Journal of Cyber Security
DOI:10.32604/jcs.2021.026735
Article

Tech Science Press

Research on ABAC Access Control Based on Big Data Platform

Kun Yang[1], Xuanxu Jin[2] and Xingyu Zeng[1,*]

[1]Beijing University of Posts and Telecommunications, Beijing, 100876, China
[2]Dresden University of Technology, Dresden, 01069, Germany
[*]Corresponding Author: Xingyu Zeng. Email: xingyuz@bupt.edu.cn
Received: 23 October 2021; Accepted: 21 December 2021

Abstract: In the environment of big data, the traditional access control lacks effective and flexible access mechanism. Based on attribute access control, this paper proposes a HBMC-ABAC big data access control framework. It solves the problems of difficult authority change, complex management, over-authorization and lack of authorization in big data environment. At the same time, binary mapping codes are proposed to solve the problem of low efficiency of policy retrieval in traditional ABAC. Through experimental analysis, the results show that our proposed HBMC-ABAC model can meet the current large and complex environment of big data.

Keywords: Big data; access control; ABAC; Hadoop

1 Introduction

With the rapid development and application of Internet of Things, cloud computing and big data technology, the access control scheme of cloud computing platform under big data application environment must be highly scalable, flexible and efficient. However, the access control currently adopted by the traditional big data platform, such as Hadoop [1–3], is based on the static policy specified by the user/user group, and cannot be authorized in groups according to multiple attribute tags of users, let alone the dynamic change of permissions according to the changes of users' attributes, which makes it only suitable for the rights management of a small number of users. The data under the environment of big data is large and dynamic, so the access control list is large and difficult to maintain, the phenomenon of over-authorization and under-authorization is more and more serious, and rights management is complex and difficult [4].

Based on the attribute-based access control model, this paper proposes an access control model suitable for Hadoop. This model makes full use of the advantages of attribute access control [5], and effectively solves the problems of difficult permission change, complex management, over-authorization and insufficient authorization in big data environment. At the same time, it is proposed to use binary mapping code to solve the problem of slow matching in traditional attribute access control policies.

2 Background

2.1 Traditional Access Control

The autonomy in DAC (Discretionary Access Control) model [6] is mainly reflected in that agents with certain access rights can grant part or all of their access rights to other subjects, so the DAC model can also be called arbitrary access control model. Although the model has certain directness, flexibility and easy to implement in the way of authorization, because the user can transfer the authority arbitrarily directly, for example, user A can transfer the access to the object O to user B. user B, which does not have the access right to object O, can access O, and the arbitrary transfer of authority will pose a threat to the data security of the system. Therefore, the security mechanism provided by the DAC model is relatively

low and cannot provide sufficient security protection for the system.

Compared with the DAC model, the mandatory access control MAC (Mandatory Access Control) model [7] has more stringent access control policies, and was first mainly used in the information management of the US government and military. It has two main characteristics: one is mandatory, which is also the most prominent feature of mandatory access control, which stipulates that the security attributes and levels of the subject/object within the system are pre-defined by the system security administrator, except for the system administrator. No subject/object has the right to modify the security attributes of an entity. Another feature is restriction, that is, the system administrator compares the security level of the subject/object, and then determines whether the subject can operate on the object in the way it wants. However, the disadvantage of MAC is also very obvious, that is, the authorization flexibility is poor, and the access control rules are booked in advance.

In view of the shortcomings of DAC model and MAC model in authorization, D. Ferraiolo and R. Kuhn put forward the idea of role-based access control [8] (Role-Based Access Control, RBAC) for the first time in the computer security seminar held by the National Institute of Standards and Technology in 1992. its main content is to add roles between users and permissions in the traditional access control model, and to control the granting and revocation of user rights through role control. Thus, it can effectively reduce the complexity of authorization management caused by too many users of the system. Although the definition of the model has significant advantages in the open network environment, the complexity of authorization management makes the practical application of the model relatively difficult.

Access control is a means to ensure information security, and it is a technology that most systems (including computer systems and non-computer systems) need to use. Traditional access control, such as discretionary access control (Discretionary Access Control, DAC), mandatory access control (Mandatory Access Control, MAC), role-based access control (Role Based Access Control, RBAC), is based on predefined policies that clearly distinguish between allowed access and denied access. It has the characteristics of static and strict policy rules for authorized access, high access granularity, poor flexibility in authorization management and tedious authorization management. Unable to adapt to the current dynamic and distributed network environment.

