A Trust Management Framework for Secure Cloud Data Storage Using Cryptographic Role-Based Access Control

Lan Zhou, Vijay Varadharajan[✉], and Michael Hitchens

Advanced Cyber Security Research Centre, Department of Computing, Macquarie University, North Ryde, Sydney, NSW, Australia
{lan.zhou,vijay.varadharajan,michael.hitchens}@mq.edu.au

Abstract. In recent times, there has been an increasing development of storing data securely in the cloud. The Role-based access control (RBAC) model, a widely used access control model, can provide a flexible way for data owners to manage and share their data in the cloud environment. To enforce the access control policies in the cloud, several cryptographic RBAC schemes have been proposed recently, which integrate cryptographic techniques with RBAC models to secure data storage in an outsourced environment such as a cloud. However, these schemes do not address the issue of trust in such a data storage system. In this paper, we introduce a trust management framework which can enhance the security of data in cloud storage systems using cryptographic RBAC schemes. The trust management framework provides an approach for each party in such a cloud storage system to determine the trustworthiness of other parties. The framework consists of a series of trust models, which (i) enable the users and the data owners to decide whether to interact with a particular role for accessing and sharing data in the system and (ii) allow the role managers to evaluate the trustworthiness of users and data owners. These trust models take into account role inheritance and hierarchy in the evaluation of trustworthiness of the roles. In addition, we present a design of a trust-based cloud storage system which shows how the trust models for users and roles can be integrated into a system that uses cryptographic RBAC schemes.

Keywords: Role-based access control · Trust model · Cryptographic RBAC

1 Introduction

Controlling the access to data is an important issue in data storage systems. A proper access control mechanism is needed depending on the context and the requirement of the system. Many access control models have been proposed over the years in the literature. Role-based access control (RBAC) is a well-known access control model which can help to simplify security management especially in large-scale systems. In RBAC, roles are used to associate users with permissions on resources. Users are assigned roles and permissions are allocated to roles instead of individual users; only users who have been granted membership to roles can access the permissions associated with the roles and hence can access the resources. Since being first formalised in the 1990's [5], RBAC has been widely used in many systems to provide users with flexible controls over the access to their data. The RBAC model was extended and updated in 1996 [11], and a NIST RBAC standard was proposed in 2000 [12].

In traditional systems, access control policies are usually specified and enforced by a central authority who has administrative control over all the resources in the system. With the rapid increase in the amount of digital information that needs to be stored, cloud storage has attracted much attention in recent years because of its ability to deliver storage resources to users on demand in a cost effective manner. In such an environment, there may not exist a central authority as the data may be stored in distributed data centres which cannot be under the control of a single authority. One approach to control the access to data in an untrusted environment is to encrypt the data and give the key to users who require access to the data.

Several cryptographic schemes have been developed to allow data encryption in the context of the RBAC model. A hierarchical cryptographic access control scheme [1] was proposed in 1983. Because of the similarity in structures between hierarchical access control and RBAC, a hierarchical cryptographic access control scheme can be easily transformed into a cryptographic RBAC scheme. The problem of access control for securely outsourcing data using cryptographic techniques was first considered in [7]. Some other schemes were proposed afterwards, such as in [3,10,16]. Recently, a new role-based encryption (RBE) scheme has been proposed in [15]. In this scheme, the user memberships are managed by individual roles as opposed to a central administrator as in other cryptographic RBAC schemes. These schemes combine cryptographic techniques and access control to protect the privacy of the data in an outsourced environment where data can be encrypted in such a way that only the users who are allowed by the access policies can decrypt and view the data.

In some cases though the access control policies may be specified by the cloud provider authority itself in a centralised way, there could be multiple authorities to enforce these access policies distributed throughout the cloud system. Therefore there would be a need to trust these authorities to correctly specify the access control policies and enforce them properly. In some cryptographic RBAC schemes, roles and their users are managed by administrators who hold the master secrets of the systems. All the administration tasks in these schemes are centralised. Therefore, if one wants to know if a RBAC system is secure, it is primarily dependent on the trustworthiness of the administrator of the system.

