Hindawi
Security and Communication Networks
Volume 2021, Article ID 5582921, 12 pages
https://doi.org/10.1155/2021/5582921

Research Article

An Attribute-Based Access Control Policy Retrieval Method Based on Binary Sequence

Ruijie Pan, Gaocai Wang, and Man Wu

School of Computer and Electronic and Information, Guangxi University, Nanning, China

Correspondence should be addressed to Gaocai Wang; wanggcgx@163.com

Received 20 January 2021; Revised 8 February 2021; Accepted 21 March 2021; Published 5 April 2021

Academic Editor: Zhe-Li Liu

With the widespread application of new technologies, fine-grained authorization requires a large number of access control policies. However, the existing policy retrieval method applied to a large-scale policy environment has the problem of low retrieval efficiency. Therefore, this paper proposes an attribute access control policy retrieval method based on the binary sequence. This method uses binary identification and binary code to express access control requests and policies. When the policy is retrieved, the appropriate group is selected through the logical operation of the access control request and the policy binary identification. Within the group, the binary code of the access control request is matched with the binary code of all rules to find suitable rules, thereby reducing the number of matching attribute-value pairs in the rule and improving the efficiency of policy retrieval. Experimental results show that the policy retrieval method proposed in this paper has higher retrieval efficiency.

1. Introduction

The rise and development of new technologies such as cloud computing and the Internet of Things provide us with convenient data sharing, integrated computing, and other services and improve the efficiency of data processing, making full use of computing and storage resources. IoT devices increasingly prefer to synchronize all data resources to the cloud [1]. These resources contain a large amount of user privacy data, but the protection of this information by relevant agencies is not satisfactory [2]. Once such private information is leaked, it may cause immeasurable losses to individuals and organizations. For example, the private information of nearly 100,000 Westpac customers was leaked, and the data of 49 million Instagram users was exposed. These are all due to illegal access by illegal users and unauthorized access to resources by legitimate users. In recent years, as an effective means to prevent users from illegally accessing resources, access control technology has attracted the attention of domestic and foreign researchers. The object resources are protected by standardizing and restricting access to the requestor by verifying the visitor's identity information and establishing appropriate policies. This ensures that legitimate users can access and use resource services in complex network environments, while preventing illegal users from stealing and abusing resources and illegal access by legitimate users.

As the number of users increases and the access environment changes dynamically, discretionary access control, mandatory access control, and role-based access control [3] are no longer suitable for the current dynamic and real-time network environment due to their lack of dynamics and fine-graininess. However, because of its fine-grained and flexible nature, attribute-based access control (ABAC) [4] is an ideal access control solution for application scenarios such as cloud computing and the Internet of Things. Although the existing ABAC model has a relatively large advantage in the expression and formulation of security policies, the application of the ABAC model to a computing environment with a huge amount of information will have the problem of low retrieval efficiency. According to the related theories of ABAC and XACML (extensible Access Control Markup Language), when a user initiates a request to access object resources, the policy decision point will call related attributes to analyze and match all policies. This means that the evaluation of XACML-based access control policies needs to traverse all policies to make judgments, and attribute matching requires comparing all attribute information of the

access control request with the attribute information of the rules in the policy set. All the above information matching includes string matching and numerical comparison. Only when all the attribute information of this rule matches the attributed-based access request (AAR) will the subject be granted the privilege to access the object. In addition, when the last attribute of the rule is detected, the rule is found to be inappropriate. If there are too many such rules, it will cause a waste of retrieval time.

This paper proposes a method of attribute access control policy retrieval based on binary sequence to improve the efficiency of policy retrieval. The rest of this paper is organized as follows: Section 1 describes the related research based on attribute access control. Section 2 introduces the related theories of ABAC. Section 3 describes the retrieval method of attribute access control policy based on the binary sequence. Section 4 focuses on the experiments and analysis. Finally, Section 5 presents the conclusion.

2. Related Work

ABAC is an access control model that allows or denies user access based on the access control policy formulated by the administrator. The successful deployment of an attribute-based access control model requires two key parts. The first is a well-defined policy description language, and the second requires an effective policy retrieval method for attribute-based access control. A good policy description language helps prevent cloud platform resource data leakage and unauthorized access, and an effective policy retrieval method can ensure that access control requests are responded to in a time when they arrive. Regarding how to define a good access control authorization policy description language, scholars have studied different access control models and application scenarios and proposed a universal access control policy language XACML [5], a graph-based application for web services policy visual description language [6], and policy description language RestPL [7] suitable for RESTful interface and so on. Among them, the most commonly used is XACML, which is suitable for ABAC models, but in actual cloud deployment, each enterprise prefers its own policy description language. In order to make the policy language suitable for different access control models, Luo et al. [8] proposed a metamodel-based access control policy description language PML and implementation mechanism PML-EM. The policy description language supports multiple access control models such as RBAC and ABAC and has better flexibility. Matthew et al. [9] proposed a rule-based ABAC policy mining algorithm for the moderate problem of privilege. This algorithm can generate the least privilege ABAC policies that balance between underprivilege and overprivilege assignment errors.

For some large organizations, the scale of ABAC policies is getting bigger and bigger, and the number of users making access requests at the same time is also increasing. Therefore, effective retrieval of these policies is essential for real-time response to user access control requests and directly affects user experience. On how to effectively retrieve these policies, researchers have achieved some results. These research work can be roughly divided into two categories. The first category is an improvement of the policy retrieval method based on the XACML standard for the policy access control language, and the second category is an improved policy retrieval method based on the next-generation access control NGAC [10] standard.

Since XACML's policy description language was proposed in 2003, there have been many studies on how to improve the efficiency of policy retrieval. In the retrieval process of the traditional policy retrieval method, when there is an access control request, all policies need to be traversed. If there are policy redundancy and conflicts, XACML supports four combination algorithms which are (1) permit-override, (2) deny-overrides, (3) first applicable, and (4) only-one-applicable [11]. Li et al. [12] proposed a retrieval method with a priority policy for access control. This method establishes an associated policy table for each access object, adopts the optimization principle of space for time, and has high retrieval efficiency. However, when the number of access objects is huge, this method needs to establish a large number of policy tables, and it is relatively difficult to establish and maintain the policy tables. Zhou et al. [13] proposed a policy retrieval method based on a prefix-based tag, which adds binary prefixes based on policy attributes to narrow the scope of policy retrieval. This retrieval method does improve retrieval efficiency. However, when the access control policy matches the attribute name of the access request, this prefix calculation causes a waste of time. In response to this problem. Liu et al. [14] added binary identification based on the attribute access control policy and then introduced a policy decision tree at the attribute value level to improve retrieval efficiency. This method has good scalability, but it has certain limitations when there are many attributes. Huang et al. [15] conducted research on the basis of [13], grouping according to the first three subject attribute values of the policy prefix and further narrowing the search scope of the policy to improve retrieval efficiency.