2.2 Attribute-Based Access Control Model ABAC

Attribute-based access control model is a kind of access control model to solve the trusted relationship of industry distributed applications. It uses relevant entity attributes as the basis of authorization to study how to carry out access control. The basic authorization idea is shown in Fig. 1. According to the pre-defined security policy of the information system, the subject of the access request is authorized according to the attribute feature set, the object feature set and the corresponding environmental attribute set. Three kinds of entity attributes are mainly involved in the basic ABAC model, which are subject attributes, object attributes and environment attributes.

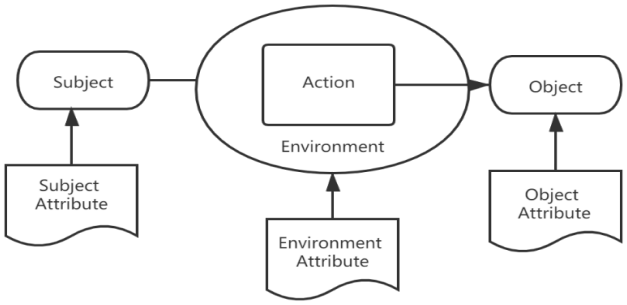

Figure 1: ABAC model authorization thought diagram

Subject: The initiating entity of the access behavior. The attribute corresponding to the subject is called the Subject Attribute (SA). For example, internal characteristics such as the creation time of the

principal, the basic information, or the security level of the principal.

Object: The target object of the access behavior can be data or a service. The attribute corresponding to the object is called the Object Attribute (OA). For example, internal characteristics such as the type, basic information, or security level of a resource.

Environment: The contextual state of the environment when the access behavior occurs. The corresponding attribute of the environment is called the Environment Attribute (EA).

Access: Some kind of behavior to the object initiated by the subject, such as a user applying to play a video or downloading an academic paper. The interview process involves subject, object, environment and behavior, but it is not necessary to know the specific subject, object and environmental information. During the access process, the subject is abstracted as a collection of SA, and different subjects may be abstracted into the same set in some domains. For example, students of the class of 2021 at BUPT University can be abstracted as: {SA (year of enrollment) = 2021, SA (identity) = student}. Objects are abstracted as a set of OA in the process of access, and different objects may be abstracted into the same set in some domains. For example, the video file on computer disk D can be abstracted as: {OA (type) = video, OA (location) = D:\\ video). The environment itself is a collection of EA, which can be the information of the system itself, such as the usage of the system CPU or the security level of the system, or the information that exists objectively, such as the current time or location information.

2.3 XACML Overview

2.3.1 Policy Language Model of XACML

XACML [9] (extensible Access Control Markup Language) is an extensible, platform-independent and highly secure access control policy description language defined by structured Information Standards Promotion Organization (Organization for the Advancement of Structured Information Standards, OASIS). It mainly adds the element component of access control on the basis of ABAC language to realize the accurate description of subject and object, context information, authorization policy and the definition of access control authority of sensitive information such as network service and digital copyright. It provides an important technical platform for the construction of XML model and the compilation of access control policy. First of all, in terms of policy writing specification, XACML defines a standardized policy language model, as shown in Fig. 2.

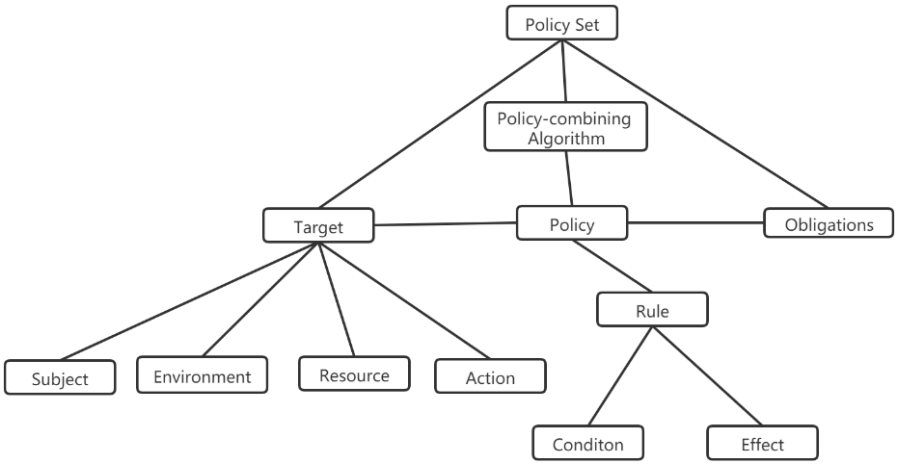

Figure 2: ABAC Model Authorization thought diagram

Rule: It is not only the most basic unit of XACML language model, but also the main component of XACML strategy. Each rule is composed of target (Target), condition (Condition) and effect (Effect), and the final authorization decision result is determined according to the evaluation of conditions. Among

them, the goal indicates the object of the rule, which is usually composed of four kinds of objects: subject, resource, environment and action. Conditions limit the scope of application of the rules, which are usually described by a set of Boolean expressions. The effect is the result or conclusion of the expected evaluation of the rule, which is generally expressed as "Permit" or "Deny".