However, in large-scale RBAC systems, it is impractical to centralise the task of managing these users and permissions, and their relationships with the roles in a small team of security administrators. Reference [15] proposes a new cryptographic RBAC scheme called Role-based Encryption (RBE) in which the user management can be decentralised to individual roles; that is, the administrators only manage the roles and the relationship among them while the role managers have the flexibility in specifying the user memberships themselves. In this paper, we consider trust models for cloud storage systems that are using cryptographic RBAC schemes like RBE, where each individual role manager can manage their user memberships without the need of involving the administrators. We believe this case is more general and can be used in large-scale RBAC systems. In such systems, the trust on the individual roles needs to be considered instead of the trust on the administrators.

There have been several trust models [2,14] for RBAC proposed in the literature. These trust models considered the trust on users to assist the decision making about whether or not to grant permissions to the users. In a cloud storage system using cryptographic RBAC schemes, it would be helpful if a user could determine whether or not a role in the system is trusted before joining it. This would be useful especially in systems where there is a cost for users to join a role, for example, users need to pay the subscription fee for joining roles. When a user evaluates the trust value of a role, she or he will only proceed with joining the role if the trust value of the role is above a certain trust threshold (this threshold being set by the users, and being different for different applications and context dependent). In a system where owners are allowed to choose the roles to which to assign their data, from the users' perspective, malicious owners can also cause negative behaviours of roles by assigning bad resources (e.g. virus, malware) to roles. Therefore, roles will also need to consider the trust of the data owners so that only data from well-behaved owners will be accepted.

In this paper, we introduce a trust management framework for securing data storage in cloud storage systems that are using cryptographic RBAC schemes. Though much work exists on trust models in RBAC, none of this work considers the trust on the RBAC system itself. The proposed trust management framework addresses this missing aspect of trust in cryptographic RBAC schemes to improve the decision making for entities in the cloud system. This framework consists of trust models which can assist (i) the users to evaluate the trust in the roles in a RBAC system and use this trust evaluation to decide whether to join a particular role or not, (ii) the role managers to evaluate the trust in the owners in the RBAC system and use this trust in the decision to accept data from an owner, (iii) the data owners to evaluate the trust in the roles in a RBAC system and use this trust evaluation to decide whether to store their encrypted data in the cloud for a particular role, and (iv) the role managers to evaluate the trust in the users in the RBAC system and use this trust in the decision to grant the membership to a user. Theses trust models take into account the effect of role inheritance in RBAC systems on trust evaluation. If a role A inherits all the permissions that a role B has, then we say role A is a ancestor role of role B, and role B is a descendent role of role A.

We give the formal definition of the first two trust models, users' trust in role managers and role managers' trust in data owners. These two trust models form a natural pair as they consider trust from a user's perspective.

We refer to these trust models as User-Role RBAC and Role-Owner RBAC trust models respectively. These two trust models can not only prevent users from joining roles which have bad historical behaviour in terms of sharing poor quality resources or misleading users on the content of resources, but also assist the role managers to identify the malicious owners who have caused a negative impact on the roles' trustworthiness. We also present the architecture of a trust-based cloud storage system which integrates the these two trust models in a cryptographic RBAC system. Then we describe the other two trust models, data owners' trust in role managers and role managers' trust in users. Furthermore, we describe the relevance of all the trust models in the framework by considering practical application scenarios and then illustrate how the trust evaluations can be used to enhance the quality of secure decision making by different entities of cloud storage service.

The paper is organised as follows. Section 2 reviews relevant preliminary knowledge that is needed for the design of our trust models. Section 3 describes the trust issues in a cryptographic RBAC system and discusses the trust requirements for each type of entities. We give the formal User-Role and Role-Owner RBAC trust models in Sect. 4. The architecture of our secure cloud storage system is presented in Sect. 5. We describe the owners' trust models in Sect. 6. In Sect. 7, we illustrate how the trust models in the framework can be used in a cloud service application to enhance the quality of security decision making. Section 8 discusses relevant related works and compares them with our proposed trust management framework. Section 9 concludes the paper.

2 Preliminaries

2.1 Experience-Based Trust

Trust has played a foundational role in security for a long period of time. It is clear that two entities may not trust each other on the basis of identity alone. There are a range of other attributes and credentials such as different types of privileges, the state of the platform being used as well as reputations, recommendations and histories that come into play in decision making. An experience-based trust model is a trust management system which enables the trust decisions to be made based on the historical behaviour of an entity. Such a system allows an entity to rate the transactions with other entities, and the trustworthiness of an entity is determined using the collection of ratings of the transactions that other entities have had with this entity. In most experience-based trust systems, one entity derives the trustworthiness of another entity from both experience of the former with the latter and the feedback on transactions provided by other entities which have had interactions with target entity in the past. An entity is able to evaluate its trust in another entity and the former can make a decision as to whether to not to continue its transaction with the latter, based on whether the trust value exceeds a certain threshold; this threshold is dependent on the context of the application at hand.

