Linear Policy Recommender Scheme for Large-Scale Attribute-Based Access Control

Jing Wang(✉), Weijia Huang, Wenfen Liu, Lingfu Wang, and Mingwu Zhang

School of Computer Science and Information Security,
Guilin University of Electronic Technology, Guilin, China
wjing@guet.edu.cn

Abstract. In the large-scale data sharing platform, access control mechanism plays an important role in protecting data security and privacy. Significantly, Attribute-Based Access Control (ABAC) can support fine-grained access control, which would make the data sharing platform more flexible, efficient and manageable. However, in ABAC, data owner need to manually assign access policies for each data, which would incur lots of workload and limit the usability of the system. Thus, we propose a linear policy recommender scheme for ABAC in this work. Firstly, we propose a general form of access policy named linear policy, which describes the policy as a linear function. Comparing with other forms of policy, linear policy is more flexible and efficient. Secondly, we propose a matrix factorization based linear policy recommender scheme. The scheme learns a policy matrix and a security threshold vector from access logs and recommends the optimal linear policy for each data by a binary matrix factorization model. Intuitively, the policy matrix and security threshold vector can be viewed as the optimal linear policy of each data, which is helpful for ABAC to improve policy generation and management. Finally, sufficient experiments are given to present the performance of the proposed policy recommender system. The result shows that our policy recommender system is efficient and accurate in calculation.

Keywords: Data sharing · Attribute-based access control · Access policy · Matrix factorization

1 Introduction

With the development of Internet and cloud, data sharing services become more and more popular. Thus, the data security of such large-scale data sharing system arouses lots of concerns. Access control mechanism provides an efficient way to protect the data security and privacy for such systems. Different from traditional access control mechanism, Attribute-Based Access Control (ABAC) provides fine-grained access control which supports the complex access requirement of the large-scale data sharing system. In ABAC, the access privilege of each data

W. Shi et al. (Eds.): SPNCE 2021, LNICST 423, pp. 175–191, 2022.
https://doi.org/10.1007/978-3-030-96791-8_13

is described by an access policy which usually presented as a logical formula based on user attributes. A user is permitted to access a data iff the user attribute satisfies the access policy of the data. Since the flexibility of policy, ABAC is more flexible, efficient and manageable than traditional access control mechanisms.

However, in such a large-scale and complex system, the management of access policies will cost lots of workload. For instance, a data owner will cost a lot of time to deal the access policy for each data before he/she uploads the data to the center. Intuitively, such problem will seriously limit the application of ABAC. Additionally, considering of the personal security requirement, it is hard to assign an access policy with unified standards for each data, it requires the precise matching of data and policies. Thus, in this paper, we propose a policy recommender scheme for ABAC. The scheme can recommend a linear policy for each data, which considers the fine-grained data security requirement. The linear policy is a novel form of access policy, which presents as a linear function of user attributes. It is important that the linear policy can provide strong flexibility and computability.

One of the core challenges of our policy recommender scheme is presenting the formalized model to extract the access policy of each data from access logs. On the one hand, data owner is usually hard to give an accurate and comprehensive description of authorized users and unauthorized users in the large-scale system. On the other hand, it is hard to extract the differences of each data from the sparse access logs. Thus, it requires a generalized, simple, and intuitional expressing form of access policy, which presents interpretability and controllability. Next, we should give a formalized model to extract the information of access policies from access logs.

In this paper, we propose a matrix factorization based linear policy recommender scheme for the ABAC mechanism, which can extract the fine-grained data access privilege from access logs and present as the linear policy for each data. First, in the proposed recommender scheme, access logs are transformed into a user-data access matrix. Then, a matrix factorization algorithm is provided to estimate the access privilege of each data for each user. In fact, the algorithm outputs two factored matrices named policy matrix and user attribute matrix, respectively. It implies that the policy matrix is used to describe the access policy of each data and the user attribute matrix is used to describe the attribute feature of each user. Meanwhile, it is reasonable to simplify the access matrix and user attribute matrix as binary matrix in this work. In fact, there are only two case of a access matrix element: *unaccessable* denoted by 0 and *accessable* denoted by 1, respectively. Similarly, in this work, we only consider nunnumerical user attribute which can be easily translated into a binary vector by one hot encoding. Thus, the attribute threshold and security threshold are introduced to binarize the predicted access matrix and the user attribute matrix, respectively. Finally, the recommender scheme outputs a linear policy for each data. The linear policy is a novel form of policy proposed in this work, which presents the policy as a linear function of user attributes. Furthermore, each linear policy is assigned a security threshold to distinguish its security hierarchy. Thus, the linear policy is flexible and manageable cause of the efficient linear function and the intuitional security threshold. In brief, the contributions of this paper can be summarized as follows:

1. A novel kind of access policy, named linear policy, is provided to describe access privileges, which presents a linear relation between user attributes and access privileges;
2. A matrix factorization based policy recommender scheme is proposed to extract the policy of each data, which considers the fine-grained security requirement of data;
3. Lots of experiments demonstrate the proposed linear policy recommender scheme is efficient and accurate.

2 Related Work

2.1 Access Control

The access control mechanism protects and prevents data from unauthorized access. Discretionary Access Control (DAC) and Mandatory Access Control (MAC) are two early mechanisms of access control that were applied in Department of Defense applications in the 1960s and 1970s. With the growth of networks, DAC and MAC are hard to satisfy the access control with complex access requirements. Role-based Access Control (RBAC) realizes the flexible access control based on the predefined roles assigned to users by the system administrator [1]. However, RBAC only realizes the coarse-grained access control and lacks of dynamics, which is not suitable for the current dynamic and real-time network environment. ABAC permits or denies a user's access request to a specific data based on the attribute associated with the user and the data [2,3]. It is a suitable access control mechanism for the current network environment, because ABAC realizes the dynamic and fine-grained access control.

