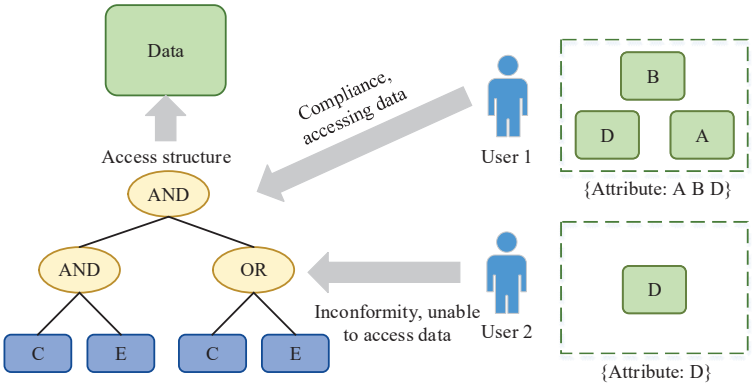
Bilinear pairing is often used in general encryption schemes to build systems. Let G be the p order cyclic addition group, G_r be the p order cyclic multiplication group, g as G the generator of the addition group, p be a prime number whose length is a safety parameter. There are bilinear maps e: $G \times G \to G_r$ that are bilinear, nondegenerate, and computable.

The Attribute-Based Encryption (ABE) method regards attributes as identity information. Both the ciphertext and the key are related to the attributes. When the attributes are within the range specified by the encryptor, the information can be decrypted and obtained. In this method, the server for information storage can be any server that is not trusted, and there is no need to check the server, so the speed of information storage is faster. The data owner only needs to pay attention to the encryption protection of the data. The use of attributes for access control can ensure the confidentiality of the data. When decrypting, the random value in the key participates in the calculation to resist malicious attacks. ABE is mainly divided into KP-ABE and CP-ABE, as shown in Figure 2.

A cuckoo filter can test for the presence or absence of an element in a collection, and can dynamically adjust the elements in the collection. The filter can also achieve higher performance and faster execution speed than general filters in a space with a higher capacity, and can resolve the problem of false positives of elements in a general filter bank with advantage. Each cuckoo hash table is composed of a set of storage group arrays, each element of which can obtain two addresses in the table according to two hash functions, and one is used as an alternative address. Add elements to the cuckoo hash table, check the obtained address, if one of the two addresses is empty, add an element to this position. When an element collides, the alternate address is enabled. When a collision occurs in the alternate address, the new element will select any position and delete the element at the position. The deleted element will repeat the above steps until a new empty position is found. If the space occupancy rate of the table has reached the preset value, it is determined that the table is full, and a new cuckoo hash table demands to be constructed. Compared with the cuckoo hash table, the cuckoo filter has basically the same location structure as it, but is more space-saving.



(a) KP-ABE Application Diagram



(b) CP-ABE Application Diagram

Figure 2 Two types of ABE.

The cuckoo filter obtains the bit string related to the element to be stored through a specific hash function, that is, the fingerprint of the element. The size of the fingerprint will affect the false positive rate of the element. The longer the number of fingerprint digits, the lower the false positive rate. The size of the fingerprint will grow logarithmically with the number of storage addresses as well [23]. Cuckoo filters improve space efficiency by storing element fingerprints. When inserting the fingerprint information of an element in the filter, the addresses of two candidate slots must first be obtained through a hash function, as shown in formula (3).

$$\begin{cases} h_1(x) = hash(x) \\ h_2(x) = h_1(x) \oplus hash(x's\ fingerprint) \end{cases} \tag{3}$$

It can be seen from formula (3) that the vacancy address of the other candidate position can be calculated through one of the two candidate positions. $h_1(x)$ and $h_2(x)$ denote two alternative addresses, $hash(x)$ is the element address, while $hash(x's\ fingerprint)$ means the fingerprint of the element.

3.2 Facing the Hidden Scheme of Security Access Control Structure in the Fog Computing Environment

This research proposes an access control hiding plan for fog computing, and the system model of the scheme is shown in Figure 3.