It is mentioned in the next-generation access control standard [16] that the policy decision point is responsible for retrieving the rules that meet the access control request in the access control policy. When performing policy retrieval, it is necessary to compare the attribute information of the access control request with the attribute information of each rule in the access control policy until a matching rule is found. This is undoubtedly time-consuming. To this end, Nath et al. [10] proposed a data structure called PolTree, which uses two variants, PolTree and N-PolTree, to store ABAC policies and reduce the number of attribute-value pair comparisons during policy retrieval. Therefore, the retrieval efficiency of the policy is improved.

Most of the policy retrieval methods mentioned above need to match the attribute value of the access control request with the corresponding attribute value of each access control rule when performing the policy retrieval. If every time the last attribute value of the rule is detected, it is found that it does not match the access control request, which will cause a waste of retrieval time. Therefore, this paper considers the processing of rule attributes and attribute values and proposes a new policy retrieval method. This method

performs binary identification on the policies, groups them according to the binary identification, forms several policy groups, and performs binary coding on the policies. When accessing, first, we perform binary identification and encoding on the access control request. Then, through the logical operation of AAR and policy, the appropriate group is selected to filter out a large number of irrelevant policies. Finally, the binary code value of the access control request is logically operated with the binary code value of the rule in the group to avoid a single match of the attribute information in the policy and improve the retrieval efficiency.

3. ABAC Model

3.1. ABAC Policy Model. ABAC is an access control model that decides whether to grant the subject access to the object based on whether the attribute information of the subject, object, environment, and privilege attributes conforms to the established access control policy. We refer to [4, 17] to make the following definitions of ABAC.

Definition 1 (ABAC). ABAC can be abstracted as a quadruple (S, O, E, P). The Subject (S) represents a person or some nonhuman entity (a device that can initiate an access request), $S = \{s_1, s_2, s_3,...,s_n\}$. Objects ($O$) are usually resources that are requested to be accessed by the subject $O = \{o_1, o_2,o_3,\ldots,o_m\}$. The environment ($E$) represents the context state when the visit occurs, which includes the location and time when the subject initiates the visit, $E = \{e_1, e_2, e_3,\ldots e_j\}$. Privilege (P) indicates the operations that the subject will perform on the object, including read, write, and modify. $P = \{p_1, p_2, p_3, \ldots, p_k\}$. Among them, n, m, j, and k are all greater than or equal to 1.

Definition 2 (attribute). Attributes are used to describe the characteristics of the subject, object, environment, and privilege. Attributes are represented by attribute-value pairs. We use SA, EA, OA, and PA to represent the subject attribute, object attribute, environment attribute, and privilege attribute, respectively. Attr(SA), Attr(OA), Attr(EA), and Attr(PA) represent the value range of subject attribute, object attribute, environment attribute, and privilege attribute, respectively. Avsa, Avoa, Avea, and Avpa represent the attribute-value pairs of the subject, object, environment, and privilege, respectively, namely, two-tuples (SA, attr(SA)), (OA, attr(EA)), and (PA, attr(PA)).

Definition 3 (policy). ABAC policies are represented by attributes, and attribute values can be divided into discrete and continuous types. The access control policy is defined as (permit, deny) ← (Avsa, Avoa, Avea, Avpa).

Access control rules stipulate the attribute information that users must have when they want to access a particular resource. It is the basic unit of policy and the smallest policy execution unit $p = \{r_1,r_2,r_3,...,r_k\}$.

3.2. ABAC Process. The access control mechanism is composed of four key parts, which are mainly responsible for the retrieval and management of policies, and the management of attributes. They cooperate with each other to complete the authorization process for access control requests. The four key service sites shown in Figure 1 are Policy Administration Point (PAP), Policy Information Point (PIP), Policy Decision Point (PDP), and Policy Enforcement Point (PEP). According to the types of operations performed by the ABAC system, it can be divided into two stages: the preparation stage and the execution stage [17]. The dotted line in Figure 1 is an extension of the original access control model. BI is binary identification; BC is binary coding. BI and BC are both for access control requests and policies. The functions of these two modules will be described in Section 3.

4. Attribute Access Control Policy Retrieval Method Based on Binary Sequence

4.1. Build Policy and Attribute AARs Based on Binary Identification. Binary identification of ABAC policies and AARs is as follows. This method counts all the attributes that appear in the access control policy set and arranges the attributes in a specific order of subject, object, environment, and privilege. Moreover, the values of the subject, object, environment, and privilege attributes are also arranged in a certain order to ensure the validity and uniqueness of the binary identification. Attributes are divided into category attributes and noncategory attributes [18]. Category attributes refer to some decision information attribute values, usually including {permit, deny}; each rule has only one decision attribute. Noncategory attributes refer to the attribute information that needs to be measured to make a decision, including subject attributes, object attributes, environment attributes, and privilege attributes. Binary identification and encoding are for noncategory attributes.

On cloud computing and Internet of Things platforms, users request services through their respective terminal devices [19]. Therefore, we can construct the attribute information shown in Table 1 and use attribute information to construct ABAC policies (as shown in Table 2). The dimension of the binary identifier is the number of nondirectory attributes; that is, the total number of all attributes in the access control policy is the number of bits of the binary identifier. For example, the maximum number of noncategory attributes in a single rule in Table 2 is 6, so 6 binary bits are used to encode the policy.

As mentioned in Section 2, ABAC's access control process is divided into a preparation phase and an execution phase. In the preparation phase, PAP needs to perform binary identification for each rule in the policy set. When performing a policy retrieval, only binary identification of the access control request is required, and then, a logical AND operation is performed with the binary identification of the policy. When the operation result is consistent with the binary identifier of the access control rule, other attributes of the rule are checked; that is, it is judged whether the attribute of the access control request conforms to the attribute information of the rule. In order to further reduce the policy retrieval time, grouping is carried out according to the binary identification of ABAC policy. As shown in Figure 2,