Because of the advantage of ABAC, transforming the traditional access control mechanism to the ABAC mechanism is attractive. ABAC policy mining has become an important approach for migrate from a traditional access control mechanism to an ABAC mechanism. Xu et al. [4] proposed a policy mining algorithms for different ABAC mechanisms. Their algorithms realize the policy mining with positive authorization. Iyer et al. [5] considered the importance of negative authorization and proposed a policy mining algorithm to support negative authorization. However, those algorithms are not suitable to mine policy from multiples access control mechanisms that have different policy frameworks. Thus, Karimi et al. [6] proposed an unsupervised learning based approach for mining ABAC policies which clusters the access right tuples based on the similarity and extracts policies for clusters. Additionally, Bui et al. [7] considered part of the attribute values of some entities are unknown, and proposed an ABAC policy mining algorithm with unknown values.

Except the mechanism migration, policy mining also uses to solve the inherent flaws of the ABAC mechanism in the policy assignment phase. In the ABAC mechanism, the number of user attributes and the complexity of policy definition are increasing with the size of the network. In order to reduce the number of user attributes in the large-scale system and increase the flexibility of the ABAC mechanism, Benkaouz et al. [8] proposed a policy mining algorithm based on k-nearest neighbors. In addition, over-privilege and under-privilege are

two kinds of assignment errors in the access control mechanism. Over-privilege grants the privilege to the unauthorized user that increases the security risk of the access control mechanism. Under-privilege denies the access request from the authorized user that reduces the availability of the access control mechanism. Thus, Sanders et al. [9] proposed a rule mining approach to create least privilege ABAC policies which achieves a balance between minimizing over-privilege and under-privilege assignment error.

In this paper, we propose an access policy recommender scheme to extract access policies for each data which considers both the fine-grained security requirement and the user attribute. In addition, we propose a linear policy model that consists of a policy vector and a security threshold. It is easy to adjust the security level of data by the magnitude of the security threshold.

2.2 Recommender System

Recently, there are a lot of recommender systems have been proposed, which usually generate personalized recommendations for users. It has been widely used in various real scenes, such as restaurant recommendations [10], shopping recommendations [11], movie recommendations [12], and so on. The neighborhood approach is a kind of collaborative filtering technique for recommender system which focus on relationships between items or between users [13]. Su et al. [14] designed a rating prediction algorithm based on the users' historical behavior probabilities which can improve the accuracy of predictions and reduce the number of the required neighbors. Yue et al. [15] proposed a modified collaborative filtering algorithm by combining the advantages of user-based and item-based collaborative filtering methods. Furthermore, this algorithm makes full use of the positively and negatively correlated neighbors to further improve the prediction accuracy.

Matrix factorization is another kind of collaborative filtering technique which approximate the original data matrix by the product of factored matrices [16–18]. Because of the accuracy and scalability of matrix factorization, it is getting more and more attention from researchers. Funk [19] proposed the singular value decomposition (SVD) approach, it can be viewed as the first matrix factorization based recommender system. Koren [20] proposed an SVD++ approach, which consists of SVD and the neighborhood model. Significantly, the SVD++ model is better than the SVD model in prediction accuracy. Furthermore, Koren [21] proposed the timeSVD, which considered the affect of record time. In order to improve the accuracy and the interpretability of the matrix factorization model, some researchers proposed various convolutional matrix factorization methods by combining the matrix factorization and Convolutional Neural Networks (CNN) [22,23]. Additionally, many researchers considered various meanings of missing value in different applications and proposed different ways to weight the missing value [10–12]. Such recommender system models are helpful for the application of recommender system in different scenes.

Intuitively, such recommender systems are feasible to recommend a corresponding access policy for each data. Thus, in this paper, we propose a matrix

factorization based linear policy recommender scheme. Different from the traditional recommender system, our linear policy recommender scheme recommends a policy vector and a security threshold for each data. It can decrease the difficulty of policy definition and greatly improve the efficiency of deploying the ABAC mechanism.

3 Linear Access Policy Recommender system

3.1 ABAC system

An ABAC system can be described by a quadruple (S, O, P, E), where S denotes a subject, O denotes a object, P denotes the permission and E denotes the environment conditions [2]. Generally, a subject can be a user who want to access a data; a object represents a data which shares by the data owner; the permission represents a operation that the subject requests to perform on the object, including read, write, modify, and so on; the environment usually includes current time, location of a user, threat level, and so on. In this work, environment conditions is only considered as user attributes.

In general, the access control system includes the Policy Enforcement Point (PEP), the Policy Decision Point (PDP), the Policy Information Point (PIP), and the Policy Administration Point (PAP). The whole ABAC system includes two operational phases: the preparation phase and the execution phase [3]. As shown in Fig. 1, our ABAC system includes the Policy Decision Point (PDP), the Policy Administration Point (PAP), and the Policy Recommendation Point (PRP). In the preparation phase, PAP collects attribute sets and access logs, and requests access policies from PRP. Then, PRP extracts the fine-grained access privilege from access logs and attribute sets and generates linear policies, that is the proposed linear policy recommender scheme for PAP to manage access policies of data. The proposed scheme extracts the relation of attributes and privileges by matrix factorization and describes the access policy by the linear policy model. It is efficient and accurate for the relationship extraction. Finally, PAP assigns and manages access policies for each data based on the recommended linear policy. In the execution phase, PDP requests the corresponding access policy from PAP when it receives an access request from a subject. If the attribute set of the subject satisfies the access policy, PDP permits the access request (i.e. queries the object and responses to the subject). Otherwise, denies the access request. In most ABAC systems, PDP decides by matching the attribute value of the access request with each corresponding attribute value in the access policy. In our work, the linear model needs only one comparison which can reduce the workload of PDP.