In Figure 4, a trusted institution is a completely trusted institution set, which is responsible for generating system parameters and master keys. The trusted institution in this scheme will not be in league with other entities and will not be attacked by default. Data users need to obtain the information uploaded by the data uploader, and they need to pass their own attribute set to the trusted organization during the registration phase, and request their own keys from it as well. Data uploaders share data by uploading information, and they have the right to specify the access structure. The fog node stores completely hidden access structure. Before the user downloads the ciphertext, it detects the attributes used to check whether the attributes conform to the

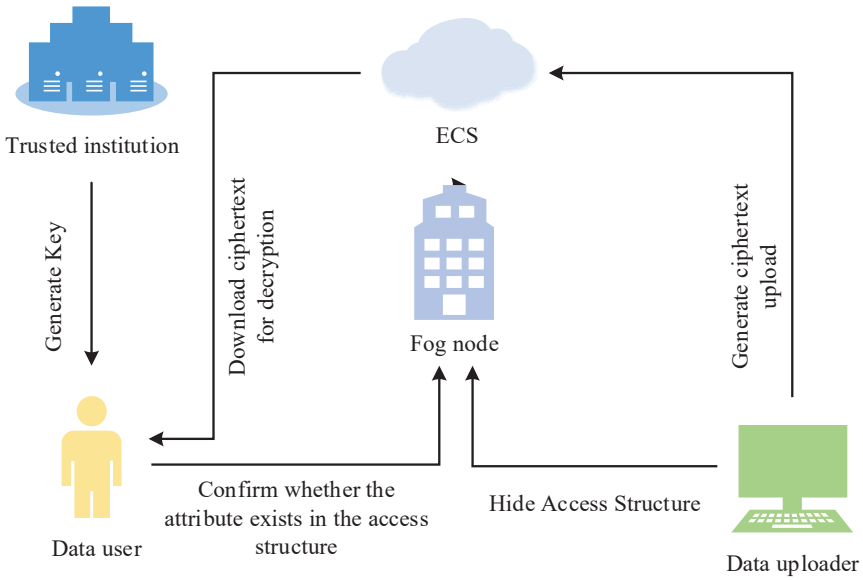


Figure 3 Scheme system model.

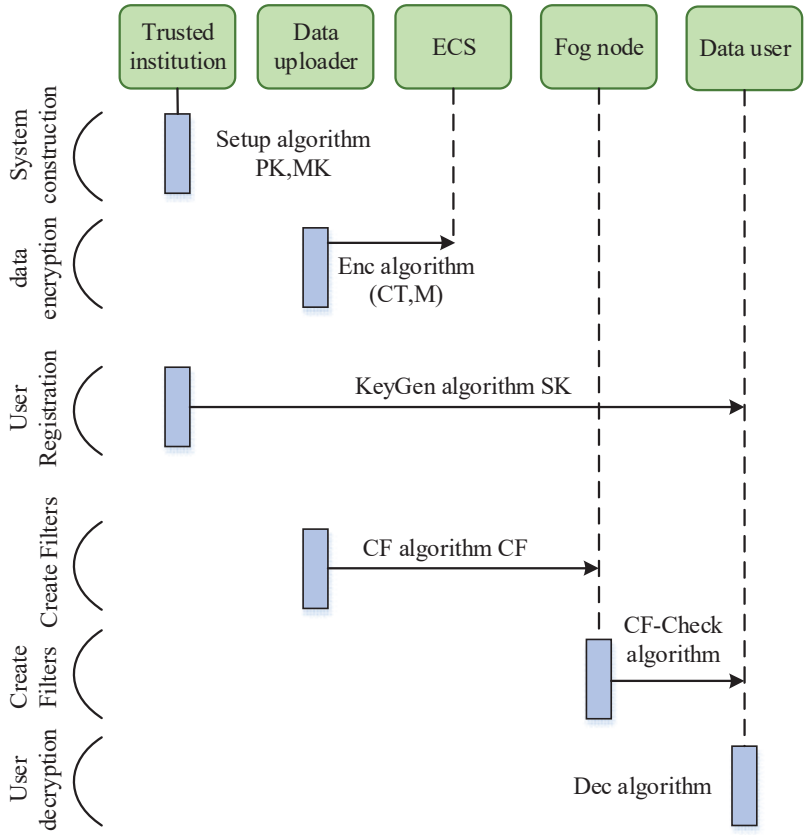


Figure 4 Data flow diagram.

access structure emplace by the shared data. The cloud server allows data uploaders and data users to store ciphertext in it, and the generated wood grain is uploaded for storage, and after the attribute is confirmed, the relevant ciphertext is downloaded [24].