2.2 Bayesian Trust Model

Many approaches have been proposed that use probabilistic models to evaluate trust based on evidence which contains the number of "positive" and "negative" transactions in which a given entity have been involved. Perhaps the most common probabilistic model is the one based on Bayesian trust [6,8,9] using the beta probability distribution function. The beta family of distributions is a collection of continuous probability density functions defined over the interval [0, 1]. Suppose a beta distribution used for a parameter θ is defined as

$$P(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

where α and β are two parameters controlling the distribution of the parameter θ, and $0 \le \theta \le 1,\ \alpha > 0,\ \beta > 0$. Assume $X = \{x_1, \ldots, x_n\}$ is the collection of the feedbacks from the past n transactions, and X has r "positive" feedbacks and s "negative" feedbacks. Then the likelihood function can be defined as

$$P(X|\theta) = \prod_{i=1}^{n} P(x_i|\theta) = \theta^r(1-\theta)^s$$

The posterior distribution $P(\theta|X)$ is proportional to the multiplication of the prior $P(\theta)$ and the likelihood function $P(X|\theta)$, and we then have

$$\begin{aligned} P(\theta|X) &= \frac{P(X|\theta)P(\theta)}{P(X)} \\ &= \frac{\Gamma(r+\alpha+s+\beta)}{\Gamma(r+\alpha)\Gamma(s+\beta)}\theta^{r+\alpha-1}(1-\theta)^{s+\beta-1} \end{aligned}$$

Now let x_{i+1} be the possible feedback of the next transaction. The probability that x_{i+1} is a "positive" feedback given the transaction history X can be represented as

$$\begin{aligned} P(x_{i+1}|X) &= \int_0^1 d\theta\ P(x_{i+1}|\theta)P(\theta|X) \\ &= \int_0^1 d\theta\ \theta P(\theta|X) \\ &= E(\theta|X) \end{aligned}$$

Then we write the probability that the next transaction will be a "good" one as follows:

$$\mathcal{E}(r,s) = P(x_{i+1}|X) = \frac{r+\alpha}{r+\alpha+s+\beta} \qquad (1)$$

Using Eq. 1, one entity can derive the probability that the next transaction with another entity will be positive from the transaction history of the other entity. Most Bayesian trust systems assume that the parameters $\alpha = \beta = 1$, such as in [6]. Some other approaches allow the parameters α and β to be chosen depending on the system context.

3 Trust Issues in Using Cryptographic RBAC Schemes in Secure Cloud Storage

Cryptographic RBAC schemes integrate cryptographic techniques with RBAC models to secure the data storage. They inherit the features and concepts from RBAC models, and also have additional components that are specific to data storage systems. In the standard RBAC model, permissions are assigned to roles by the administrator of the system. However, in a system using cryptographic RBAC schemes, "permissions" are the data encrypted to roles, and the security policies are specified to control the users' access to data. Because data are usually not owned by a single party, cryptographic RBAC systems assume that data can be encrypted to a role by whoever owns the data as opposed to the administrator in the standard RBAC system. In this paper, we address trust issues for cryptographic RBAC systems. Therefore we adopt the above described concepts for cryptographic RBAC systems in our trust models.

Using cryptographic RBAC schemes in cloud storage systems, a data owner can encrypt the data to a role, and only the users who have been granted membership of that role or an ancestor role of that role can decrypt the data. In this paper, we assume that the data owners and users reside outside this role system infrastructure (where the roles are being administered). Hence the entities in a cloud storage system need to consider the following issues. Users consider their trust in roles (role managers) in order to ensure that joining roles guarantee access to data assigned to these roles, and data owners consider the trust of role managers in order to ensure that their data is secure after being assigned to the roles. The role managers need to consider trust on both data owners and users; role managers consider their trust in data owners to ensure that data owners who have assigned malicious data to the roles will not be allowed to assign data to the roles any more, and they consider the trust of users so that users with negative behaviours are excluded from the roles, which in turn makes owners trust these roles. In this section, we discuss the trust issues that need to be considered by different entities of a cryptographic RBAC system.