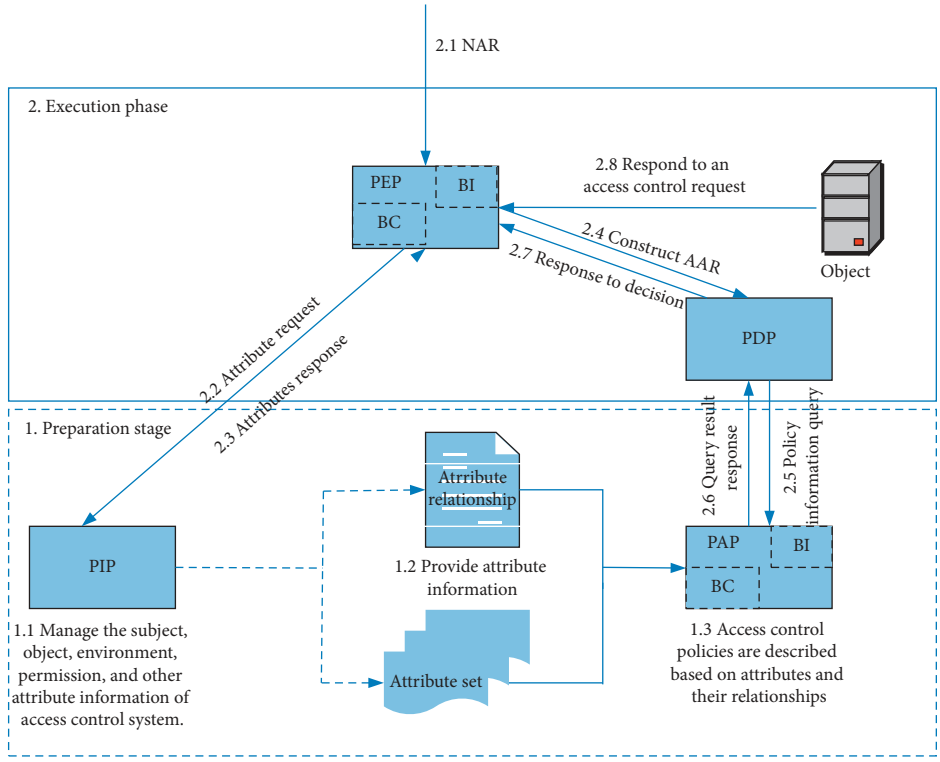

Figure 1: ABAC framework.

Table 1: Attribute combination.

SA		OA		EA	PA
SA_Role	SA_trust	OA_type	OA_trust	EA_Network	PA_permission
Student	Low	Personal	Low	Home	Delete
Teacher	Middle	Common	Middle	Work	Read
Admin	High		High	Public	Write

Table 2: Policy set.

Rule	attr(SA)	Attr(OA)	Attr(EA)	Attr(P)	Decide	Binary identification
R_1	Student low	Personal low	Home	Delete	Permit	111111
R_2	Low	Personal low	Public	Delete	Deny	011111
R_3	Student	Low	Work	Delete	Deny	100111
R_4	Student	Low	Home	Delete	Permit	100111

the root node represents all rule information of the policy set. Each node except the root node of the first level represents a group. Rules with the same binary identifier are in the same group, and rules with different identifiers are in different groups. The number of rule groups is determined by the complexity of the policy. If the complexity is n, the rule set can be divided into 2^n-1 groups at most.

The attributes of the access control rule represent the requirements for the access control request; that is, to successfully match the access control rule, the access request needs to have the attributes and attribute values required by the rule. This means that the matching of access control requests and rules is not a completely consistent match. In fact, the access control policy only detects the attributes required by the rules in the access request, and the additional attributes in the access request do not affect the success of its matching. Assuming that an access control request is (SA_Role = "student," SA_trust = "low," OA_type = "personal," OA_trust = "low,"t EA_Network = "public," PA_permisssion = "delete"), according to the matching rules, R_2 in Table 2 matches the access control request.

4.2. Binary Encoding of Attribute Information. Binary identification and grouping of ABAC policies can limit the scope of the policy to several groups that conform to the binary identification. A large number of irrelevant rules can be filtered out, thereby shortening the time for policy retrieval. However, when retrieving in this group, it is necessary to match the value of each attribute of each rule

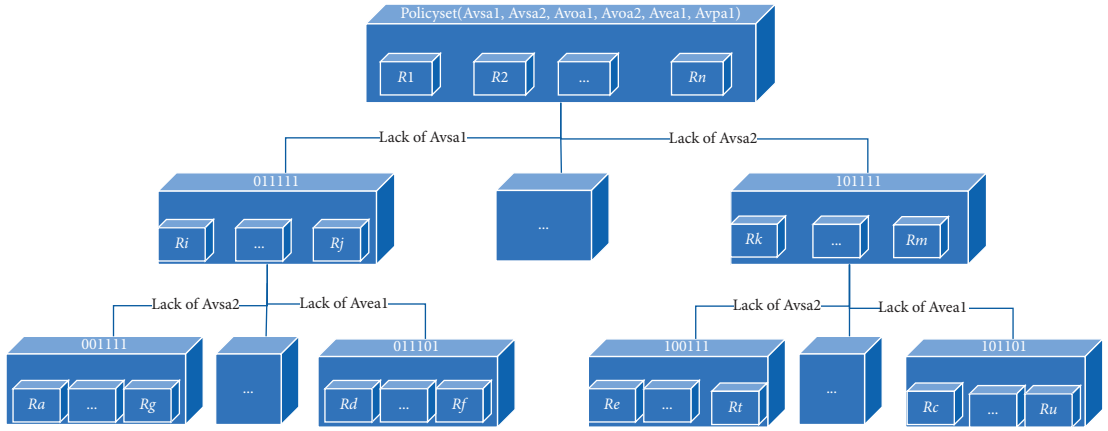

Figure 2: Policy set grouping.

with the corresponding value of the access control request and check whether the access control request satisfies the rule. The worst case is that every time the last attribute of the rule is detected, the rule does not match, which will cause a waste of time. If there are too many such rules, it will take too long to retrieve the policy. For this reason, we propose to binary code the access control policy of each group. In this way, only the binary code based on the attribute access control request and the binary code based on the attribute access control policy need to be logically operated to make an authorization decision, and there is no need to judge each attribute value of each rule in the policy set. Therefore, this retrieval method saves the time of policy retrieval and improves the efficiency of policy retrieval.

The attribute value based on attribute access control consists of two types, namely, discrete attribute value and continuous attribute value. For example, the value range of the attribute value of the subject attribute SA_Role mentioned above is {student, teacher, admin}, which is a discrete attribute value. In a specific access control environment, it is necessary to specify users to access resources in a certain interval, which requires setting continuous attributes. The process of mapping discrete attribute values and continuous attribute values into a binary sequence is called binary encoding of attribute values.

4.2.1. Discrete Attribute Value Coding. The values of discrete attributes are independent of each other, and the range of values can be represented by enumeration. We use dummy variable encoding to encode discrete attribute values, and dummy variable encoding uses a smaller dimension to represent the value of the attribute. If there are M types of discrete variables, the dummy variable coding can represent M possible values using only the M-1 dimension.