3.2 Matrix Factorization Based Access Privilege Recommendation

The matrix factorization model aims to learn two matrices with a product that can approximate the original data matrix [16,17], where the two matrices represent cluster centroid attributes and cluster indicator information [18]. In our

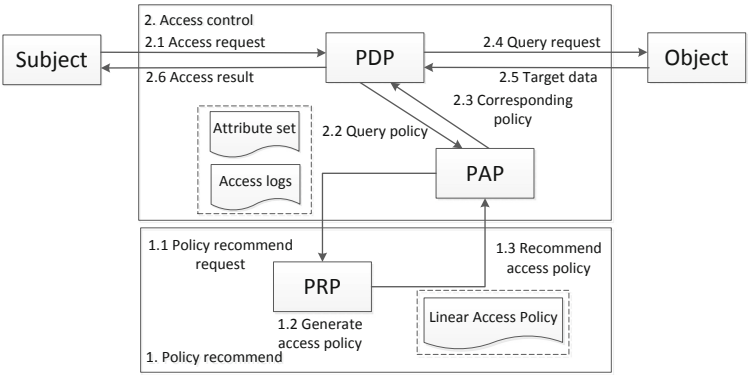


Fig. 1. The framework of ABAC system

matrix factorization based policy recommender system, it tries to learn a user attribute matrix and a policy matrix with a product that can approximate the access matrix.

Let $\mathbf{P} = (p_{u,j})_{m \times f}$ denotes the user matrix, where m denotes the number of users, f denotes the number of user attributes and each row represents a specified user. $\mathbf{Q} = (q_{j,i})_{f \times n}$ denotes the item matrix, where n denotes the number of items and each column represents a specified item. The predicted access matrix $\mathbf{R}$ is calculated as the inner product of $\mathbf{P}$ and $\mathbf{Q}$. It tries to approximate $\mathbf{R}$ as follows:

$$\min_{p,q} \sum_{(u,i)\in\kappa} \Big((r_{u,i} - \sum_{j=1}^{f} p_{u,j} q_{j,i})^2 + \lambda \sum_{j=1}^{f} (p_{u,j}^2 + q_{j,i}^2) \Big), \tag{1}$$

where κ is the set of the (u, i) pairs for which $r_{u,i}$ is known in rating matrix (the training set), $\lambda \in [0, 1]$ denotes the regularized coefficient. Furthermore, the Mini-Batch Gradient Descent approach is proposed to solve the solution of Eq. (1). In fact, the system predicts each $r_{u,i}$ in the training set and computes the prediction error $e_{u,i} = r_{u,i} - \sum_{j=1}^{f} p_{u,j} q_{j,i}$. Let $\gamma \in [0, 1]$ denotes the learning rate, b denotes the size of batch. Then, $p_{u,j}$ and $q_{j,i}$ are updated as follows:

$$p_{u,j} \leftarrow p_{u,j} + \sum_{i'=i}^{i+b} \gamma (e_{u,i'} q_{j,i'} - \lambda p_{u,j}), \tag{2}$$

$$q_{j,i} \leftarrow q_{j,i} + \sum_{u'=u}^{u+b} \gamma (e_{u',i} p_{u',j} - \lambda q_{j,i}), \tag{3}$$

3.3 Linear Policy Recommendation

Access policy is also known as access structure which influences the efficiency and the flexibility of the ABAC system. It is usually defined as follow:

Definition 1 (Access Policy [24]**).** *Let* $\{P_1, P_2, ..., P_n\}$ *be a set of parties. An access structure is a collection* $\mathbb{A}$ *of non-empty subsets of* $\{P_1, P_2, ..., P_n\}$*, i.e.* $\mathbb{A} \subseteq 2^{\{P_1, P_2, ..., P_n\}} \setminus \{\emptyset\}$*. Furthermore,* $\forall A \in \mathbb{A}$ *is called the authorized set, and* $\forall \bar{A} \notin \mathbb{A}$ *is called the unauthorized sets. A collection* $\mathbb{A}$ *is monotone iff* $\forall B \in \mathbb{A}, B \subset C \Leftrightarrow C \in \mathbb{A}$*.*

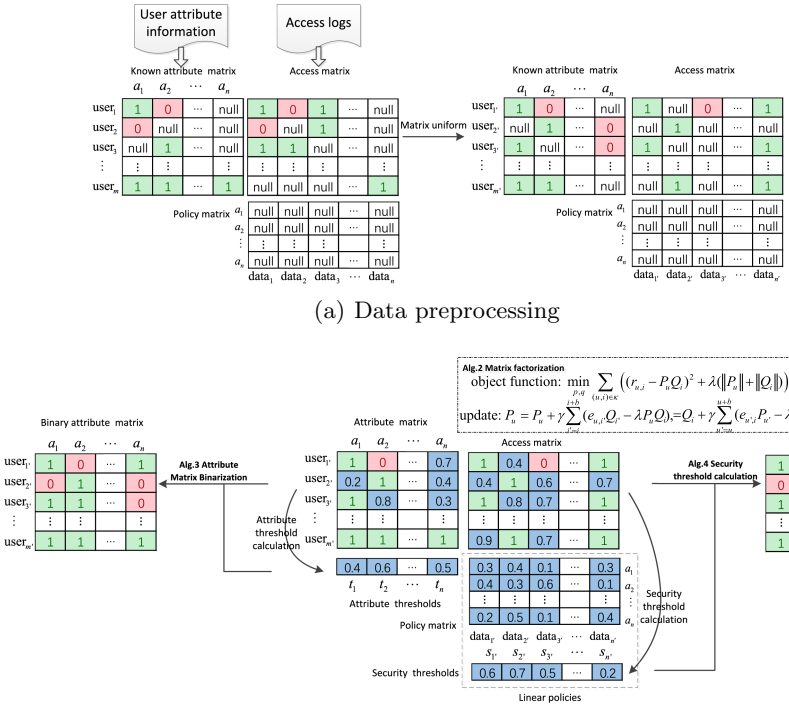
Attributes collection set $\mathbb{A}$ is a generalized form of access policy. It describes a access policy by a *minimum authorized set* or a *maximum unauthorized set.* Furthermore, the access policy is usually described by various specific forms, such as Access Tree [25], Linear Secret Sharing Scheme Matrix (LSSS) [26], AND-gates Access Structure [27], Threshold Access Structure [24], and so on.