3.1 Data Users' Trust in Role Managers

In some RBAC systems, user-role assignment is managed by administrators of the systems where the administrators check the qualification of users and grant role membership to them. In these systems, users trust all the roles at the same level as they are all managed by the same administrators. The roles are trusted as long as the administrators are trusted.

In RBAC systems that use cryptographic RBAC schemes, users-role assignment can be decentralised to individual role managers to allow more flexibility in user management, especially in large-scale systems. Assume for example in these systems users join a role based on subscription for accessing the data assigned to that role. It is clear that users need to choose a trusted role when subscribing.

If the data that a user wants to access is encrypted to one role only, the user considers the trustworthiness of that role in deciding whether or not to join that role. When the same data is encrypted to multiple roles, users will need to evaluate the trustworthiness of these roles to choose the most reliable role to join. From the user's perspective, a trusted role should meet the following requirements:

- *Requirement 1: The role manager should grant membership to the users who are qualified for that role.*
  In order to access data, a user needs to join a role to which the data is encrypted. When the user requests to join the role, the role manager should give access (grant the membership) to the user if the user qualifies for that role, e.g. the user has paid the subscription fee. Refusing to give access will be considered as bad behaviour of the role manager.
- *Requirement 2: The data that a role claims to have should have been encrypted properly to that role.*
  When users want to access data, they need to know what data has been encrypted to which role so that they can choose a particular role to join. The list of the data is provided by roles. However, a user may find that she or he cannot locate or decrypt the data even after she or he has joined that specific role. This may happen if the data was not encrypted properly to that role by the owner, or the role claims to possess data that has not been encrypted to the role. Each role should take the responsibility of providing a valid and up-to-date list of the data that is in its possession.
- *Requirement 3: The data that the descendant roles of the role claim to have should have been encrypted properly to the descendant roles.*
  Since a role can inherit permissions from its descendant roles, a user who has joined a role should be able to access the data that is encrypted to any of its descendant roles. Each role is liable for the validity of the data that its descendant roles claim to have, as it is considered to be part of the data that this role has.

3.2 Role Managers' Trust in Data Owners

In cryptographic RBAC systems, owners can encrypt their data to any role. Obviously, role managers do not want owners to encrypt malicious data (e.g. virus, malware) to their roles. Therefore, role managers need to decide whether or not to accept data that owners want to assign to them. Having malicious data assigned to a role may result in a low trust value of the role because users who have joined the role will place negative trust records against the role if they detect that the data they get from the role is malicious. In the case where roles are profiting from users' subscriptions, low trust values in roles imply the risk of losing business.

To help role managers detect malicious owners, and hence avoid accepting data from them, another trust model is required to assist role managers in evaluating the trustworthiness of owners. Each time an owner wants to assign data to a role, the role manager will use the trust model to determine whether the data is coming from a trusted owner or not. From a role manager's perspective, a trusted owner should meet the following requirements:

- *Requirement 1: The data from the owner should be the same as its description.* When owners encrypt and assign data to a role, the role manager may not be able to verify each individual record from the owners. When a user who has joined a role finds that the data she or he has accessed is not the data it claims to be or contains malicious records, the user will inform the role manager about the malicious data, and the role manager should place a negative trust record against the owner who owns that data. Then next time this owner wishes to assign data to the role, this trust record will be used by the role manager in making the decision whether or not to accept the data.
- *Requirement 2: The owner should not be considered as untrusted by role managers of any role to which the owner has assigned data before.*
  An owner may have had interactions with more than one role in the system. A trusted owner is supposed to act consistently in the interaction with different roles. An owner may still be considered untrusted even though she or he has good interaction histories with a small portion of roles in the system. Therefore a trusted owner should try to maintain good interaction histories with all the roles in the system. When a role manager is interacting with an owner with which it has not interacted before, the trust opinions from the role managers of other roles can assist this role manager to determine the trustworthiness of the owner.