System initialization algorithm $Setup$ is operated by a trusted authority to produce relevant parameters and master keys. Input the security parameters, the algorithm will choose the 1^{λ} two cyclic groups of G and G_r on the G prime number order, p and the mapping relationship G_r between $G \times G \to G_r$ and, the trusted authority chooses the producer to $g \in G$ generate random elements related with the attributes $h_1, \ldots, h_N \in G$, N representing the quantity of attributes in all. The trusted organization chooses randomly μ, $\mu \in Z_p^*$, specifies L_{att} the maximum bit extent of the attribute, indicates

the maximum bit length of L_{att} the line number in the matrix showed by LSSS, uses H_f the anti-collision hash function to H_e indicate the fingerprint information of the generated element, and indicates that the element is filtered with the cuckoo element. Finally, the trusted institution gains the parameters and master key by this algorithm, as shown in formula (4).

$$\begin{cases} PK = \langle g, L_{att}, L_{row}, H_f, H_e, h_1, \ldots h_N, e(g,g)^{\mu}, g^{\mu} \rangle \\ MK = g^{\mu} \end{cases} \tag{4}$$

The user registration $KeyGen$ algorithm is run by a trusted institution, and the decryption key of the data user is generated by it. When data users decrypt the ciphertext downloaded from the ECS, they first need to apply for a private key, send their own attribute set to the trusted institution, and the trusted institution that receives the attribute set will run the algorithm to produce the private key and distribute it. While the the algorithm runs, it selects $t \in Z_p^*$ randomly, and get the customized private key as shown in Formula (5).

$$SK = \langle g^{\mu}, g^{\mu t}, g^t, \{E_x = h_x^t\}_{x \in U}, U \rangle \tag{5}$$

Encryption information algorithm Enc is run by the data uploader, and the plaintext data is encrypted through the specified access structure, and the corresponding ciphertext is generated and uploaded to the server. Enter the system parameters PK, the specified access structure (M, ρ) and plaintext data m, and Enc the ciphertext corresponding to the plaintext data will be returned CT. The data user runs Enc algorithm to calculate the corresponding ciphertext of the plaintext data as shown in formula (6).

$$CT = \langle me(g,g)^{\mu s}, g^s, \{C_i = g^{\mu \lambda_i} h_{\rho(i)}^{-s}\}_{i=1,\ldots,l} \rangle \tag{6}$$

Cuckoo filtering algorithm CF is run by the data uploader, which inputs (M, ρ) the access structure, completely hides it through the cuckoo filter, generates the hidden one and transmits it to the fog node for storage. When the data owner needs to put a certain attribute in the access structure into the cuckoo filter, first $H_f(att_x)$ obtain the corresponding fingerprint information of the attribute through the hash function f, and then $H_e(att_x)$ obtain the position where the element is mapped in the cuckoo filter. If there is already information at this location, then another alternative location address is calculated to store the corresponding fingerprint information of this attribute, as shown in formula (7).

$$\begin{cases} Ads_1(x) = H_e(att_x) \\ Ads_2(x) = Ads_1(x) \oplus H_e(f) \end{cases} \tag{7}$$

In addition, it is also necessary to process the elements. After accessing each element in the structure, connect the line number of the attribute element in matrix M. Set that the connected elements need to meet the requirements of $\lambda - bit$ length, and fill the left side with 0 to reach the $\lambda - bit$ length. After the attribute processing, the element group is obtained as shown in Formula (8).

$$U_x = \{i \| att_x\}_{i \in [i,l]} \tag{8}$$

In Equation (8), i represents the row number, l represents the number of rows of the matrix, and the mapping relationship between each row and the attribute elements needs to be satisfied $att = \rho(i)$. The algorithm constructs U_x a hidden access structure according to the constructed one CF. When inserting a new element into the table, first calculate the fingerprint information $f = H_f(x)$ of the element, and then calculate the position of the element in the table $i_1 = H_e(x)$, and then obtain another alternative location address in the table through an algorithm $i_2 = i_1 \oplus H_e(f)$, and set the conditions as shown in formula (9).

$$f \oplus x = CX \tag{9}$$

When either of the two alternative locations is empty, place $\langle f, CX \rangle$ at the address.

Confirmation property algorithms $CF - Check$ are run by fog nodes. When the system parameters PK are entered, the hidden access structure CF and the attribute set assigned by the individual data user, first U calculate the first position in the cuckoo filter for the attribute of the data user, and then calculate the value of the attribute $H_e(att_x)$. Fingerprint information f, another candidate address information is calculated according to the fingerprint information and the first location information, as shown in formula (10).

$$\begin{cases} Ads_1(x) = H_e(att_x) \\ Ads_2(x) = Ads_1(x) \oplus H_e(f) \end{cases} \tag{10}$$

Check whether the fingerprint information $H_f(att_x)$ of the attribute exists in the two alternative addresses. If not, the attribute of the user does not belong to the access structure. If it exists, it restores the formula (11) in the hidden access structure.

$$x = H_f(att_x) \oplus (CX) \tag{11}$$

In formula (11), CX is the information placed on the element mapping address when the cuckoo filter is created. Since x connects the line number

of this attribute in matrix M after passing the element, and zero complement operation is carried out when the length of $\lambda - bit$ is not satisfied, so the zero bit on the left side of the string is first deleted, and then the length of L_{att} is used to intercept the attribute string att, and the length of L_{row} is used to intercept the line number row. After getting att, compare it with the attribute assigned by the input individual in the system to att_x and judge whether the two are completely consistent. If they are completely consistent, the user is a user who conforms to the access structure, and the mapping function between the calculated reconstruction attribute and the row number is ρ shown in formula (12).