Policy: It is the smallest unit of interactive access for information systems in XACML. It defines a series of attribute constraints that the subject must meet in order to obtain access to the object. It is mainly composed of four parts: goal, a set of rule sets, obligations and rule combination algorithm identifiers. Among them, the obligation mainly refers to some duties that the visitor needs to perform while implementing the access operation, such as keeping a good record of the authorized visit process. The rule combination algorithm defines the process of obtaining authorization decision based on the evaluation results of a set of rules.

Policy Set: It consists of four parts: goal, a set of policy set, policy combination algorithm and obligation. The main function of the policy combination algorithm is to obtain the authorization decision-making process according to the evaluation results of a set of policies.

In the XACML specification, the combined algorithms that can be applied to both rules and policies are:

(1) Reject the priority algorithm. The idea of this algorithm is that if the evaluation result of a < Rule > or < Policy > element is "Deny", then the authorization decision result of the whole policy combination or rule combination is Deny.

(2) Allow priority algorithm. The basic idea is that a "Permit" evaluation result will make the decision result of the whole rule combination or policy combination as Permit.

(3) The algorithm is applied for the first time. The result of the combined evaluation is equivalent to the evaluation of the first rule or policy element that applies to the current access request.

2.3.2 Policy Language Model of XACML

XACML defines a standardized architecture diagram, as shown in Fig. 3.

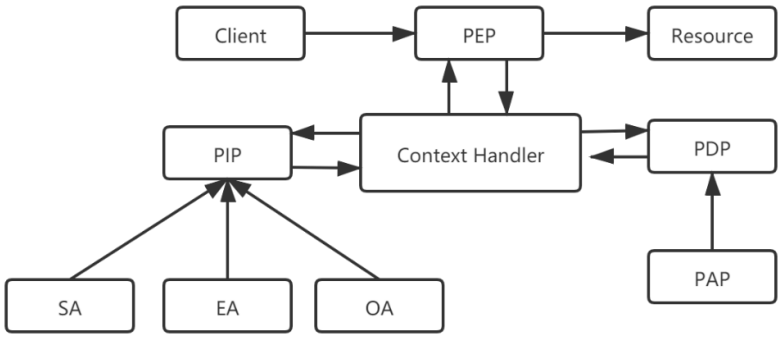

Figure 3: XACML architecture diagram

The policy enforcement component (PEP): It is responsible for receiving the access control request from the principal and sends the access control request to the Context Handler component to receive.

Context Handler: it is responsible for the conversion of input and output formats of data in different application environments. Its main function is to convert the original access request into a request in XACML format and send it to the PDP component.

Policy Decision Point (PDP): It receives the attribute-based access control request from Context Handler; obtains all kinds of attributes and attribute values related to the access request from the AA component, then evaluates the access request according to the access control policy provided by PAP, and returns the evaluation result to the Context Handler component.

The Policy Management Point (PAP): It is responsible for writing, managing and maintaining access control policies and policy sets for PDP authorization decisions.

The Policy Information Point (PIP): It retrieves the attributes and attribute values related to the subject (OA), the object (SA) and the environment (EA), and sends the retrieved result set to the Context Handler component.

3 Related Works

At present, there are many researches on access control, but few of them are related to big data access control. In order to find out the model of access control suitable for big data scenario, although many existing access control technologies are not proposed for big data scenario, they can be used to deal with the problems of access control in big data scenario.

Kunhlman et al. [10] put forward the concept of role mining. Role mining is mainly used to solve the problem of how to generate roles and establish the mapping of user-role and role-permission. Literature [11] uses the algorithm based on machine learning to mine the role of access log, and transforms the problem of role mining into a problem of text classification. Reference [12] is an early work to introduce risk into the field of access control. It defines the concepts of risk quantification and access quota, and gives some guiding principles and suggestions that risk-based information systems should meet. In reference [13], an access control model (Benefit and Risk Access Control, BARAC) is proposed to balance the risks and benefits of information sharing, which measures whether the risk caused by access behavior is acceptable to the system. Therefore, when some unexpected access behavior occurs, it can be accessed according to the magnitude of the risk.

In terms of big data's access control, the early Hadoop did not have any system security mechanism. The first generation Hadoop0.20.0 version began to add a Kerberos-based identity authentication mechanism [14]. After passing the identity authentication, users can apply for data services or submit jobs from the master node of the Hadoop cluster. After the permission is granted, it is no longer supervised, and the illegal operation of legitimate users cannot be prevented.