Definition 4 (attribute encoding dimension). Different subjects, objects, environments, and privilege entities have different attributes, and different attributes have different attribute values. NUM (s_i), NUM (o_i), NUM (e_i), and NUM (p_i) represent the number of attributes owned by s_i, o_i, e_i, and p_i, respectively. NumV (NUM(sa_i)), NumV(NUM(oa_i)), NumV(NUM(ea_i)), and NumV(NUM(pa_i)) represent the number of attribute values owned by subject attribute name sa_i, object attribute name oa_i, environment attribute name ea_i, and permission attribute name pa_i, respectively. The dimension $V_{s_i}V_{o_i}$, V_{e_i}, and V_{p_i} required to encode subject s_i, object o_i, environment e_i, and privilege p_i can be identified as

$$V_{s_i} = \sum_{i=0}^{n} \mathrm{Num}V\left(\mathrm{Num}\left(sa_i\right)\right) - \mathrm{Num}\left(s_i\right), \tag{1}$$

$$v_{o_i} = \sum_{i=0}^{m} \mathrm{Num}V\left(\mathrm{Num}\left(oa_i\right)\right) - \mathrm{Num}\left(o_i\right), \tag{2}$$

$$V_{e_i} = \sum_{i=0}^{c} \mathrm{Num}V\left(\mathrm{Num}\left(ea_i\right)\right) - \mathrm{Num}\left(e_i\right), \tag{3}$$

$$V_{p_i} = \sum_{i=0}^{d} \mathrm{Num}V\left(\mathrm{Num}\left(pa_i\right)\right) - \mathrm{Num}\left(p_i\right), \tag{4}$$

where n, m, c, and d are the number of attributes contained in s_i, o_i, e_i, and p_i. The dimension of the binary identifier required by the encoding rule is $V_{\text{total}} = V_{s_i} + V_{o_i} + V_{e_i} + V_{p_i}$. Assuming that s_i has two attributes, sa_1 and sa_2, the value range of sa_1 is $\{sa_1v_0, sa_1v_1, sa_1v_2, \Phi\}$, and the value range of sa_2 is $\{sa_1v_0, sa_1v_1, \Phi\}$, and the dimension required to encode s_i is $4 + 3 - 2 = 5$.

Definition 5 (attribute encoding rules). Arrange the attribute names of all attributes in a specific order, and the attribute values contained in the attributes also need to be arranged in a specific order to form a large attribute array. If a specific attribute value appears in the rule, the corresponding position is set to 1.

If the subject, object, environment, and privilege attributes appear in a rule at the same time, the attributes of the rule are arranged in a specific order of subject, object, environment, and privilege attributes. If the attribute value ha_iv_j appears in the rule, the Pth position of V_{total} is set to 1 according to

$$P = \begin{cases} \mathrm{Num}V\left(\mathrm{Num}\left(sa_1\right)\right) + ... + \mathrm{Num}V\left(\mathrm{Num}\left(sa_{i-1}\right)\right) + j \quad ha_i v_j \in SA, \\ V_{s_i} + \mathrm{Num}V\left(\mathrm{Num}\left(oa_1\right)\right) + ... + \mathrm{Num}V\left(\mathrm{Num}\left(oa_{i-1}\right)\right) + j \quad ha_i v_j \in OA, \\ V_{s_i} + V_{o_i} + \mathrm{Num}V\left(\mathrm{Num}\left(ea_1\right)\right)) + ... + \mathrm{Num}V\left(\mathrm{Num}\left(ea_{i-1}\right)\right) + j \quad ha_i v_j \in EA, \\ V_{s_i} + V_{o_i} + V_{e_i} + \mathrm{Num}V\left(\mathrm{Num}\left(pa_1\right)\right) + ... + \mathrm{Num}V\left(\mathrm{Num}\left(pa_{i-1}\right)\right) + j \quad ha_i v_j \in PA. \end{cases} \tag{5}$$

The encoding of subject s_i according to the encoding rules is shown in Table 3.

4.2.2. Continuous Attribute Coding. For continuous attribute values that cannot be enumerated, we first discretize them and then encode them. As shown in Figure 3, assuming that the attribute named sa_3 is a continuous attribute, the discretization process of sa_3 is as follows.

(1) Traverse all the rules, divide the value range of sa_3, remove duplicate attribute values, and arrange them in ascending order. Suppose we get{sa_3v_1, sa_3v_2,..., sa_3v_n}.

(2) Remove the smaller attribute value that obtains the same access privileges of the same object in the rule, and keep the largest attribute value. For example, if the attribute values sa_3v_1, sa_3v_2, and sa_3v_3 have read privilege for the same object resource content at the same time, sa_3v_1 and sa_3v_2 are deleted and sa_3v_3 is retained. Assume that the set of these largest attribute values is {sa_3v_3, sa_3v_9,..., sa_3v_n}.

(3) Set labels L_1, L_2,...,L_m with sa_3v_3, sa_3v_9,..., sa_3v_n as the thresholds, respectively, where m is the number of elements in {sa_3v_3, sa_3v_9,..., sa_3v_n}. The value ranges of L_1, L_2,..., L_m are {0, sa_3v_3}, {sa_3v_3, sa_3v_9},..., {sa_3v_g, sa_3v_n}.

(4) Use L_1, L_2,...,L_m to replace the attribute value of sa_3 in the rule.

After the continuous attribute value is converted to the discrete attribute value, the encoding method is completely consistent with that of the discrete attribute, and the description will not be repeated here.

4.3. Analysis of Policy Retrieval Method Based on Binary Sequence. We make the following definitions for the retrieval rules of the binary sequence-based policy retrieval method.

Definition 6 (group selection principle). Perform a logical AND operation based on the binary identification of the attribute access control request and the group number of the rule grouping. If the calculation result is consistent with the group number, the group is a suitable group; otherwise,e the group is not suitable.

Definition 7 (rule selection principle). Perform logical AND operation based on the binary code of the attribute access control request and the rule binary code. If the result is consistent with the binary code of the rule, the rule is the rule to be found; otherwise, the access control request does not comply with the rule.

In order to further describe the policy retrieval method mentioned in this article, we take the policy set in Table 2 as an example, select the attribute values in Table 1 to construct the attribute-based access control request, aar_1 = {SA_role = student, SA_trust = low, OA_trust = low, EA_Network = -work, PA_permission = delete}, and complete the retrieval process.

In the access control preparation phase, the preparation work that PAP needs to complete is as follows: the results of the binary identification and coding of the rules in Table 2 according to the coding rules proposed in Section 4.2 are (R_1, 111111, 10010010100100100), (R_2, 011111, 00010010100001 100), (R_3, 100111,10000000100010100), and (R_4,100111, 10000000100100100). The group numbers of R_1 and R_2 are 111111 and 011111, respectively. The group number of R_3 and R_4 is 100111, and they are in the same group.