$$\rho = \{row, att\}_{att \in U} \tag{12}$$

Decryption algorithm Dec is run by the data user. For the mapping function reconstructed after the attribute detection is successful ρ', the data user will download the corresponding ciphertext and decrypt the ciphertext with the private key. If one enters the ciphertext corresponding to the plaintext data, the specified access structure (M, ρ) and a, private key SK, then the Dec algorithm will return m. When the data user satisfies the access structure, the calculation of the original data m is shown in formula (13).

$$m = \frac{me(g,g)^{\mu s}}{e(g,g)^{\mu s}} \tag{13}$$

The data flow diagram of this operation is shown in Figure 5.

4 Application Performance Analysis of the Security Access Control Structure Hiding Scheme

When analyzing application performance, we focus first on analyzing the comparison of computing overhead caused by the difference in filter types. The analysis mainly starts with the calculation method of the filter, and compares the Bloom filter with the cuckoo filter. The specific comparison results are as presented in Table 1.

As can be seen from Table 1, the main difference between the two filters is that Bloom filter needs to undergo k hashing operations in the operation process, while Cuckoo filter only needs to undergo two hashing operations. At the same time, when the cuckoo filter hides the access structure, when it reaches the preset space occupancy boundary, it will use a new cuckoo hash table and fill in the corresponding data. This part of the overhead is

Table 1 Comparison of filter operation cost

Type of Computational Overhead	Filter Type	
	Cuckoo	Bloom
Encryption overhead	$j*(2+2\Gamma)+2\Omega+\phi$	$j*(2+2\Gamma)+k\Omega+\phi$
Accessing structures completely hides the overhead	$n*(2\Omega+\phi)$	$k\Omega+\phi$
Confirm attribute overhead	$n*(2\Omega+o_{(1)})+\phi$	$k\Omega+\phi$
Decryption overhead	$i(2\Delta+1)+j\Delta+n*(2\Omega)+\phi$	$i(2\Delta+1)+j\Delta+k\Omega+\phi$

also included in the calculation of the method. From the calculation point of view, the calculation of the cuckoo filter is more efficient. During the application test, the 1 core 2G Memory device is used to simulate the fog node device, and the MurMurHash v3.3 function is used to implement the cuckoo filtering algorithm. The research uses the cloud computing security model in literature [25] as a comparison model [25]. The structural time change under different attribute numbers hidden access is shown in Figure 5.

From Figure 6 it is visible that the growth trend in the early stage is stable, and the obvious growth trend occurs when the number of attributes is between 30 and 40. When the quantity reached 40, the time consumed by the algorithm when building the hidden access structure rises to 0.22 ms, and the later the growth rate stabilizes. When it reaches 60, the time consumed by the algorithm when constructing the hidden access structure is up to 0.26 ms. The time consumed by the algorithm showed a linear increase trend with the rise of the number of attributes. During this increase, the attribute confirmation time also shows a gradual growing trend, but the rate of rise is relatively slow. The overall trend is relatively stable, rising from 0.1 ms when the quantity of attributes is 10 to 0.6 ms when the number of attributes is 60. For the same number of attributes, the attribute confirmation time of the cloud server environment is higher than that of the fog server. It can be seen that with the rise of the volume of attributes, the advantage of the attribute confirmation time of the fog server is gradually obvious. The obtained delay test results in the cloud environment are shown in Figure 6.

Figure 7 shows the distribution and reception status of cloud environment address data packets for the same user at four time points, using four indicators, respectively, the amount of distributed data, the amount of received data, the time consumed by round-trip data, and the system cache time TTL (Time