Although the above research methods solve the problems of over-authorization, lack of authorization and complex rights management to some extent, the above methods have access risks and are likely to cause illegal users to operate on the system. Therefore, based on the attribute-based access control model, this paper proposes a big data access control model suitable for Hadoop, which effectively solves the problems of difficult privilege change, complex management and more and more serious over-authorization and insufficient authorization in big data environment.

4 Access Control of Big Data Based on ABAC

4.1 The Limitations of Big Data's Traditional Access Control

Big data has the characteristics of large volume (Volume), high speed (Velocity), many kinds (Variety) and so on (often referred to as 3V). Big data's 3V characteristics bring great challenges to access control, in which there are a large number of entities and entities are dynamically added or generated, and the relationship between permissions is complex, so it is difficult to manage effectively through the traditional identity-based access control model.

Access control is the key technology to ensure that big data can be shared safely and effectively. The traditional access control model authorizes one by one through static policies, so it will lead to the exponential growth of static policies in the face of a large number of users. This not only brings huge time and space costs [15], but also makes the subsequent permission changes very complex.

The traditional big data access control authorizes a single user or user group. when the rights of multiple users need to be managed uniformly, and these users are not in the same user group, they can only modify the policies of each user one by one, and cannot effectively abstract a class of users.

In big data environment, in the face of dynamically changing users and data resources, the traditional

access control model cannot dynamically change permissions with the change of user attributes, and use a single user ID to distinguish users, so it is difficult to meet the complex user model in big data environment. Based on ABAC, the concept of user group is abandoned and the concept of user attribute set is introduced. The access decision is made by using the attribute, and the access control is carried out according to the subject-object attribute and environment attribute. The corresponding permissions can be obtained according to the attributes of the requesting user, which greatly enriches the authorization mode, makes the access control more flexible, and can be effectively applied in the environment of big data.

4.2 Limitations of Traditional ABAC

In traditional ABAC, the request of subject to object is allowed or denied according to subject attribute, object attribute, environment attribute, request action and ABAC authorization rules, in which attributes and policies are the core elements of ABAC. ABAC makes permission decisions based on the inherent attributes of entities, which largely avoids the difficulty of defining entity permission labels.

When a user requests access to a resource, the attribute information related to the user is analyzed and compared with the policies in the policy set. Usually, the policy analysis in the traditional ABAC model refers to traversing each policy in the policy set to determine whether the user-related attribute information conforms to the policy or not, and if so, grant the corresponding permissions to it. This kind of retrieval is what we often call violence detection, and the disadvantage of violence detection is inefficiency. The main reason is that when analyzing each strategy, it is necessary to analyze the attributes in the strategy in a fine-grained way. In the worst case, the last attribute of the policy does not match the existing attribute of the user. Therefore, it may be necessary to compare more than a dozen or even dozens of attributes of the policy in order to find that the policy does not meet the requirements. Tens of thousands or even hundreds of thousands of policies may be stored in the large policy set in the cloud environment, which will greatly reduce the efficiency of the system.

Suppose that in the control domain, the attribute set contains a bar of attributes, and the policy set contains N policies, with an average of K attributes per policy. Below, we will analyze the time complexity of various algorithms in this domain.

The violence detection algorithm compares the newly added strategy with each policy of the existing policy set in turn. A total of N comparisons are made throughout the comparison process to determine whether it conflicts with the existing policies of the policy set. The condition sets of the two policies need to be compared each time, and the comparison time complexity of the two unordered attribute sets is $O(M^2)$. In the process of comparison, no additional space is needed except for the use of finite variable space, so the time complexity is $O(NM^2)$.

In the scenario of big data, due to too many policy sets, too much time is spent on matching policy sets, which is not conducive to the user experience. In order to solve the problem that the matching speed of ABAC to rules is too slow in big data scene, we propose to use binary mapping code to accelerate the matching speed of rules.

4.3 Optimization Based on Binary Mapping Code ABAC

4.3.1 Definition and Storage of Attribute Set

In order to improve the speed of strategy matching in big data scene. In this paper, we propose to use binary mapping codes to solve this problem. In the traditional access control model, although each attribute in the attribute set has its own serial number, but the serial number does not have much meaning, it is only the ID number set for easy storage. In this paper, the attribute sequence number has a special meaning, and the binary mapping code is also based on the attribute sequence number. We build a table of subject attributes (SA), object attributes (OA) and environment attributes (EA) and number each attribute sequentially. Table 1 shows some of the property sets in the ABAC model.

Table 1: ABAC model property set

ID	Type	Name
0	SA	Att_0
1	SA	Att_1
2	SA	Att_2
…		
n-1	EA	Att_{n-1}

As shown in Table 1, we map each attribute to its own ID number and attribute type, and the attribute ID follows the principle of self-increment starting at 0. Because most of big data is a distributed environment, we store the table in big data storage component HBase database, which can not only use distributed storage, but also ensure the consistency of multiple machines accessing ABAC attribute sets.