The access control execution phase is divided into two phases: processing the attribute-based access control request and retrieving the rule set. Processing of access control requests is as follows: the result of the binary identification of aar_1 according to the binary identification principle proposed in Section 4.1 is 110111; according to the encoding method mentioned in Section 4.2, the result of the binary encoding of aar_1 is (aar_1, 10010000100010100).

Retrieve the rule set: the binary identification of aar_1 and the group number AA are logically ANDed according to the principles defined in Definition 6, namely, 110111 and 111111 = 110111≠111111, indicating that aar_1 does not meet the attribute combination requirements of the group. Query the group with the group number 011111, that is, 110111 and 011111 = 010111≠011111, so aar_1 does not meet the attribute combination requirements of the group, and the group is filtered out. Query the rule group with the group number 100111, 110111, and 100111 = 100111, indicating that this group meets the requirements, which needs further retrieval. The result of logical AND operation between the binary code of the R3 rule and the binary code of the access control request is equal to the binary code of the rule, that is, 10000000100010100 and 10010000100010100 = 1000000010 0010100. According to Definition 5, this rule meets the requirements. Because R_3's decision attribute is deny, the decision result of the PDP will be to deny this access, and this policy retrieval ends. The specific policy retrieval process is shown in Figure 4.

In the preparation phase, PAP performs binary identification and coding on all access control policies and groups them according to the binary identification. When there is an access control request, the PEP obtains the attribute information from the PIP to construct the AAR and performs binary

Table 3: Binary encoding of attributes.

Subject attribute value range	Binary code
$attr(sa_1) = \{sa_1 = sa_1v_1\}$ $attr(sa_2) = \{sa_2 = sa_1v_0\}$	01010
$attr(sa_1) = \{sa_1 \neq sa_1v_1\}$ $attr(sa_2) = \{sa_2 = sa_2v_0\}$	01101
$attr(sa_1) = \{sa_1 = \{ sa_1v_1, sa_1v_2 \}$ $attr(sa_2) = \{sa_2 = sa_2v_0\}$	01110
$attr(sa_1) = \{sa_1 = sa_1v_1\}$	00010
$attr(sa_2) = \{sa_2 = sa_2v_0\}$	01000

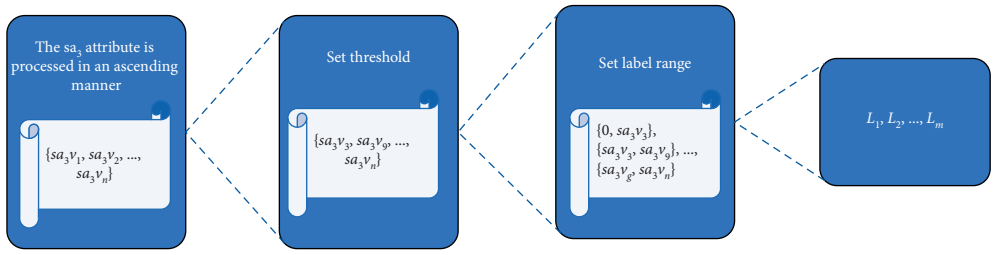

Figure 3: Continuous attribute discretization process.

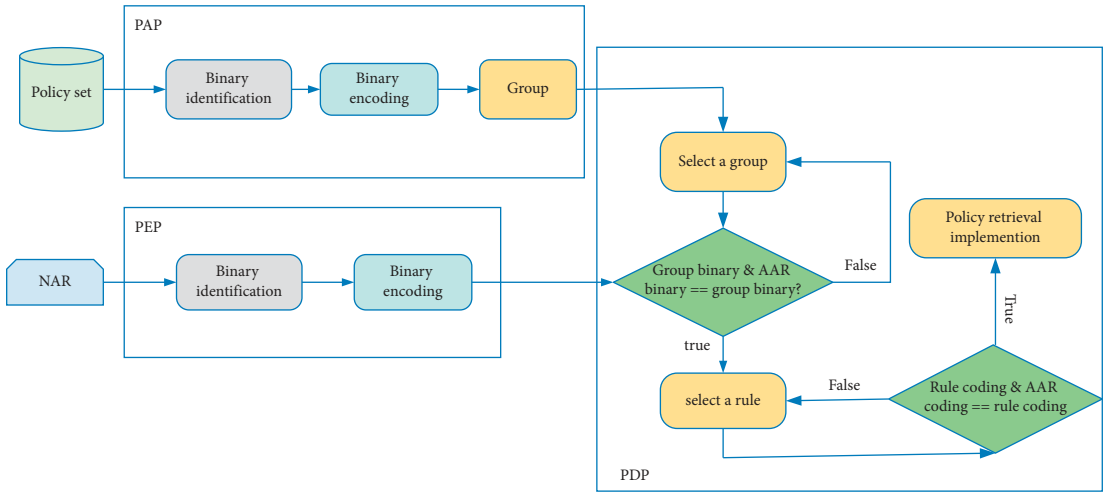

Figure 4: Policy retrieval mechanism based on the binary sequence.

identification and encoding of the AAR. The PDP selects a group from the rule group to make a logical AND operation of its binary identifier and the binary identifier of the access control request and judges whether the result is equal to the binary identifier of the AAR. If they are not equal, then the binary identification of the next group is judged. If they are equal, make the binary code of each rule in the group and the binary code of the access control request do a logical AND operation, and judge whether it is equal to the binary code of the rule. If they are equal, it indicates that the rule is the rule to be searched; otherwise, the binary code of the next rule in the group is logically operated.

The core Algorithm 1 of the attribute access control policy retrieval method based on binary sequence is as follows.