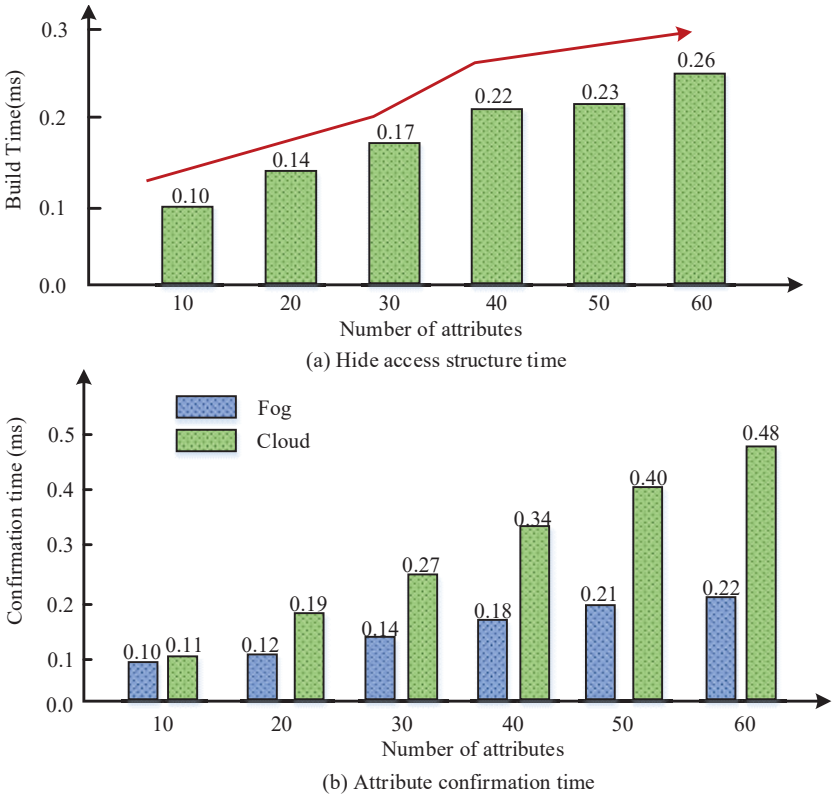


Figure 5 Time change during the modification of attribute number.

To Live). For the purpose of this analysis, the two indicators of sending data and receiving data are used to detect data omissions during data transmission, and the time spent on round-trip data is used to test data transmission delay. Moreover, TTL time is used to detect whether data transmission time exceeds the system cache time. When the time nodes are 1, 2, 3 and 4, the round-trip data consumption time is 47 ms, 46 ms, 46 ms and 47 ms respectively. The TTL time is 51 ms, which is shorter than the round-trip data consumption time. On this basis, the study tested the fog environment under the same conditions, and the specific results are shown in Figure 7.

Figure 8 shows the distribution and reception of fog environment address data packets at four time points for the same user. The four indicators of data distribution, data reception, data round-trip consumption time and system cache time TTL time are also used for analysis. Under time node 1, data

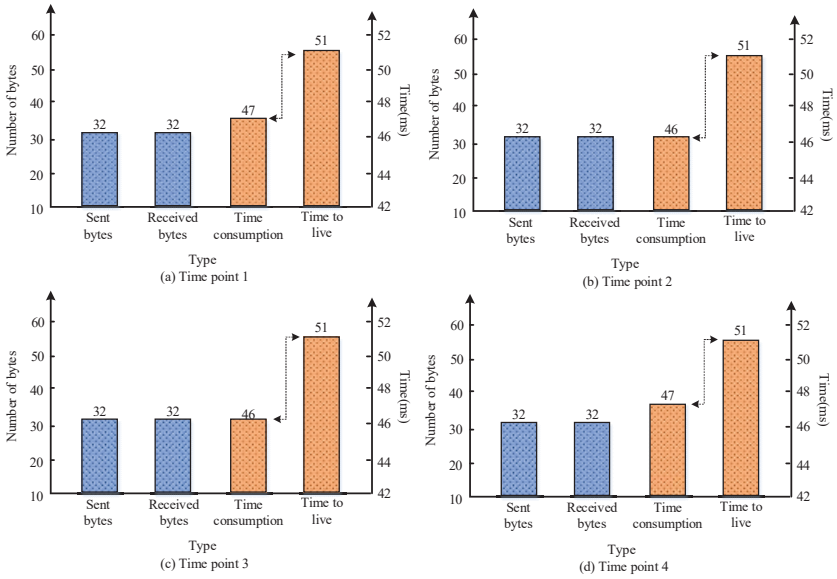


Figure 6 Cloud environment delay test results.