4.3.2 Construction of Binary Mapping Code Policy Set

Binary mapping codes are used to represent the attributes contained in each policy. According to the ID number of the attribute, the corresponding binary mapping code is generated for the policy, which is used to mark the set of attributes involved in a policy.

As shown in the Fig. 4, Binary mapping code is a piece of data of length n. Each bit has two values of 0 or 1. Bit i is 1, indicating that it contains a qualification for the attribute numbered i, 0 indicates that it does not contain this attribute. In the process of practical use, the length of Binary mapping code can be adjusted appropriately.

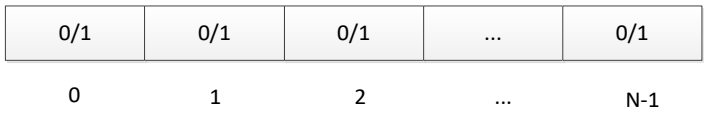

Figure 4: Binary mapping code diagram

Assume that the ID numbers of the 8 attributes in the policy set are the same as their subscripts, and there are only these 8 attributes in the model, that is, the value of n corresponding to Table 1 is 8. Then, the policy set can be represented as Table 2 after the introduction of binary mapping codes, and the policy part is omitted in this table, where Num represents the ID number of the policy set, S represents the sum of binary mapping codes, B represents the binary mapping codes, and P represents the policy.

Table 2: Binary mapping code policy table

Number	S	B	P
0	5	111001010	P0
1	6	111010011	P1
2	3	000111000	P2

After introducing binary mapping code policy set, this paper improves PIP module and PDP module in XACML framework. The BMC module supporting binary mapping code is added in PIP and PDP, respectively. Some adjustments to the XACML system architecture are shown in Fig. 5.

After PIP collects the information of subject, object and environment attribute. The corresponding binary mapping code and binary mapping code and are generated by the BMC module and sent to the PDP along with the attributes. BMC in PDP module based on the analysis. First, compare the sum of the binary mapping codes. If the sum of the binary mapping codes is greater than or equal to the sum of the binary mapping codes in the policy set, then carry out logic or operation with the binary mapping codes of

the policy set in turn. If the calculation result is the same as the original binary mapping code, it may satisfy the relevant policy, and then the attribute information is compared with the policy.

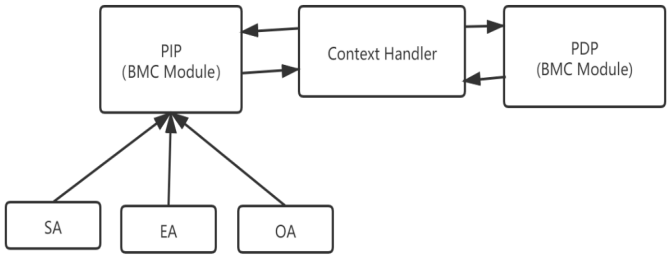

Figure 5: Part of the frame improvement diagram

The sum of binary mapping code and binary mapping code is an optimization to improve the retrieval speed. The second column stores the sum of the binary mapping code S, which represents the number of attributes contained in the policy, and the third column stores the binary mapping code B. Only if the sum of the binary mapping code of the policy to be matched is greater than or equal to the sum of the binary mapping code of the policies in the policy set, the binary mapping code of the policy to be matched will "OR" operate with the binary mapping code of the policies in the policy set. In addition, the fine-grained attribute information comparison analysis should be carried out only if the result of "OR" operation is the same as that of the original binary mapping code; otherwise, the policy will be skipped without matching.

4.3.3 Strategy Set Matching Algorithm Based on Binary Mapping Code

Assume that user U is assigned subject attributes att0, att1, object attributes att2, att3, and environment attributes att5, att6, att7. This attribute set constitutes policy Px, and the policy set P of the system is shown in Table 2. Then, when a user sends an access request to the system, the policy retrieval process can be divided into the following steps:

Step 1: Generate policy Px according to the attributes of the access policy, and obtain binary mapping code Bx, sum of binary mapping code Sx;

Step 2: Judge whether the traversal of the policy set has been completed. If the traversal has been completed, the detection process will end; otherwise, continue;

Step 3: Read the first policy Pi, binary mapping code Bi, sum of binary mapping code Si;

Step 4: Compare the sum of the binary mapping code Sx with the sum of the binary mapping code of the policy Si. Continue if Sx is greater than or equal to Si, otherwise proceed to Step 2;

Step 5: The logic operation "or" of Bx and Si, and judge whether its operation result $Bx \mid Bi$ is equal to the Bx, if they are equal then continue, otherwise go back to Step 2;

Step 6: Judge whether each attribute value of Sx conforms to Si. If so, continue; otherwise, return to Step 2;

Step 7: Return the matching policy.