5. Experimental Results and Analysis

In order to evaluate the efficiency of the retrieval method based on the binary sequence policy, this paper uses the C++ language to write the test code in Qt Creator on the Win10 platform and uses MATLAB as the data analysis tool. Evaluating the retrieval method proposed in this paper requires a large number of ABAC policies and AARs. Unfortunately, due to the confidentiality of the access control policy, we cannot obtain real industry data. Therefore, referring to the attribute information mentioned above, a simple policy and AAR generator were implemented, and 4000 policies and 2000 AARs were generated. In order to ensure the validity of binary identification, we refer to the data classification method in [18] to standardize policies and access control requests. The experiment in this paper uses these data as the basis to test the efficiency of the binary sequence-based attribute access control strategy retrieval method in terms of policy preprocessing, policy evaluation time, and total strategy retrieval time. At the same time, in order to ensure the generality of the test results, the data obtained in the following experiments are the results of performing 10 experiments and averaging them. The experimental environment is as follows: CPU: 11th Gen Intel(R)Core(TM)i5-

```
    Input: AAR
    Output: the decision
    //: Binary identification and encoding of AAR
(1) Init(AAR.policy_groupbinary);
(2) Init(AAR.policy_code);
(3) for(i = 0;i < AAR.policy_groupbinary.size(); i++) do
(4)    if(AAR[i]) exist then
(5)       AAR.policy_groupbinary.setposition(p) = 1;
(6)    end if
(7) end for
(8) for(j = 0; j < AAR.policy_code.size(); j++) do
(9)    if(AAR[j].attrvale) exit then
(10)      a = getposition(AAR[j].attrvale);
(11)      AAR.policycode.setposition(a) = 1;
(12)   end if
(13) end for
    //policy retrieval
(14) for(k = 0; k < Group.size; k++) do//
(15) if (Group[k].groupbinary&AAR.groupbinary) = = Group[k].groupbinary) then
(16)    for(g = 0; g < Group[k].size; g++) do
(17)       if (Group[k].ruleset[g].policycode&AAR.policy_code = = (Group[k].ruleset[g].policycode) then
(18)          Dodecide();
(19)          break;
(20)       end if;
(21)    continue;
(22)    end for;
(23)    continue;
(24) end if
(25) end for
```

Algorithm 1: Policy retrieval algorithm based on binary identification.

1135G7@2.40GHZ 2.42 GHz,RAM:16.00 GB; Qt Creator 3.3.0(opensource) based on Qt 5.4.0(MSVC 2010, 32 bit), MATLAB version: 8.6.0.267246 (R2015b).

SG-PRM and AG-PRM in the following experiments refer to the prefix-based policy retrieval method mentioned in [13] and the attribute grouping-based access control policy retrieval method mentioned in [15]. B–S-PRM refers to the strategy retrieval method proposed in this article. Experimental comparisons are based on the same access control policy and access control request. Policy complexity refers to the number of attribute key-value pairs contained in each rule in the policy. The complexity of the policy used in all experiments is 6.

5.1. Policy Preprocessing Time. The policy preprocessing time is completed in the preparation phase of the access control framework diagram in Figure 1, mainly deploying policy attributes. The policy preprocessing time of the policy retrieval method proposed in this paper refers to the time that PAP adds binary identification, grouping, and coding to each rule in the policy set. SG-PRM performs binary identification of the policy in the policy preprocessing stage. AG-PRM performs binary identification and grouping of policies in the policy preprocessing stage. It can be seen from Figure 5 that when the number of policies is set to 500, 1000, 1500, 2000, and 2500, the retrieval time of B–S-PRM, SG-PRM, and AG-PRM policies all increases with the increase of the number of policies. Among them, the policy preprocessing time of the AG-PRM and SG-PRM policy retrieval methods is relatively close and lower than that of the B–S-PRM policy retrieval method. This is because the

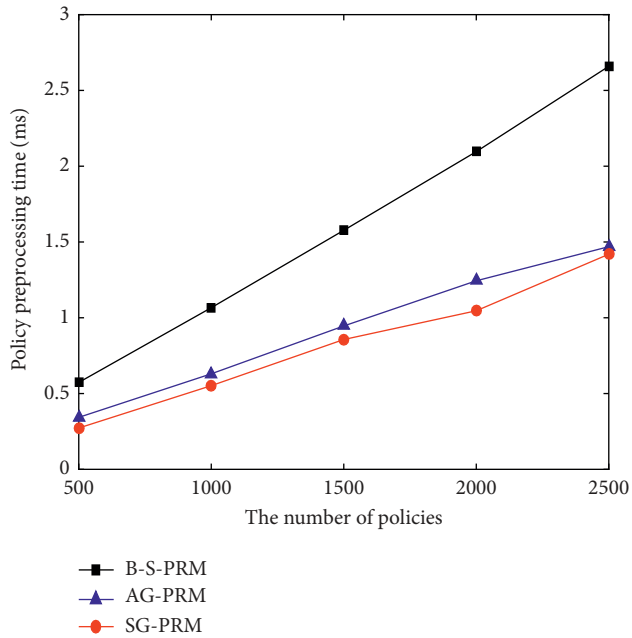

Figure 5: Policy preprocessing.