3.3 Data Owners' Trust in Role Managers

In a cloud storage system, owners are the parties who want to share the data. When they encrypt their data to the roles (in an RBAC system), they need to determine the trustworthiness of the role managers to reduce the risks of unauthorised parties accessing their data. For instance, a data owner may choose not to encrypt the data to a specific role if the role manager is found to have "bad" behaviour histories. Let us now consider some of the key requirements that the owner must consider in determining whether a role manager should be trusted or not. From the owner's perspective, a trusted role manager should meet the following requirements.

- *Requirement 1: The role manager should grant membership to users who are qualified for that role.*
  When a data owner encrypts her or his data to a role, the intention of the owner is to allow the data to be decrypted by the users who are qualified to be in that role. Therefore, it is a basic requirement that the qualified users should have the access to the data. The violation of this requirement is detected by checking whether or not the qualified users can decrypt the data. Not granting the membership to a qualified user is therefore considered as a bad behaviour of a role.
- *Requirement 2: The role manager should not grant membership to users who are not qualified to that role.*
  Another requirement that is expected by a data owner is to prevent users who are not qualified from accessing the permissions to decrypt the data stored

in the cloud. A trusted role manager should only grant membership to a user when the qualifications of the user are verified. Granting membership to an unqualified user is therefore considered as a bad behaviour.

– *Requirement 3: The qualified users in a role should not leak the data to unqualified users.*
  Even if a role manager grants membership only to the qualified users, it is possible that a qualified user may leak the data to unqualified users. For example, consider the situation whereby a user, who is allowed to access the private information that an owner has stored in the cloud, leaks it to another user to whom the owner does not want to reveal the information. The violation of this requirement is detected if it is found that an unqualified user has knowledge of the data. It may or may not be possible to discover this situation. In general, we assume that it is not possible to track down the user who leaks the data; this implies that all the users in that role will need to be under suspicion when such a data leak is detected.
  In a hierarchical RBAC system, a role can inherit permissions from other roles. The users of a role have access to the data encrypted to any of its descendant roles. When a leakage is detected in the data encrypted to one of the descendant roles of a role, the users of this role are also under suspicion as they have the potential ability to cause the leakage. Therefore, when an owner wants to determine the trustworthiness of a role manager, the behaviour histories of role managers of descendant roles of this role need to be taken into account in the evaluation, as the users in this role could be the cause of the leakage of its descendant roles' data which are not reflected in the behaviour history of the role manager of this role.
– *Requirement 4: The role managers of ancestor roles of the role under consideration should be trusted.*
  Since a role's permissions are inherited by all its ancestor roles, when an owner encrypts data to a role, all its ancestor roles also have access to the data. So the data owners need to consider the trustworthiness of not only the role to which they want to encrypt the data, but also of all the ancestor roles of this role, as encrypting data to this role is equivalent to encrypting data to any of the ancestor roles of this role.

3.4 Role Managers' Trust in Data Users

Since roles have the role managers to manage their user memberships, it is role managers' responsibility to build up their own reputation. Therefore it is important for each role manager to be able to evaluate the trustworthiness of users. Role managers can exclude malicious users from the roles; so these users would not affect the trustworthiness of the roles. The ability to evaluate the trust of users is also useful when a user wants to join the role. The role manager can determine the trustworthiness of the new user and decide whether or not to grant the membership to that user. The proper management of users can result in a good behaviour history for a role manager, which in turn affects the owners' decisions on the role manager. From the role managers' perspective, a trusted user should meet the following requirements.

- *Requirement 1: The user should not be involved in the event of leaking resources of the role.*
  When a leak of data is detected, we assume that the role manager can track which users have accessed the data but the role manager does not know who leaked the data. Here we say that a user is involved in leaking data if the data was found to be leaked, and this user has accessed the data before the leakage is detected. A user who has been involved in a leaking event m times will be considered as less trusted than the user who has been involved in a leaking event n times if $m > n$.
- *Requirement 2: The user should not be considered as untrusted by role managers of any role of which the user is or was a member.*
  A user may belong to different roles in a RBAC system. Therefore, the role managers of some other roles to which the user belongs may also hold trust opinions on the user. A trusted user is supposed to act consistently in different roles. Though a user may behave well in one role, she or he will still be considered as untrusted if she or he has bad behaviours in the other roles. The trust opinions of the role managers of other roles on a user can support the evaluation of the user's trustworthiness. A role manager who does not have any trust records in regards to a user (e.g. when a new user requests to join the role) will still be able to determine the trust of the user.