The algorithm flow chart is shown in Fig. 6.

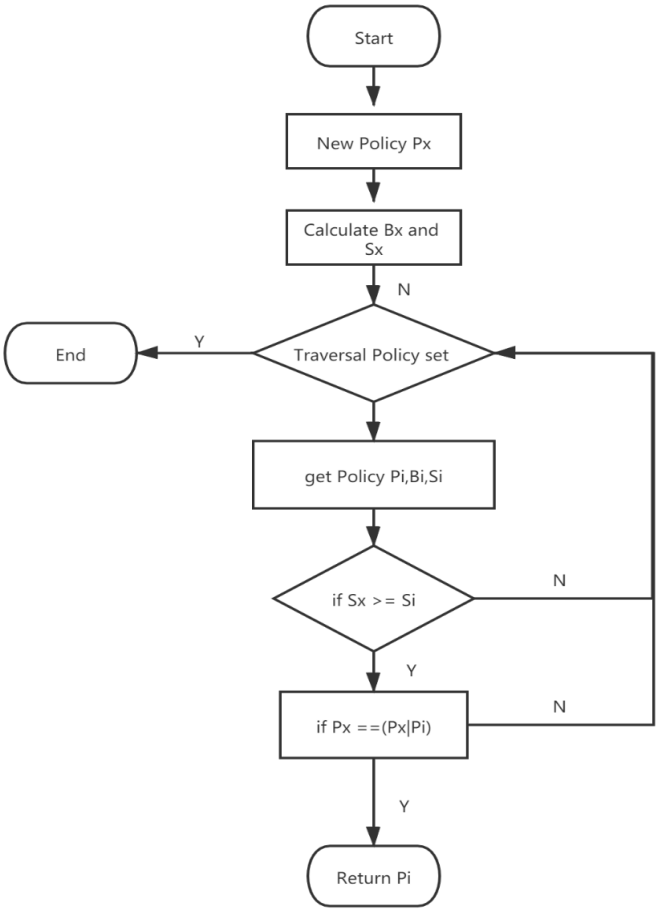

Figure 6: Flowchart of the proposed algorithm

4.4 Based on Improved ABAC Big Data Access Control Framework

Based on the improved ABAC attribute big data access control architecture, HBMC-ABAC, as shown in Fig. 7.

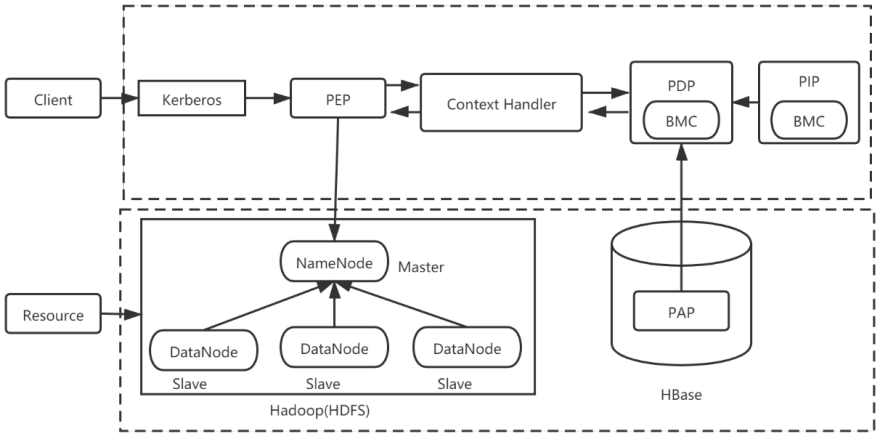

Figure 7: HBMC-ABAC architecture diagram

(1) Function and implementation of PAP module

PAP is a policy set in the HBMC-ABAC model, whose main task is to match the applicable attribute policies for PDP. The format of the policy is the data structure corresponding to Table 2, and the ABAC attribute set corresponding to Table 1 is also stored in PAP module. These two tables are stored in HBase.

(2) Function and implementation of Context Handler module

Send the original access request to the PDP component or receive the result of the PDP decision to the PEP.

(3) Function and implementation of PDP module

PDP is a policy decision module in HBCM-ABAC model. Its main task is to get the decision result of access request by using matching algorithm of matched policy set in PAP and return it to PEP. The policy matching algorithm is the binary mapping code policy matching algorithm proposed in the last section.

(4) Function and implementation of PEP module

PEP is the policy execution module in the HBMC-ABAC model, whose main task is to accept the access request, send it to PDP, accept the decision result of PDP and execute it.