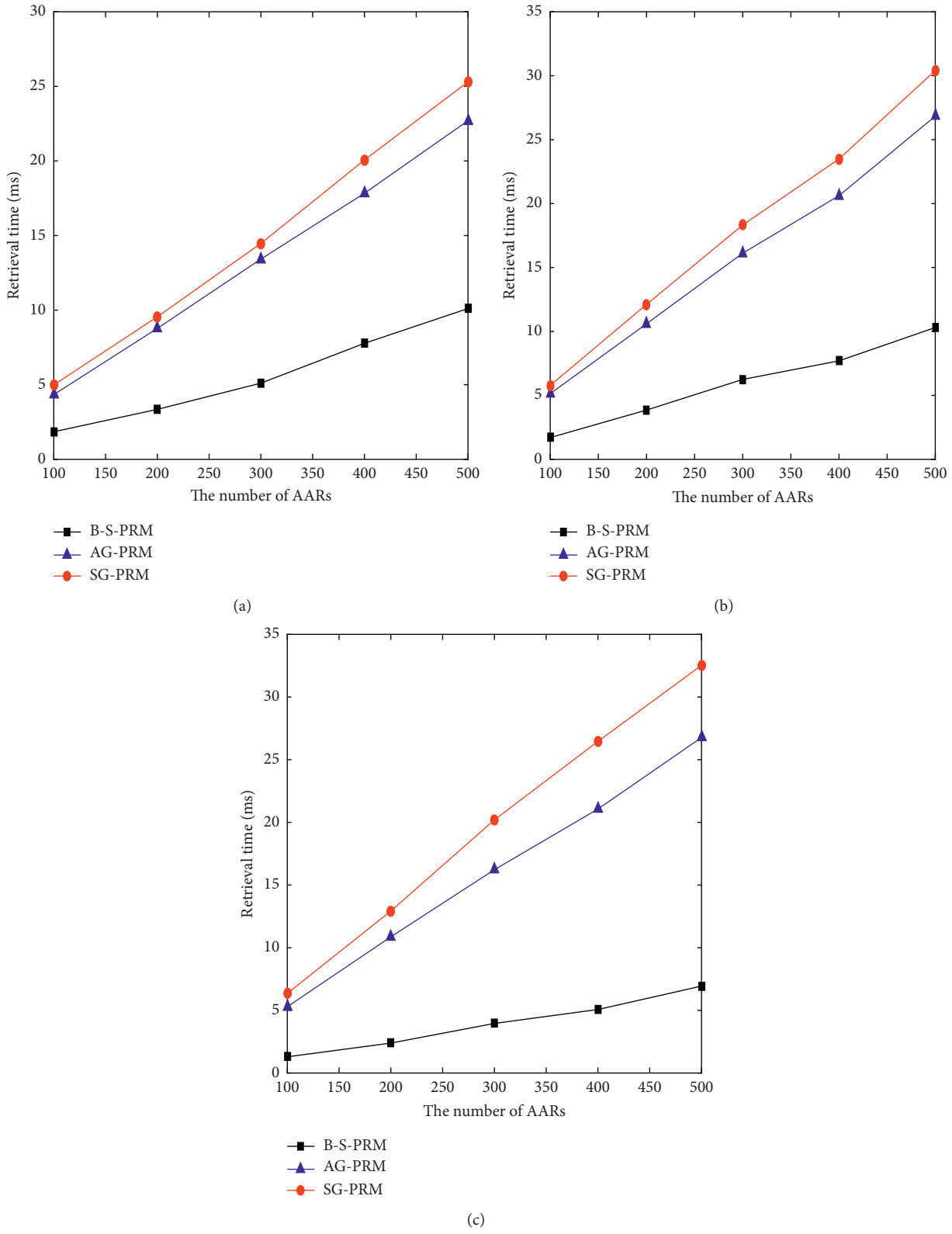

Figure 6: Changes in policy evaluation time. (a) The number of policies is 2000. (b) The number of policies is 2500. (c) The number of policies is 3000.

policy retrieval method proposed in this paper adds a binary encoding process to the first two policy retrieval methods. However, the preprocessing of the attribute-based access control strategy occurs in the preparation phase of the access control system. Therefore, the policy retrieval method in this article does not affect the user's real-time experience.

5.2. Policy Evaluation Time. Policy evaluation refers to the completion of the PDP in the execution phase in Figure 1. It mainly refers to the process of PDP retrieval in accordance with the rules of the AAR. Figures 6(a)–6(c) show the change of the policy evaluation time when the number of AARs increases under the condition of different numbers of policies. The policy complexity of this set of experiments is 6. The number of policies is 2000, 2500, and 3000, respectively. SG-PRM performs a logical OR operation between the prefix of the access control request and the prefix of the policy in the policy retrieval phase, and the result of the operation is consistent with the prefix of the policy and then matches whether the attribute information matches. AG-PRM performs attribute-based grouping of policy sets to reduce the scope of policy retrieval. It can be seen from Figures 6(a)-6(c) that under different access control policy conditions, as the number of access control requests increases, the policy evaluation time of these three policy retrieval methods gradually increases. Among them, the policy evaluation time of the strategy retrieval method (B–S-PRM) proposed in this paper shows a downward trend as the number of policies increases. When the number of strategies is 3000, the evaluation time of the BS-PRM strategy is about half of that when the number of strategies is 2000, and the fluctuation range of the policy evaluation time is gradually reduced, while the policy evaluation time of the other two policy retrieval methods has not changed significantly. The strategy evaluation time of SG-PRM fluctuates between 4 and 32 ms, the policy evaluation time of AG-PRM fluctuates between 4 and 27 ms, and the B–S-PRM fluctuates between 1 and 11. It can be seen that the retrieval time of the policy retrieval method proposed in this paper is more stable, and the policy retrieval efficiency of B–S-PRM is about 3 times that of SG-PRM and AG-PRM. This shows that the policy retrieval method proposed in this paper can be applied to high policy environments.

5.3. Total Retrieval Time. The total retrieval time based on the attribute access control policy includes the time for PEP to process AAR and the time for PDP to evaluate the policy. As shown in Figures 7 and 8 , the total number of policies is 3000. The complexity of this set of experimental policies is 6. With the increase in the number of access control requests, the processing time of the three policy retrieval methods for access control requests has gradually increased and is close to linear growth. The processing time of the policy is all milliseconds. Among them, the policy retrieval method proposed in this paper takes about twice the processing time of access control requests than SG-PRM and AG-PRM. This is because, in the process of access control, SG-PRM adds a prefix to the access control request, BS-PRM needs to binary code and identify the policy, and AG-PRM only needs to add the group identification and prefix identification to the access control request. However, it can be seen from Figure 8 that although the strategy retrieval method proposed in this article has a slower processing time for AAR in PEP, its total retrieval time has dropped significantly. Among them, the policy retrieval efficiency of B–S-PRM is about 4 times that of SG-PRM and 3.5 times that of AG-PRM. As the number of access control requests increases, the policy retrieval method proposed in this paper has more advantages and the strategy retrieval time becomes more stable. Because when the policy increases, unlike the other two policy retrieval methods, the attribute access control policy retrieval method

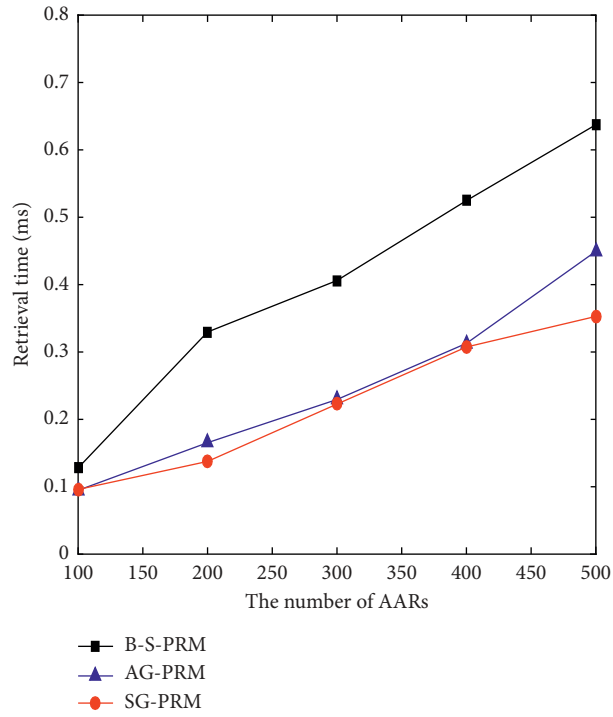

Figure 7: The processing time of the AARs.