4 User's Trust Models for Cryptographic RBAC

In this section, we consider the user's trust models for a cryptographic RBAC system. There are three types of entities in our trust models, *Owner*, *User* and *Role*. Our trust models can assist a *User* to decide whether a *Role* to interact with is trusted, and assist a *Role* in determining the trustworthiness of an *Owner*. We first review these three entities. *Owner* is the entity who owns the data and stores it in an encrypted form for particular roles in the cloud. *User* is the entity who wishes to access the data stored in the cloud. *Role* is the entity that associates users with the access to owners' data, and each role manages the user membership of itself. Here when we say that users are managed by a role, we refer to the managers of the role who determine the user set of that role.

In our trust models, we assume that all the feedback and recommendations provided are honest. In other words, we assume that the trust system has the ability to verify the submitted feedback and recommendations, and only the valid ones will be considered in the trust evaluations.

4.1 User-Role RBAC Trust Model

In this subsection, we consider the trust model for user's trust in roles in a RBAC system.

Definition 1 ***(Interaction).*** From a user's perspective, an interaction is a transaction in which a user accesses data that is encrypted to a role to which the user belongs.

A successful interaction is an interaction where a user has successfully accessed the data. An unsuccessful interaction is an interaction where a user failed in accessing the data to which she or he should have legitimate access. Next we define two types of unsuccessful interactions.

User Management Failure: User management failure is an unsuccessful interaction caused by incorrect user membership management of a role; that is, the role did not grant the membership to the user even when the user qualifies for the role.

Permission Management Failure: Permission management failure is an unsuccessful interaction where the data encrypted to a role is invalid, or the data is not encrypted to the role. In other words, the owner of the data did not encrypt the data to the role in question or encrypted an invalid data to the role.

Definition 2 ***(Trust Vector).*** We define a trust vector to represent the behaviour history of a role as follows:

$$\boldsymbol{v} = (r, s_U, s_P)$$

In the trust vector, r is the value related to successful interactions that users have had with a given role, s_U is the value related to *User Management Failure* of the role, and s_P is the value related to the *Permission Management Failure.*

Using the function $\mathcal{E}$ in Eq. 1, we define the trust function $T(\boldsymbol{v})$ that represents the trust value derived from the trust vector $\boldsymbol{v}$ as

$$\mathcal{T}(\boldsymbol{v}) = \mathcal{E}(r, s_U + s_P)$$

Definition 3 ***(Interaction History).*** We assume that there exists a central repository in the system that collects and stores the ratings from users on the interactions between users and roles. We define the trust record history derived from the ratings of the role R from n users as

$$Hist_{\mathcal{U}}(R) = \{H_1^R, H_2^R, \cdots, H_n^R\}$$

Each entry H_i^R in $Hist(R)$ is defined as a pair of parameters, $H_i^R = \langle U_i, \boldsymbol{v}_{i,R} \rangle$, where $\boldsymbol{v}_{i,R} = (r, s_U, s_P)$ is a trust vector that represents the trust record of interactions that the user U_i has had with the role R. r is the number of U_i's positive feedbacks on the interactions with R, s_U is the number of negative feedbacks on the interactions with R due to *User Management Failure*, and s_P is the number of negative feedbacks on the interactions with R due to *Permission Management Failure.*

In a cryptographic RBAC system, a user who belongs to a role not only has access to the data of the role, but also has access to the data of descendent roles. Therefore, an invalid resource from a descendent role may also cause an unsuccessful interaction. Since a role knows whether a resource comes from its descendent roles, we assume that users give feedback to the roles to whom the resources are directly assigned; that is, if a user detects an invalid resource from a descendent role, she or he will update the feedback for the descendent role directly instead of the role she or he belongs to.

As discussed in Sect. 3.1, from the users' perspective, the trustworthiness of a role is affected by the interaction history of the role and its descendant roles. Therefore users need to consider the following types of trust classes when evaluating the trustworthiness of roles.

Individual Trust. Individual trust is a belief that is derived directly from the interaction history of the role R.