In this paper, we proposed a form of access policy named linear policy. For each data, the access policy can be expressed as a policy vector and a security threshold. Let $\mathcal{A} = \{a_k, 1 \leq k \leq f\}$ be a set of attributes. The attribute set of a user u can be expressed as a binary vector $P_u = (p_{u,1}, \ldots, p_{u,f})$ where $\{p_{u,k} = 1, 1 \leq k \leq f\}$ iff the user get the k^{th} attribute of $\mathcal{A}$. Meanwhile, a linear policy of a data i can be given as (Q_i, s_i), where $Q_i \in [0,1]^f$ denotes a policy vector and $s_i \in [0,1]$ denotes a security threshold. Thus, the data i can be accessed by user u iff $P_u \times Q_i \geq s_i$.

In the proposed policy recommender scheme, the access matrix $\mathbf{R}$ is factorized as the product of policy matrix $\mathbf{Q}$ and attribute matrix $\mathbf{P}$, where $\mathbf{Q}$ can be used to describe the linear policy of each data. In fact, the linear policy model is more flexible than Threshold Access Structure and more efficient than complex access structures, such as Access Tree and LSSS. In most cases, a complex access structure can directly transform into a linear policy. A data owner can request linear policy for a data by the proposed recommender scheme as follow. Firstly, the data owner uploads the data access logs or labels part of authorized users and unauthorized users. Then, the proposed policy recommender scheme can recommend a linear policy (i.e. a policy vector and a security threshold) for the data by the access information and some user attribute information. Finally, the data owner can assign an access policy for the data base on the recommended policy. Furthermore, the data owner can adjust the security threshold according to different security requirements.

4 Linear Policy Recommender Scheme Based on Binary Matrix Factorization

Our policy recommender scheme takes the access logs $\mathbf{A}$ and the user attribute information $\mathbf{U}$ as inputs and outputs fine-grained access policy for each shared data, the whole scheme includes four functions: data preprocessing, access matrix factorization, attribute matrix binarization, and security threshold calculation. As shown in Fig. 2. Firstly, it transforms the access logs $\mathbf{A}$ and the user attribute information into a *uniform* user-data access matrix $\mathbf{R}_{m \times n}$ and a binary known user attribute matrix $\mathbf{P'}_{m \times f}$, which can be viewed as the data preprocessing. Secondly, it gets the user attribute matrix $\mathbf{P}_{m \times f}$ and the policy matrix $\mathbf{Q}_{f \times n}$



(a) Data preprocessing

(b) Linear policy recommending

Fig. 2. The overview of the linear policy recommender scheme

from $\mathbf{R}$ and $\mathbf{P'}_{m\times f}$ by a matrix factorization algorithm. Because a part of user attributes are unknown, our policy recommender scheme also learns the unknown user attribute by the matrix factorization algorithm. Obviously, the user attribute matrix consists of the known user attribute matrix and the learned user attribute matrix. Because the attribute matrix is a binary matrix, we introduce a method to binarize $\mathbf{P}$. Finally, a security threshold is chosen for each data. The policy matrix $\mathbf{Q}$ and security threshold vector $\mathbf{s}$ are used to describe access policies.

4.1 Data Preprocessing

The policy recommender scheme needs to extract the relationship of user attributes from access logs and user attribute information, and recommend access policies for each data. In this section, we transform access logs $\mathbf{A}$ into a binary matrix $\mathbf{R}$, where each row of $\mathbf{R}$ denotes a user, each column of $\mathbf{R}$ denotes a data. We consider the following cases:

1. The u^{th} user used to access the i^{th} data successfully, $r_{u,i}$ is set to be 1;
2. The u^{th} user ever failed to access the i^{th} data, $r_{u,i}$ is set to be 0;

3. The u^{th} user never accessed the i^{th} data before, $r_{u,i}$ is set to be *null*.

Similarly, the user attribute information $\mathbf{U}$ is transformed into a binary matrix $\mathbf{P}'$, where each row of $\mathbf{P}'$ denotes a user, and each column of $\mathbf{P}'$ denotes an attribute. We consider the following cases:

1. The u^{th} user gets the j^{th} attribute, $p'_{u,j}$ is set to be 1;
2. The u^{th} user dose not get the j^{th} attribute, $p'_{u,j}$ is set to be 0;
3. The user attribute information dose not explicitly describe weather the u^{th} user get the j^{th} attribute or not, $p'_{u,j}$ is set to be *null*.

In order to make the elements (i.e. '0' or '1') are *uniform* distributed in $\mathbf{R}$, Analysis of Variance (ANOVA) [28] is introduced to determine the matrix $\mathbf{R}$. Let y_i be the total of known element in the i^{th} row of $\mathbf{R}$ and $k = \lceil m/b \rceil$, we get

$$S_e = \sum_{j=0}^{k-1} \sum_{i=bj}^{(j+1)b-1} (y_i - \bar{y}_j)^2 + \sum_{i=kb}^{m-1} (y_i - \bar{y}_k)^2, f_e = m - (k+1), \tag{4}$$

$$S_A = b \sum_{j=0}^{k-1} (\bar{y}_j - \bar{y})^2 + b'(\bar{y}_k - \bar{y})^2, f_A = k, \tag{5}$$

where

$$\bar{y}_j = \begin{cases} \frac{1}{b} \sum\limits_{i=bj}^{(j+1)b-1} y_i, & j < k, \\ \frac{1}{b'} \sum\limits_{i=bk}^{m-1} y_i, & j = k, \end{cases} \tag{6}$$

$$\bar{y} = \sum_{i=0}^{m-1} y_i, \tag{7}$$

$$b' = m - kb. \tag{8}$$

In fact, S_e denotes the sum of *intraclass* variance and S_A denotes the sum of *interclass* variance, they can describe the distribution of known elements of $\mathbf{R}$. Then, we calculate F as follow:

$$F = \frac{S_A / f_A}{S_e / f_e}. \tag{9}$$

The matrix $\mathbf{R}$ can be accepted as a *row-uniform* matrix, iff $F_{1-\alpha}(f_A, f_e) < F < F_{1-\alpha}(f_e, f_A)$, where $\alpha \in [0, 1]$, f_e and f_A denote the degree of freedom, $F_{1-\alpha}(f_A, f_e)$ and $F_{1-\alpha}(f_e, f_A)$ denote the upper percentage points of F-distribution. Similarly, the notion of *column-uniform* matrix can be given in the same way here. Furthermore, a matrix can be accepted as a *uniform* matrix iff it is *row-uniform* and *column-uniform*.