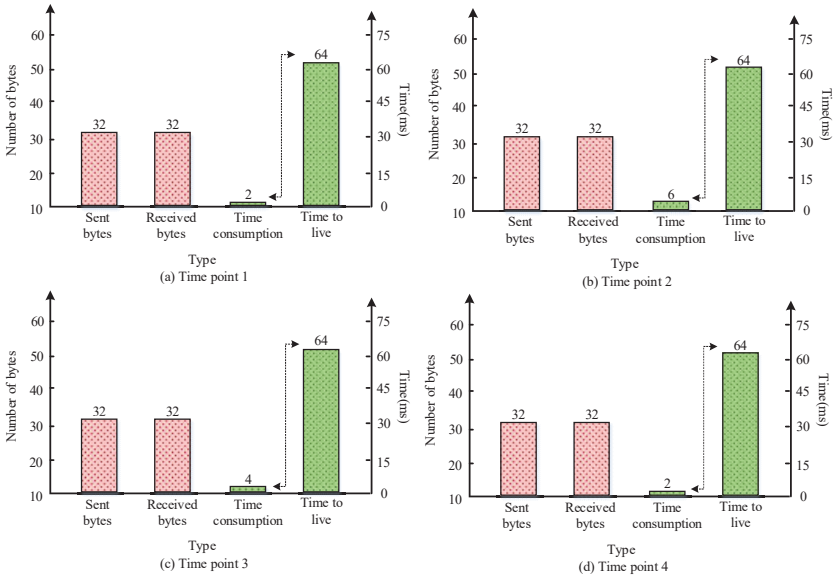


Figure 7 Fog environment delay test results.

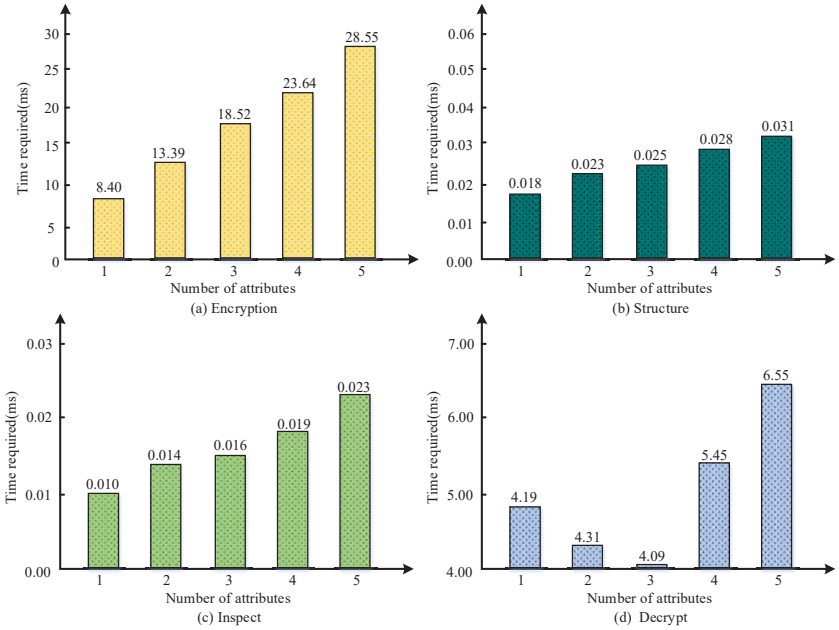


Figure 8 Comparative analysis of system time.

round-trip consumption time is 2 ms, which is much lower than 47 ms and 64 ms of TTL time in cloud environment. When the time node is 2, the round-trip data consumption time is 6 ms, and the TTL time is 64 ms. Under time nodes 3 and 4, the round-trip data consumption time is 4 ms and 2 ms respectively, and the TTL time is still 64 ms. The round-trip data consumption time is shorter than the TTL time. Compared with the cloud environment, the overall delay in the fog environment is shorter and the TTL time is longer. The average delay in the cloud environment is 46 ms, while the average delay in the fog environment is only 3 ms, and the delay in the fog environment is even lower. The system encryption time, construction time, inspection time and decryption time are compared and analyzed under different numbers of attributes, as shown in Figure 8.

From Figure 9 it can be seen that with the rise of the quantity, the encryption time of the system shows a gradually increasing trend. With the rise of the quantity, the construction time of the system shows a trend of steady growth but the increase rate is relatively slow. The growth trend is stable, and there is little difference in the construction time under different number of attributes; with the increase of the attribute number, the system inspection time shows

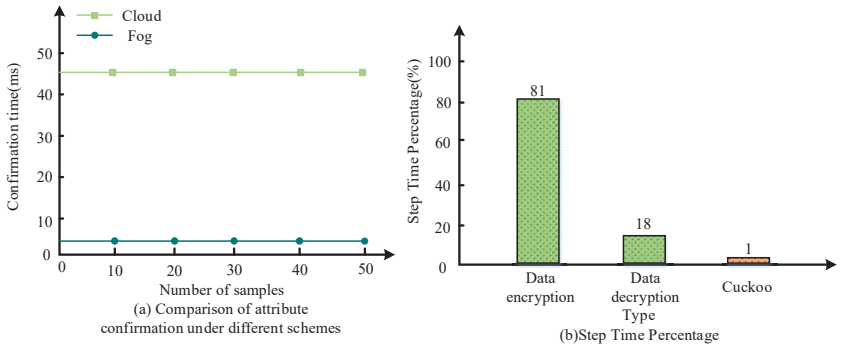


Figure 9 Proportion of scheme attribute confirmation time to total time of different steps.