When a user U_k wishes to evaluate the trust value of a role R, the user first obtains the interaction history $Hist_{\mathcal{U}}(R)$ of the role from the central repository. Assume that w_u is the weight that the user U_k assigns to the feedbacks from other users. Then the individual trust value of the role R can be computed as follows,

$$T_{\mathcal{U}}(R)^D = \mathcal{T}(\boldsymbol{v}_{k,R}^D),$$

$$\text{where} \quad \boldsymbol{v}_{k,R}^D = \boldsymbol{v}_{k,R} + w_u \cdot \sum_{i=1, i \neq k}^{n} \boldsymbol{v}_{i,R}$$

where the trust vector $\boldsymbol{v}_{k,R}^D$ is a combination of all trust vectors in $Hist_{\mathcal{U}}(R)$ considering the weighting for the trust vectors from other users.

Inheritance Trust. Inheritance trust is a belief that is derived from the interaction history of the descendant roles of a given role.

Assume a role R has m immediate descendant roles $\{R_1, \cdots, R_m\}$, and a weight vector $\boldsymbol{w}_{R_i}$ is defined as $(w_{R_i}^R, 0, w_{R_i}^R)$ where $w_{R_i}^R \in [0,1]$ is the weight assigned to the inheritance relationship between R and R_i. The second element is set to zero because *User Management Failure* is not considered in inheritance trust as user management of descendant roles will not cause any unsuccessful interaction for this role. The inheritance trust of roles in a hierarchy is computed as follows:

$$T_{\mathcal{U}}(R)^I = \mathcal{T}(\boldsymbol{v}_{k,R}^I),$$

$$\text{where} \quad \boldsymbol{v}_{k,R}^I = \sum_{i=1}^{m} [(\boldsymbol{v}_{k,R_i}^D + \boldsymbol{v}_{k,R_i}^I), \boldsymbol{w}_{R_i}]$$

In the above equation, $[\boldsymbol{v}, \boldsymbol{w}] := \boldsymbol{v}^T \boldsymbol{w}$ is the usual dot product on $\mathbb{Z}_q^3$.

Combination Trust. To compute the trust value of a role, we define a combination trust function for a role R as $T_{\mathcal{U}}(R)$ to combine the above described two types of trust together. Assume that $w \in [0,1]$ is the weight of the inheritance trust. The trust value is computed as

$$T_{\mathcal{U}}(R) = (1-w) \cdot T_{\mathcal{U}}(R)^D + w \cdot T_{\mathcal{U}}(R)^I$$

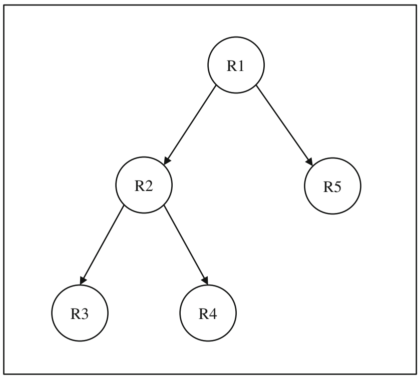


Fig. 1. Hierarchical RBAC example.

4.2 Example of User-Role RBAC Trust Model

Now we use an example to show how the users' trust in a role is affected by feedback for different roles in a RBAC system. In this example, we consider all the bad feedback as *Permission Management Failure*, as our intention is to show how the role hierarchy affects the trust value of roles. Consider the role hierarchy example shown in Fig. 1.

In Fig. 1, the role R_1 inherits from role R_2 and role R_5, and the role R_2 inherits from R_3 and R_4. We set the weight between every two roles and the weight of other owners' feedback to 1; that is, the weight vector for each role R_k where $k \in [1,5]$ is defined as $\boldsymbol{w}_{R_i} = (1,0,1), \forall\, i \in [1,5], i \neq k$, and $w_u = (1,1,1)$. When a user wants to access a resource that has been assigned to the role R_2, she or he will need to evaluate the trust value of R_2 to decide whether R_2 is reliable to join. In Fig. 2, we show the trust values of R_2 when only different individual roles in the RBAC system have feedback. For example, the curve for $R_1, GFP = 75\,\%$ shows the trust values of R_2 when only R_1 in the RBAC system has feedback, and 75 % of the feedback is positive.

When the good feedback percentage is 75 %, the trust value for R_2 goes up with the increasing number of feedbacks that R_2 and R_3 have. This trend implies that the more resources a role has, the more impact the good feedback percentage has on the trust value of the role. Note that the feedback for R_1 does not affect the trust value of R_2. This is because an untrusted R_1 will not cause an unsuccessful interaction of R_2. When the feedback is only given for R_2, the increase in the trust value is the fastest. This is because the individual trust of the role has more weight than the inheritance trust by our assumption. It is clear that the increase in the trust value of R_2 is slower when the feedback is for R_3 only, because inheritance trust has less weight in this example.