(5) Function and implementation of PIP module

PIP is responsible for providing PDP with the attribute information needed for decision-making, and PIP maintains the attribute information tables of subject, object, operation and environment.

(6) Resource module

Currently, our resource module only supports Hadoop's HDFS module. If you need other big data components, you can add them according to the demand

In the HBMC-ABAC model, the Resource Accessor Client Access request determination process includes the following steps:

Step 1: Client's identity authentication through the Kerberos.

Step 2: When the client passes the authentication, send an access request to the PEP.

Step 3: PEP Send the request to the Context Handler, including attribute information such as resource, action, and environment in the access request, and the Context Handler component resolves the original access control request and send it to the PDP component.

Step 4: PDP parses access request and sends a search request for attribute information to the PIP through the Context Handler component.

Step 5: PIP looks for attribute information such as the subject, resource, environment, etc. associated with access requests, and returns the status information of the search result and the resource to the Context Handler component.

Step 6: PDP sends a policy match request to the PAP, search the relevant policies.

Step 7: PAP uses a binary mapping code algorithm to find matching policies in the policy set and sends all matching policies to the PDP.

Step 8: PDP returns the result to the PEP.

Step 9: PEP returns the result to the client. client accesses data resources in HDFS based on the determination results.

5 Experiment

Experiment 1: Compare ABAC Policy Set Matching Method

In order to test the efficiency of the search algorithm based on binary mapping code, this paper uses Java language to write test code on a PC configured with Intel (R) Core (TM) i5-8500 CPU, Ubuntu 18.04.1 LTS operating system and 8 G memory. The efficiency of the search strategy in the traditional ABAC model and the search efficiency based on the binary mapping code algorithm are tested, respectively.

This paper selects 20 attributes in the attribute set, each policy contains 10 to 14 attributes, and the number of policies is set to 1000, 2500, 5000, 5000, 7500, 10000, respectively. The search efficiency of traditional brute force detection and based on binary mapping code is tested by increasing the number of policies. The brute force algorithm and the binary mapping code are numbered A and B, respectively, and the experimental results are shown in Table 3.

Table 3: Compare the time spent on matching algorithms

K(Number of attributes per policy)	10~14				
N(Number of policy)	1000	2500	5000	7500	10000
A(Time spent /ms)	155	760	2802	4366	7544
B(Time spent /ms）	48	245	678	1011	1558

In the case of an increase in the number of policies, the binary mapping code algorithm proposed in this paper does not increase much time. Compared with A algorithm, B algorithm has good performance, as shown in the following Fig. 8.

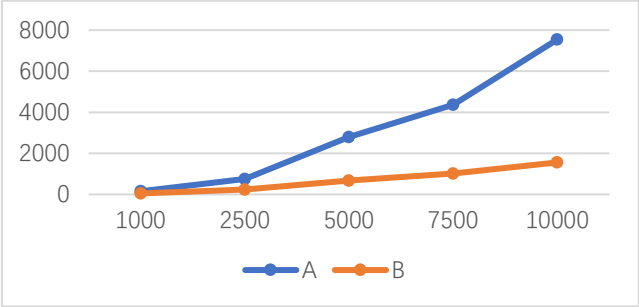

Figure 8: The relationship between policies and time diagram

Experiment 2: Access Control Effectiveness Testing

Set up a Hadoop cluster in the laboratory, which is physically configured as one master node and four slave nodes. The master node is deployed on the server, Ubuntu 18.04.1 LTS OS. Configured for Intel (R) Xeon (R) CPU E5-2650 CPU, 256 G memory, 1000 Mbps Ethernet, 13 TB hard disk. The slave node is deployed on the server, Ubuntu 18.04.1 LTS OS. Configured for Intel (R) Xeon (R) CPU E5-2650 CPU,128 G memory, 1000 Mbps Ethernet, 8 TB hard disk.

The file F defined on the HDFS system is used as the request target of the client access, and the access policy is made for the file. If you access the file F of HDFS, you need to have four attributes, att0, att1, att2, att3. Their values can be specified as val0, val1, val2, respectively. The file can be accessed only if the user contains four attributes: att0, att1, att2 and att3, and the attribute value is one of {val0, val1, val2}.

The total number of users in each experiment was set to 1000, 2500, 5000, 7500 and 10000, respectively. The random function is used to generate the attributes of each end user, so that the ratio of legal users (who have four attributes at the same time and the attribute values meet the requirements) and illegal users (who do not have the above four attributes at the same time or the attribute values are not in the range) is about 1:1 in the total number of users. Count the number of users passed in each experiment (the total number of legal users determined as legal and illegal users determined as illegal), the number of legal users passed (the number of legal users determined as legal) and the number of illegal users passed (the number of illegal users determined as illegal), the experimental results are shown in Table 4.