a steady growth trend, but the increae rate is relatively slow for the different number of attributes. There is also little difference in the inspection time; with the rise of the quantity, the decryption time of the system presents a distribution state of index values that are low in the middle and high on both sides, and the lowest decryption time is 4.09 ms. When the quantity is 3, while the highest decryption time is 6.55 ms when the number of attributes is 5. The scheme of the research design is mainly distributed in the construction time and inspection time, and the time consumption is relatively stable, while the decryption time can be controlled by the number of attributes. Figure 9 shows the ratio of the program attribute confirmation time to the total time of different steps.

In Figure 9 it is shown that the time consumed by the confirmation of the hidden access structure attributes in the fog environment is much shorter than the time consumed by the confirmation of the hidden access structure attributes in the cloud environment. The time is stable at about 46 ms, but because the simple confirmation operation time is very short, the impact of the change in the number of attributes is not significant. From the perspective of time proportion, the completely hidden access structure constructed by the cuckoo algorithm occupies a very small proportion of time in the whole system steps, only 1%. Under the circumstances, it can realize user privacy protection more effectively.

In summary, compared to cloud computing models, under the same number of attributes, the attribute confirmation time for fog server environments is relatively shorter. At the same time, as the number of attributes increases, the advantage of attribute confirmation time for fog servers gradually becomes apparent. This indicates that the fog server has the advantage

of computational efficiency. At the same time, in terms of the distribution and reception of address packets in the fog environment at four time points for the same user, the fog computing model has more advantages in terms of four indicators: the amount of data distributed, the amount of data received, the time consumed for data round-trip, and the system cache time TTL time. This indicates that compared to the cloud computing model, the fog computing model has a better overall application effect, and users can experience a fuller data transmission effect with a shorter delay. Finally, verification of hidden access structure attributes in a fog environment consumes less time than verification of hidden access structure attributes in a cloud environment. This indicates that the fog computing model designed in the study has a structural advantage in the security hidden structure, and is easy to achieve better security effects.

5 Conclusions

As a new computing mode, fog computing can provide users with more timely, convenient, and fast virtualization services. The research proposes a security access control structure hiding scheme based on the access control of fog computing. The cuckoo filter algorithm is applied to the fog computing environment, and user attributes are detected through fog nodes. If the attribute is detected to exist in the completely hidden access structure, the mapping function between the attribute and the access structure line number is returned. Through performance analysis experiments, it can be seen that with the increase in the number of attributes, the advantage of attribute confirmation time of the fog server gradually becomes apparent; The overall delay of fog computing is shorter. At 1 byte, 2 bytes, 3 bytes, and 4 bytes of time, the data round-trip consumption time is 2 ms, 6 ms, 4 ms, and 2 ms, respectively, which is shorter than the comparable cloud computing model. At the same time, in terms of TTL time comparison, the TTL time of the fog computing model designed in the study is 61 ms at 1 byte, 2 bytes, 3 bytes, and 4 bytes, which is higher than the 51 ms of the cloud computing model compared. And the average delay of the fog calculation model is only 3 ms, which is lower; The proposed scheme consumes relatively stable construction time and inspection time, while the decryption time can be controlled by the number of attributes; The completely hidden access structure constructed by the cuckoo algorithm occupies only 1% of the time in the entire system step, and the time consumed to confirm the structure properties is stable at about 3 ms, which is far lower than about 46 ms of the cloud computing model. This

indicates that the scheme can more effectively achieve user privacy protection without increasing costs. However, in the study of completely hiding the access structure, it is possible to further consider the dynamic adjustment function of the cuckoo filter and dynamically manage the policies in the access structure. In the future, the error rate of cuckoo filters should also be improved to reduce the error rate of detection attributes.

Fundings

The research is supported by: The characteristic innovation project of ordinary colleges and universities in Guangdong Province, "Research and application of fog computing security access control technology" (2022KTSCX168).

References

[1] Sabireen H, Neelanarayanan V. A review on fog computing: Architecture, fog with IoT, algorithms and research challenges. Ict Express, 2021, 7(2): 162–176.

[2] Rupa C, Patan R, Al-Turjman F, Mostarda L. Enhancing the access privacy of IDaaS system using SAML protocol in fog computing. IEEE Access, 2020, 8: 168793–168801.

[3] Pallavi K N, Ravi Kumar V. Authentication-based access control and data exchanging mechanism of IoT devices in fog computing environment. Wireless Personal Communications, 2021, 116: 3039–3060.