The data preprocessing is shown in Algorithm 1, which takes the access logs $\mathbf{A}$, user attribute information $\mathbf{U}$, batch size b, and quantile α as inputs and

Algorithm 1. Matrix Initialization $MI(\mathbf{A}, \mathbf{U}, b, \alpha)$

Input: Access logs $\mathbf{A}$; User attribute information $\mathbf{U}$; Size of batch b; Quantile α
Output: *Uniform* User-data access matrix $\mathbf{R}$; Known user attribute matrix $\mathbf{P}'$

```
Transform the access logs A and the user attribute information U into an access
  matrix R and a user attribute matrix P'
Calculate the degree of freedom f_e and f_A of rows of R as shown in Eq. (4) (5)
Calculate the statistic parameter F of rows of R as shown in Eq. (9)
while F < F_{1-α}(f_A, f_e) or F > F_{1-α}(f_e, f_A) do
    R ← randomly permute the row of R
    Calculate the statistic parameter F of rows of R
end while
Calculate the degree of freedom f_e and f_A of columns of R
Calculate the statistic parameter F of columns of R
while F < F_{1-α}(f_A, f_e) or F > F_{1-α}(f_e, f_A) do
    R ← random permute the cloumn of R
    Calculate the statistic parameter F of columns of R.
end while
return R, P'
```

initializes the access matrix $\mathbf{R}$ and the user attribute matrix $\mathbf{P}'$. Then, the matrix $\mathbf{R}$ is randomly permuted with rows and columns until it gets *uniform*. Finally, the algorithm outputs a *uniform* access matrix $\mathbf{R}$ and a corresponding user attribute matrix $\mathbf{P}'$.

4.2 Access Matrix Factorization

In a recommender system, the *null* records always contain some information. For instance, in our policy recommender scheme, we consider two cases of *null* records:

1. The user never accesses the data because he/she knows that he/she does not get the access privilege;
2. The user never accesses the data because he/she does not need to access the data.

In brief, the *null* records actually denote 0 in the first case. Thus, we deal the *null* as 0 with a little weight $\omega \in (0, 1)$. The recommender scheme can be defined as follows:

$$L = \min_{p_{u,j}, q_{j,i}} \sum_{u,i} \omega_{u,i}((r'_{u,i} - \sum_{j} p_{u,j} q_{j,i})^2 + \lambda(\sum_{j} p_{u,j}^2 + \sum_{j} q_{j,i}^2)), \tag{10}$$

$$s.t. \begin{cases} p_{u,j} \in \{0, 1\}, q_{j,i} \in [0, 1], \\ r'_{u,i} = \begin{cases} 1 & \text{if } r_{u,i} = 1, \\ 0 & \text{if } r_{u,i} = 0 \text{ or } null, \end{cases} \\ \omega_{u,i} = \begin{cases} 1 & \text{if } r_{u,i} \text{ is not } null, \\ \omega & \text{if } r_{u,i} \text{ is } null. \end{cases} \end{cases} \tag{11}$$

where L denotes the objective function and λ denotes the weight of regular terms. Considering of efficiency and accuracy, Mini-Batch Gradient Descent (MBGD) is used to iteratively search for the optimal solution of the above problem. Let γ be the learning rate, in the l^{th} round of the iteration, $p_{u,j}^{(l)}$ and $q_{j,i}^{(l)}$ is updated as follows:

$$p_{u,j}^{(l)} = p_{u,j}^{(l-1)} + \gamma \frac{1}{b} \sum_{i=kb}^{(k+1)b-1} \omega_{u,i'}((r'_{u,i'} - \sum_{j=1}^{f} p_{u,j}^{(l-1)} q_{j,i'}^{(l-1)}) q_{j,i'}^{(l-1)} - \lambda p_{u,j}^{(l-1)}), \quad (12)$$

$$q_{j,i}^{(l)} = q_{j,i}^{(l-1)} + \gamma \frac{1}{b} \sum_{u=kb}^{(k+1)b-1} \omega_{u',i}((r'_{u',i} - \sum_{j=1}^{f} p_{u',j}^{(l-1)} q_{j,i}^{(l-1)}) p_{u',j}^{(l-1)} - \lambda q_{j,i}^{(l-1)}), \quad (13)$$

where $i' = i \mod m$ and $u' = u \mod n$. The matrix factorization algorithm of our recommender scheme is shown in Algorithm 2. In particular, it firstly initializes an attribute matrix $\mathbf{P} = (p_{u,j})_{m \times f} \in \{0,1\}^{m \times f}$ with known user attribute matrix $\mathbf{P'}$ and initializes a policy matrix $\mathbf{Q} = (q_{j,i})_{f \times n} \in [0,1]^{f \times n}$ randomly. Then, $\mathbf{P}$ and $\mathbf{Q}$ are iteratively updated by Eq. (12) and Eq. (13) respectively to minimize the objective function L. The iterative algorithm would be terminated iff $|L^{(l)} - L^{(l-1)}| \leq \theta$ or $l \geq T$, where $L^{(l)}$ and $L^{(l-1)}$ denote the value of L in the l^{th} and $(l-1)^{th}$ iteration respectively, θ denotes the convergence threshold and T denotes the maximum iterations. Furthermore, in this work, $\mathbf{P}$ and $\bar{\mathbf{R}}$ are binary matrices, the binarization mechanism should be introduced into the matrix factorization algorithm. For efficiency, our scheme binarizes $\mathbf{P}$ each k iterations and binarizes $\bar{\mathbf{R}}$ after the end of Algorithm 2.