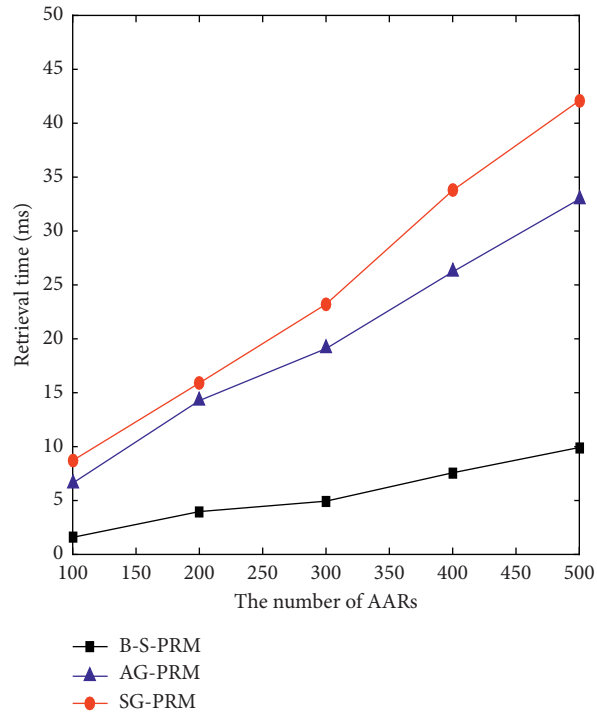

Figure 8: The total retrieval time.

based on binary sequence does not traverse the attribute-value pairs in the rule but chooses the appropriate grouping to perform logical operations on the binary code in the group. Therefore, the retrieval efficiency is greatly improved.

To sum up, although the policy retrieval method proposed in this paper takes longer than the other two policy retrieval methods in policy preprocessing time, the policy retrieval time is significantly shortened and the retrieval efficiency advantage is obvious. With the increase of the policy scale, the retrieval time of the policy retrieval method proposed in this paper has not changed significantly. In the deployment of attribute-based access control in cloud computing and Internet of Things environments, when a user's access control request comes, this method can shorten the user's waiting time and respond to the user's access control request in real time.

6. Conclusions

With the rise of cloud computing and Internet of Things technologies, today's computing environment is becoming more and more massive and dynamic, with a huge scale of users and resources, and real-time changes in the access environment of subject and object. ABAC is widely used as an effective technology to protect object resources. Users want to get the privilege to access object resources in a relatively short time. However, the existing retrieval methods based on attribute access control policies have certain deficiencies when applied to large-scale data. To solve this problem, this paper proposes an attribute access control policy retrieval method based on the binary sequence. This method uses one-hot and dummy variable encoding methods to perform binary identification and encoding of AAR and policies and group the policies according to the binary identification. AAR's binary identification and group binary identification perform a logical AND operation to select groups that meet the requirements and filter out a large number of irrelevant policies. Then, the binary code of the AAR and the binary code of the rules in the group do logical operations to select the appropriate rules, which reduces the process of matching the attributes of the rules in the policy set with the AAR attributes. The experimental results also verify that the policy retrieval method proposed in this paper has lower policy retrieval time and higher retrieval efficiency. [20]

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China under Grant no. 62062007.

References

[1] T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia, "Fog-based computing and storage offloading for data synchronization in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4272–4282, 2019.

[2] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, "A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4831–4843, 2019.

[3] K. R. S. Yaira, A. D. Steven, and S. B. Mohammed, "A service-based RBAC & MAC approach incorporated into the FHIR standard," *Digital Communications and Networks*, vol. 5, no. 4, pp. 214–225, 2019.

[4] D. Servos and S. L. Osborn, "Current research and open problems in attribute-based access control," *ACM Computing Surveys*, vol. 49, no. 4, pp. 1–45, 2017.

[5] X. Oasis, *eXtensible Access Control Markup Language XACML Version 3.0*, OASIS Standard, Burlington, MA, USA, 2013.

[6] D. Nabil, H. Slimani, H. Nacer et al., "ABAC conceptual graph model for composite web services," in *Proceedings of the 2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, pp. 36–41, Marrakech, Morocco, 2018.

[7] Y. Luo, Q. N. Shen, and Z. H. Wu, "A novel access control specification language and its permission classification," *Journal of Computers*, vol. 41, no. 6, pp. 969–986, 2018.

[8] Y. Lu, Q. N. Shen, and Z. H. Wu, "Access control policy specification language based on metamodel," *Journal of Software*, vol. 31, no. 2, pp. 439–454, 2020.

[9] M. W. Sander and Y. Chuan, "Mining least privilege attribute based access control policies," in *Proceedings of the 2019 Annual Computer Security Applications Conference (ACSAC '19)*, pp. 404–416, Tucson, AZ, USA, 2019.

[10] R. Nath, S. Das, and S. Sural, "PolTree: a data structure for making efficient access decisions in ABAC," in *Proceedings of the 24th ACM Symposium, ACM*, pp. 25–35, Toronto, Canada, 2019.

[11] M. Mejri, H. Yahyaoui, M. Azzam et al., "A rewriting system for the assessment of XACML policies relationship," *Computers & Security*, vol. 97, pp. 1–12, 2020.

[12] H. H. Li, M. F. Dong, and F. Fan, *A Retrieval Method Constructed By Access Control With Priority Policy*, China, 2018.

[13] J. S. Zhou and Y. S. Zhang, "Policy retrieval method based on sign," *Computer Engineering and Design*, vol. 36, no. 11, pp. 2943–2947, 2015.

[14] M. P. Liu, Y. Cheng, H. Li et al., "An Efficient attribute-based access control (ABAC) policy retrieval method based on attribute and value levels in multimedia networks," *Sensors*, vol. 20, no. 6, pp. 1–15, 2020.

[15] M. R. Huang and B. Ou, "Access control policy retrieval method based on attribute grouping," *Application Research of Computers*, vol. 37, no. 10, pp. 1–7, 2019.

[16] V. Hu, D. F. Ferraiolo, D. R. Kuhn, R. N. Kacker, and Y. Lei, "Implementing and managing policy rules in attribute based access control," in *Proceedings of the 2015 IEEE 16th International Conference on Information Reuse and Integration*, pp. 518–525, San Francisco, CA, USA, 2015.

[17] L. Fang, L. H. Yin, Y. C. Guo et al., "A survey of technologies in attribute-based access control scheme," *Chinese Journal of Computers*, vol. 40, no. 7, pp. 1681–1698, 2017.

[18] R. A. Shaikh, K. Adi, and L. Logrippo, "A data classification method for inconsistency and incompleteness detection in access control policy sets," *International Journal of Information Security*, vol. 16, no. 1, pp. 91–113, 2017.

[19] T. Wang, M. Z. A. Bhuiyan, G. Wang, L. Qi, J. Wu, and T. Hayajneh, "Preserving balance between privacy and data integrity in edge-assisted Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2679–2689, 2020.

[20] Z. Y. Zhao and L. Sun, "Attribute-based access control with dynamic trust in a hybrid cloud computing environment," in *Proceedings of the International Conference on Cryptography, Security and Privacy*, pp. 112–118, ACM, New York, NY, USA, 2017.