Table 4: HBMC-ABAC validity test

Total users	Passed users	Passed legitimate users	Passed illegal users	Correct rate
1000	1000	495	505	100%
2500	2500	1266	1234	100%
5000	5000	2582	2418	100%
7500	7500	3740	3760	100%
10000	10000	5050	4950	100%

Table 4 shows the total number of users in each experiment, the number of passed users, the number of legitimate users passed, and the number of illegal users passed, based on which the correct rate of each experiment can be calculated (correct rate = number of passed users/total users). According to the data in Table 4, with the increase of the total number of users, the number of users passed in each experiment is always equal to the total number of users, that is, the correct rate of the HBMC-ABAC model is 100%. And the ratio of legal users to illegal users is close to 1:1, which is in line with the expected results of the experiment. The experimental results show that the HBMC-ABAC model can realize the access control function based on user attributes, and the correct rate is 100%.

6 Conclusion

Aiming at the challenge of access control technology in big data environment, this paper studies the shortcomings of open source component Hadoop access control component. Based on ABAC model, this paper proposes an attribute-based access control model suitable for Hadoop platform. Each module in the HBMC-ABAC model is realized. Finally, the correctness of the model is verified and the time-consuming of its access control is tested. This scheme realizes the access control based on the attribute level, and can realize the dynamic change of permissions according to the dynamic attributes, and can complete the authorization of the access control system more flexibly and accurately. In addition, aiming at the problem of matching efficiency of traditional ABAC strategy, the binary mapping code is proposed at the same time, which improves the matching efficiency of traditional ABAC strategy set and is more suitable for the environment scene of big data. However, this model still has some defects in solving the problem of strategy redundancy and conflict, which needs to be improved and perfected through in-depth analysis and research.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] B. Dhyani and B. Anurag, "Big data analytics using Hadoop," *International Journal of Computer Applications*, vol. 108, no. 12, pp. 1–5, 2014.

[2] A. Rudniy, "Data warehouse design for big data in academia," *Computers, Materials & Continua*, vol. 71, no. 1, pp. 979–992, 2022.

[3] N. Ragavan and C. Y. Rubavathi, "A novel big data storage reduction model for drill down search," *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 373–387, 2022.

[4] S. Das, B. Mitra and V. Atluri, "Policy engineering in RBAC and ABAC," in *From Database to Cyber Security*, vol. 11170, pp. 24–54, 2018.

[5] K. Yang, Q. Han and H. Li, "An efficient and fine-grained big data access control scheme with privacy-preserving policy," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 563–571, 2016.

[6] J. Zamite, D. Domingos and M. J. Silva, "Group-based discretionary access control in health related repositories," *Journal of Information Technology Research*, vol. 7, no. 1, pp. 78–94, 2014.

[7] P. Alexander, L. Pike and P. Loscocco, "Model checking distributed mandatory access control policies," *ACM Transactions on Information and System Security*, vol. 18, no. 2, pp. 1–25, 2015.

[8] C. Ruan and V. Varadharajan, "Dynamic delegation framework for role based access control in distributed data management systems," *Distributed and Parallel Databases*, vol. 32, no. 2, pp. 245–269, 2014.

[9] O. Standard, Extensible Access Control Markup Language (XACML) version 2.0. [Online] Avaliable: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[10] M. Kuhlmann, D. Shohat and G. Schimpf, "Role mining-revealing business roles for security administration using data mining technology," in *Proc. of the Eighth ACM Sym. on Access Control Models and Technologies*, pp. 179–186, 2003.

[11] I. Molloy, Y. Park and S. Chari, "Generative models for access control policies: applications to role mining over logs with attribution," in *Proc. of the 17th ACM Symp. on Access Control Models and Technologies*, pp. 45–56, 2012.

[12] P. C. Cheng, P. Rohatgi and C. Keser, "Fuzzy multi-level security: An experiment on quantified risk-adaptive access control," *IEEE Symp. on Security and Privacy*, pp. 222–230, 2007.

[13] L. Zhang, A. Brodsky and S. Jajodia, "Toward information sharing: Benefit and risk access control," *Seventh IEEE International Workshop on Policies for Distributed Systems and Networks*, pp. 46–53, 2006.

[14] K. Zheng and W. Jiang, "A token authentication solution for hadoop based on kerberos pre-authentication," in *2014 Int. Conf. on Data Science and Advanced Analytics*, pp. 354–360, 2014.

[15] C. Feng, Z. Qin, D. Yuan and Y. Qing "Key techniques of access control for cloud computing," *Acta Electonica Sinica,* vol. 43, no. 2, pp. 312–319, 2015.