4.3 Attribute Matrix Binarization

In this work, user attributes are given as a binary vector, in which each row of $\mathbf{P}$ represents a user attribute set. However, there are some unknown user attributes learned by Algorithm 2 which are non-binary. Thus, a binarization algorithm of $\mathbf{P}$ is proposed to deal such learned attributes. As shown in Algorithm 3 which outputs an attribute threshold t_j for each column P_j to binarize it. The algorithm includes the following four steps:

1. Pick k samples $\{a_{j,i'}, 0 \leq i' \leq k-1\}$ of each column P_j in descending order;
2. Calculate candidate threshold set $\mathbf{T}_j = \{t_{j,i'}, 1 \leq i' \leq k-1\}$ as follows:

$$t_{j,i'} = a_{j,i'} - \frac{i'(a_{j,i'} - a_{j,i'+1})}{k}; \quad (14)$$

3. Choose the optimal threshold t_j from $\mathbf{T}_j$ which minimizing the objective function L;
4. Binarize attribute matrix P_j by threshold t_j.

Algorithm 2. Matrix Factorization $MF(\mathbf{R}, \mathbf{P}')$

Input: User-data access matrix $\mathbf{R}$; Known user attribute $\mathbf{P}'$
Output: Attribute matrix $\mathbf{P}$; Policy matrix $\mathbf{Q}$

Initialize $\mathbf{P} \in [0,1]^{m \times f}$ with $\mathbf{P}'$ and initialize $\mathbf{Q} \in [0,1]^{f \times n}$ randomly;
Calculate the objective function $L^{(0)}$ as shown in Eq. (10).
$l = 0$, $L^{(-1)} = 0$
while $|L^{(l)} - L^{(l-1)}| > \theta$ or $l < T$ **do**
  //θ denotes the convergence threshold and T denotes the maximum iterations
  for t from 1 to k **do**
    for u from 1 to m **do**
      for j form 1 to f **do**
        if $p'_{u,j} = null$ **then**
          $p_{u,j} \leftarrow p_{u,j} + \gamma \frac{1}{b} \frac{\partial L^{(l-1)}}{\partial p_{u,j}}$
        end if
      end for
    end for
    for i from 1 to n **do**
      for j form 1 to f **do**
        $q_{j,i} \leftarrow q_{j,i} + \frac{1}{b} \frac{\partial L^{(l-1)}}{\partial q_{j,i}}$
      end for
    end for
  end for
  $\mathbf{P} \leftarrow AMB(\mathbf{P})$ //i.e. call the Algorithm 3
  $l \leftarrow l + 1$
end while
return $\mathbf{P}$, $\mathbf{Q}$

Algorithm 3. Attribute Matrix Binarization $AMB(\mathbf{P})$

Input: Attribute matrix $\mathbf{P}$
Output: Binary attribute matrix $\mathbf{P}$

for j form 1 to f **do**
  Choose k samples $\{a_{j,i'}, 0 \le i' \le k-1\}$ from P_j in descending order
  $L_j \leftarrow null$
  for i' from 1 to $k-1$ **do**
    Calculate $t_{j,i'}$ as shown in Eq. (14)
    Calculate $\bar{P}_j$ as binary $P_j = \{\bar{p}_{j,i}\}$: $\bar{p}_{j,i} = \begin{cases} 0, & p_{j,i} < t_{j,i'} \\ 1, & p_{j,i} \ge t_{j,i'} \end{cases}$
    Compute the objective function L with $\bar{P}_j$
    if $L_j = null$ or $L_j \ge L$ **then**
      $t_j \leftarrow t_{j,i'}, L_j \leftarrow L, P'_j \leftarrow \bar{P}_j$
    end if
  end for
  $P_j \leftarrow P'_j$
end for
return $\mathbf{P}, t_j, 1 \le j \le f$

Algorithm 4. Security Threshold Calculation $STC(\mathbf{R}, \mathbf{P}, \mathbf{Q})$

Input: User-data access matrix $\mathbf{R}$; Attribute matrix $\mathbf{P}$; Policy matrix $\mathbf{Q}$
Output: Policy matrix $\mathbf{Q}$; security threshold vector $\mathbf{s}$

$\hat{\mathbf{R}} \leftarrow \mathbf{PQ}$
for i form 1 to n **do**
  Get the descending sort $\mathbf{i}'$ of $\hat{R}_i$, where $\hat{R}_i$ denotes the i^{th} column of $\hat{\mathbf{R}}$
  $num = 0$
  for j from 1 to $m-1$ **do**
    if $r_{i'_j,i} \neq r_{i'_{j+1},i}$ **then**
      $//r_{i'_j,i}, r_{i'_{j+1},i}$ denotes the components of original matrix $\mathbf{R}$
      $\bar{s}_{i,num} \leftarrow \hat{r}_{i'_j,i}, num \leftarrow num + 1$
    end if
  end for
  $\bar{L}_i \leftarrow null$
  for k from 1 to $num - 1$ **do**
    $s_{i,k} \leftarrow \bar{s}_{i,k} - \frac{k(\bar{s}_{i,k} - \bar{s}_{i,k+1})}{num}$
    Calculate $\bar{R}_i$ as shown in Eq. (16)
    if $\bar{L}_i = null$ or $L_i(\bar{R}_i) < \bar{L}_i$ **then**
      $s_i \leftarrow s_{i,k}, \bar{L}_i \leftarrow L_i(\bar{R}_i)$
    end if
    $k \leftarrow k + 1$
  end for
end for
return $s_i, 1 \leq i \leq n$

4.4 Security Threshold Calculation

The predicted matrix $\bar{\mathbf{R}}$ of the original matrix $\mathbf{R}$ is also a binary matrix in this work. Thus, the security threshold vector $\mathbf{s}$ is required to binarize the matrix $\hat{\mathbf{R}} = \mathbf{P} \times \mathbf{Q}$, where $\mathbf{P}$ and $\mathbf{Q}$ denote the learned attribute matrix and policy matrix, respectively. As shown in Algorithm 4, the security threshold calculation algorithm is similar to Algorithm 3, which includes the following three steps:

1. Compute the matrix $\hat{\mathbf{R}}$ by attribute matrix $\mathbf{P} \in \{0,1\}^{m \times f}$ and policy matrix $\mathbf{Q} \in [0,1]^{f \times n}$.
2. Calculate candidate thresholds. For each column $\hat{R}_i$ of $\hat{\mathbf{R}}$, a candidate threshold set $S_i = \{s_{i,k}\}$ is chosen. Note that, $s_{i,k}$ is chosen as local optimum, the processing is similar to $t_{j,i'}$ choosing.
3. Choose the best threshold. Let the objective function be simplified as follows:

$$L_i(s_{i,k}) = \sum_{u=1}^{m} w_{u,i}(r_{u,i} - \bar{r}_{u,i,s_{i,k}})^2, \tag{15}$$

$$\bar{r}_{u,i,s_{i,k}} = \begin{cases} 0, & \hat{r}_{u,i} \leq s_{i,k} \\ 1, & otherwise \end{cases}, \tag{16}$$

where $\hat{r}_{u,i} = \sum_j p_{u,j} q_{j,i}$ and $s_{i,k}$ is k^{th} candidate threshold of R_i. The final security threshold $s_i \in S_i$ is the solution which makes L_i minimum.

5 Performance Evaluation

5.1 Data Sets and Metrics

In this paper, we use Amazon Access Samples Data Set (AAS) to evaluate the performance of our scheme[1] We extract 3 datasets from AAS by years: AAS56, AAS08, and AAS10. AAS56 contains more than 60 thousand records of the access right for 4249 users to 856 data. AAS08 contains more than 130 thousand records of the access right for 7623 users to 3017 data. AAS10 contains more than 230 thousand records of the access right for 13530 users to 6059 data. There are 12 attributes for each user, we choose 10 attributes in our experiments. Table 1 shows the detail of datasets. We randomly split each dataset into a training set (80%) and a testing set (20%), and we report the average results of five experiments.

Our scheme can be viewed as a binary classification model. Normally, *precision* and *recall* are two important evaluation metrics to measure the prediction accuracy of classification, and *F1-score* is a composite metric of *precision* and *recall*. The definition is shown as follows:

$$precision = TP/(TP + FP), \tag{17}$$

$$recall = TP/(TP + FN), \tag{18}$$

$$F1\text{-}score = \frac{2 \times precision \times recall}{precision + recall}, \tag{19}$$

where TP, FP, TN, and FN denote the number of true positive, false positive, true negative, and false negative respectively.

5.2 Experimental Evaluation

In this section, all presented experiments were conducted on a cloud server with two intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz 2.60 GHz and 32 GB RAM. Let γ denotes the learning rate, λ denotes the regularization parameter, b denotes the size of batch, k denotes the frequency of binarization, $\omega_{u,i}$ denotes the weight of unknown elements, and T denotes the maximum iterations.

Firstly, we analyze the influence of the batch size b and the binarization frequency k in AAS56, where $\gamma = 0.001$, $\lambda = 0.01$, $\omega_{u,i} = 0.0001$, and $T = 300$. Fig. 3 shows that, the *F1-score* increases with the growth of the binarization frequency, but the effect of the batch size is not signification. When the binarization frequency is greater than 30, *F1-score* remains stable in the range of 0.93 to 0.94.

Then, we analyze the influence of the learning rate γ and the weight of unknown element $w_{u,i}$ in AAS56, AAS08, and AAS10. Figure 4 shows *F1-score* decreases with the growth of learning rate, where $\lambda = 0.01$, $\omega_{u,i} = 0.0001$, $b = 150$, $k = 50$, and $T = 300$. Because the greater learning rate can't make

[1] http://archive.ics.uci.edu/ml/datasets/Amazon+Access+Samples.

Table 1. The description of datasets

Dataset	*Accessable*	*Unaccessable*	Users	Items	Density
AAS56	13078	1147	4249	856	0.017648
AAS08	42759	1466	7623	3017	0.005704
AAS10	96272	2660	13530	6059	0.002865

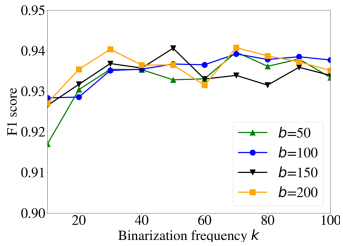


Fig. 3. The influence of the binarization frequency and the batch size for *F1-score*

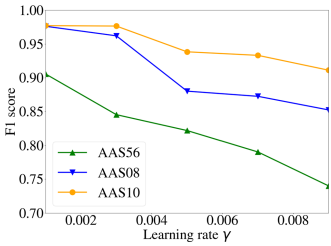


Fig. 4. Effect of learning rate in different datasets

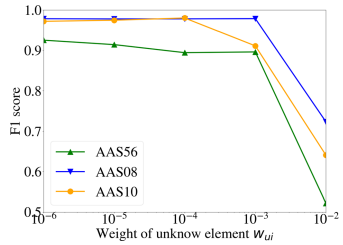


Fig. 5. Effect of the weight of unknown element in different datasets

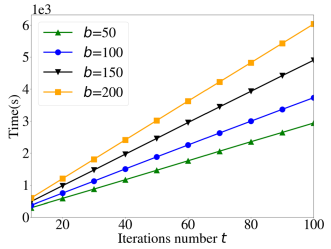


Fig. 6. The influence of the iterations number and the batch size for running time

the prediction achieving the optimum. As Fig. 5 shows, the *F1-score* increases with the decrease of the weight of unknown element, where $\gamma = 0.001$, $\lambda = 0.01$, $b = 150$, $k = 50$, and $T = 300$. In AAS56 and AAS08, *F1-score* remains stable, when the weight of unknown element is less than 0.001. However, in AAS10, *F1-score* remains stable, when the weight of unknown element is less than 0.0001. Because the rate of unaccessable in AAS10 is less than AAS56 and AAS08. In addition, the best *F1-score* in AAS56 is less than AAS08 and AAS10. Because the size of AAS56 is smaller than AAS08 and AAS10.

Finally, We analyze some parameters that affect the running time of the access policy recommender scheme. Fig. 6 shows the influence of the number of iterations and the batch size in AAS56, where $\gamma = 0.0001$, $\lambda = 0.01$, $\omega_{u,i} = 0.0001$, $b = 150$, and $k = 50$. Obviously, the running time increases with the growth of the number of iterations and the batch size.