[4] Lei K, Du M, Huang J, et al. Groupchain: Towards a scalable public blockchain in fog computing of IoT services computing. IEEE Transactions on Services Computing, 2020, 13(2): 252–262.

[5] Singh J, Singh P, Gill S S. Fog computing: A taxonomy, systematic review, current trends and research challenges. Journal of Parallel and Distributed Computing, 2021, 157: 56–85.

[6] Wang H, Liu T, Kim BG, et al. Architectural design alternatives based on cloud/edge/fog computing for connected vehicles. IEEE Communications Surveys & Tutorials, 2020, 22(4): 2349–2377.

[7] Patwary A A N, Fu A, Battula S K, Naha R K, Garg S, Mahanti A. FogAuthChain: A secure location-based authentication scheme in fog computing environments using Blockchain. Computer Communications, 2020, 162: 212–224.

[8] Taylor PJ, Dargahi T, Dehghantanha A, et al. A systematic literature review of blockchain cyber security. Digital Communications and Networks, 2020, 6(2): 147–156.
[9] Cheng R, Scott W, Masserova E, et al. Talek: Private group messaging with hidden access patterns//Annual Computer Security Applications Conference. 2020: 84–99.
[10] Goudarzi M, Wu H, Palaniswami M, et al. An application placement technique for concurrent IoT applications in edge and fog computing environments. IEEE Transactions on Mobile Computing, 2020, 20(4): 1298–1311.
[11] Abd Elaziz M, Abualigah L, Attiya I. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. Future Generation Computer Systems, 2021(124): 142–154.
[12] Yousefpour A, Fung C, Nguyen T, et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture, 2019, 98: 289–330.
[13] Tange K, De Donno M, Fafoutis X, et al. A systematic survey of industrial Internet of Things security: Requirements and fog computing opportunities. IEEE Communications Surveys & Tutorials, 2020, 22(4): 2489–2520.
[14] Mutlag A A, Abd Ghani M K, Arunkumar N, et al. Enabling technologies for fog computing in healthcare IoT systems. Future Generation Computer Systems, 2019, 90: 62–78.
[15] Abdulqadir H R, Zeebaree S R M, Shukur H M, et al. A study of moving from cloud computing to fog computing. Qubahan Academic Journal, 2021, 1(2): 60–70.
[16] Lakzaei M, Sattari-Naeini V, Sabbagh Molahosseini A, Amir Javadpour. A joint computational and resource allocation model for fast parallel data processing in fog computing[J]. The Journal of Supercomputing, 2022, 78(10): 12662–12685.
[17] Rahman F H, Newaz S H S, Au T W, Wida Susanty Suhaili, M.A. Parvez Mahmud, Gyu Myoung Lee. ENergy and TRUst-aware Virtual Machine allocation in VEhicle fog computing for catering applications in 5G[J]. Future Generation Computer Systems, 2022, 126: 196–210.
[18] Ranaweera P, Jurcut A D, Liyanage M. Survey on multi-access edge computing security and privacy. IEEE Communications Surveys & Tutorials, 2021, 23(2): 1078–1124.

[19] Bhatt S, Pham T K, Gupta M, et al. Attribute-based access control for AWS internet of things and secure Industries of the Future. IEEE Access, 2021, 9: 107200–107223.
[20] Qiu J, Tian Z, Du C, et al. A survey on access control in the age of internet of things. IEEE Internet of Things Journal, 2020, 7(6): 4682–4696.
[21] Gope P, Das A K, Kumar N, et al. Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks. IEEE transactions on industrial informatics, 2019, 15(9): 4957–4968.
[22] Qiu M, Kung S Y, Gai K. Intelligent security and optimization in Edge/Fog Computing. Future generation computer systems, 2020, 107: 1140–1142.
[23] AboDoma N, Shaaban E, Mostafa A. Adaptive time-bound access control for internet of things in fog computing architecture. International Journal of Computers and Applications, 2022, 44(8): 779–790.
[24] Martinez I, Hafid AS, Jarray A. Design, resource management, and evaluation of fog computing systems: a survey. IEEE Internet of Things Journal, 2020, 8(4): 2494–2516.
[25] Wang Z, Wang N, Su X, et al. An empirical study on business analytics affordances enhancing the management of cloud computing data security[J]. International Journal of Information Management, 2020, 50: 387–394.

Biography

Junlin Zhang obtained his master's degree in engineering from South China Agricultural University (2004). At present, he is working as an associate professor in the School of Information Technology of Guangdong Technology

College. He is also a member of the Professional Committee of Teaching Quality Management of Guangdong Private Higher Education Institutions. He has published articles in more than 10 well-known peer-reviewed journals and conference minutes. His areas of interest include computer network security, Internet of Things technology, software technology and higher education management.