6 Conclusion

In an ABAC mechanism, a policy recommender scheme can provide efficient policy management for each data. In this paper, we firstly propose a novel expressing form of access policy named linear policy. It can not only directly describe the fine-grained access privilege by a linear function, but also flexibly adjust the data security level by updating the security threshold of the data. Secondly, we propose a matrix factorization based linear policy recommender scheme for ABAC, which can recommend a linear policy for each data accurately and quickly. The experiment results show that the accuracy of the recommended policy is always over 0.9. Particularly, in the best case, the accuracy achieves 0.98. In brief, our access policy recommender scheme can greatly enhance the efficiency of policy assignation, which is helpful for the promotion and application of the ABAC mechanism.

Acknowledgment. This work was supported by the National Science Foundation of China (No. 61802083, 61862011), the Natural Science Foundation of Guangxi (2018GXNSFB A281164, 2018GXNSFAA138116) and Innovation Project of GUET Graduate Education (2021YCXS074).

References

1. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2), 38–47 (1996)
2. Pan, R., Wang, G., Wu, M.: An attribute-based access control policy retrieval method based on binary sequence. Secur. Commun. Netw. **2021**, 5582921:1-5582921:12 (2021)
3. Fang, L., Yin, L., Guo, Y., Fang, B.: A survey of key technologies in attribute-based access control scheme. Chin. J. Comput. **40**(7), 1681–1698 (2017)
4. Xu, Z., Stoller, S.D.: Mining attribute-based access control policies. IEEE Trans. Dependable Secur. Comput. **12**(5), 533–545 (2015)
5. Iyer, P., Masoumzadeh, A.: Mining positive and negative attribute-based access control policy rules. In: Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, pp. 161–172 (2018)
6. Karimi, L., Joshi, J.: An unsupervised learning based approach for mining attribute based access control policies. In: IEEE International Conference on Big Data, pp. 1427–1436 (2018)
7. Bui, T., Stoller, S.D.: Learning attribute-based and relationship-based access control policies with unknown values. In: Kanhere, S., Patil, V.T., Sural, S., Gaur, M.S. (eds.) ICISS 2020. LNCS, vol. 12553, pp. 23–44. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-65610-2_2
8. Benkaouz, Y., Erradi, M., Freisleben, B.: Work in progress: K-nearest neighbors techniques for ABAC policies clustering. In: Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control, pp. 72–75 (2016)
9. Sanders, M.W., Yue, C.: Mining least privilege attribute based access control policies. In: Proceedings of the 35th Annual Computer Security Applications Conference, pp. 404–416 (2019)

10. Liang, D., Charlin, L., McInerney, J., Blei, D.M.: Modeling user exposure in recommendation. In: Proceedings of the 25th International Conference on World Wide Web, pp. 951–961 (2016)
11. Ding, J., et al.: Improving implicit recommender systems with view data. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 3343–3349 (2018)
12. He, X., Tang, J., Du, X., Hong, R., Ren, T., Chua, T.: Fast matrix factorization with nonuniform weights on missing data. IEEE Trans. Neural Netw. Learn. Syst. **31**, 2791–2804 (2020)
13. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 77–118. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_3
14. Su, Z., Lin, Z., Ai, J., Li, H.: Rating prediction in recommender systems based on user behavior probability and complex network modeling. IEEE Access **9**, 30739–30749 (2021)
15. Yue, W., Wang, Z., Liu, W., Tian, B., Lauria, S., Liu, X.: An optimally weighted user- and item-based collaborative filtering approach to predicting baseline data for Friedreich's ataxia patients. Neurocomputing **419**, 287–294 (2021)
16. Ma, J., Zhang, Y., Zhang, L.: Discriminative subspace matrix factorization for multiview data clustering. Pattern Recognit. **111**, 107676 (2021)
17. Meng, Y., Shang, R., Shang, F., Jiao, L., Yang, S., Stolkin, R.: Semi-supervised graph regularized deep NMF with bi-orthogonal constraints for data representation. IEEE Trans. Neural Networks Learn. Syst. **31**(9), 3245–3258 (2020)
18. Ding, C.H.Q., Li, T., Jordan, M.I.: Convex and semi-nonnegative matrix factorizations. IEEE Trans. Pattern Anal. Mach. Intell. **32**(1), 45–55 (2010)
19. Funk, S.: Netflix update: Try this at home (2006). http://sifter.org/simon/journal/20061211.html
20. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434 (2008)
21. Koren, Y.: Collaborative filtering with temporal dynamics. Commun. ACM **53**, 89–97 (2010)
22. Zheng, L., Noroozi, V., Yu, P.S.: Joint deep modeling of users and items using reviews for recommendation. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 425–434 (2017)
23. Chen, C., Zhang, M., Liu, Y., Ma, S.: Neural attentional rating regression with review-level explanations. In: Proceedings of the 2018 World Wide Web Conference, pp. 1583–1592 (2018)
24. Liu, Z., Cao, Z., Wong, D.S.: Efficient generation of linear secret sharing scheme matrices from threshold access trees. IACR Cryptol. ePrint Arch., 2010:374 (2010)
25. Li, J., Yao, W., Han, J., Zhang, Y., Shen, J.: User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage. IEEE Syst. J. **12**, 1767–1777 (2018)
26. Li, J., Wang, Y., Zhang, Y., Han, J.: Full verifiability for outsourced decryption in attribute based encryption. IEEE Trans. Serv. Comput. **13**, 478–487 (2020)
27. Phuong, T.V.X., Yang, G., Susilo, W.: Hidden ciphertext policy attribute-based encryption under standard assumptions. IEEE Trans. Inf. Forensics Secur. **11**, 35–45 (2016)
28. Zhang, J., Cheng, M., Wu, H., Zhou, B.: A new test for functional one-way ANOVA with applications to ischemic heart screening. Comput. Stat. Data Anal. **132**, 3–17 (2019)