When the good feedback percentage is 25 %, the trust value for R_2 goes down with the increasing number of feedbacks that R_2 and R_3 have. Similarly, this trend implies that the more resources a role has, the more impact the good feedback percentage has on the trust value of the role. The feedback for R_1 does

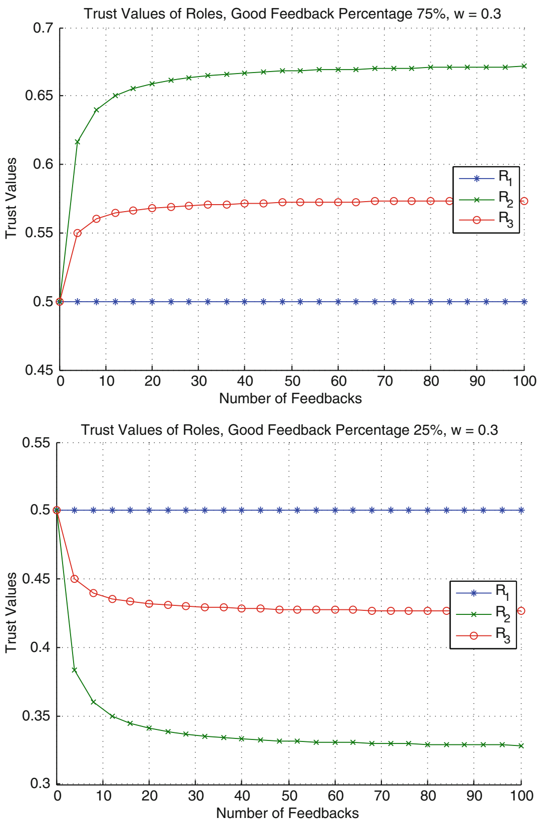


Fig. 2. Trust values on R_2 for users from feedback on different roles.

not affect the trust value of R_2 either. When feedback is only given for R_2, the decrease in the trust value is the fastest. This is because the individual trust of the role has more weight than the inheritance trust by our assumption. Therefore the decrease in the trust value of R_2 is slower when the feedback is for R_3 only.

From Fig. 2, we see that the feedbacks for different roles in the system have different impacts on the trust value of R_2. Firstly, the feedback for ancestor roles does not affect the trust of the role. Secondly, the more resources that have been assigned to a role, the more impact the feedback for the role will have on its ancestor roles as well as itself. These results show that our users' trust model is useful in assisting users to determine properly the trust of roles in RBAC systems.

4.3 Role-Owner RBAC Trust Model

In the case when any owner can choose roles to encrypt their resources to, assigning malicious resources or invalid resources to a role may cause the *Permission*

Management Failure of the role. Therefore, it would be useful to have a trust model to assist role managers in determining the trustworthiness of an owner, and hence decide whether or not to accept the resources from the owner.

As discussed in Sect. 3.2, the trust requirement on owners is simpler compared to the users' trust in roles, and we see some important differences. The trust in owners is independent from the role hierarchy; that is, the role hierarchy does not affect the trustworthiness of owners. We note that a general trust model can be used in this scenario. For completeness purposes, we also give the definition of the trust model for the role managers' trust in data owners in this subsection.

Definition 4 ***(Interaction).*** From a role manager's perspective, an interaction with an owner is a transaction in which an owner assigned a resource to that role, and that the role manager has accepted the resource.

Definition 5 ***(Trust Vector).*** We define a trust vector to represent the behaviour history of an owner as follows:

$$\boldsymbol{v} = (h, s)$$

where h is the value related to resources owned by the owner, and s is the value related to malicious or invalid resources owned by the owner.

Using the function $\mathcal{E}$ in Eq. 1, we define the trust function $T(\boldsymbol{v})$ that represents the trust value derived from the trust vector $\boldsymbol{v}$ as

$$\mathcal{T}(\boldsymbol{v}) = \mathcal{E}(h - s, s)$$

Definition 6 ***(Interaction History).*** We assume that there exists a central repository in the system that collects and stores the behaviour histories provided by role managers to which the owner has assigned the resources. We define the trust record history provided by a set $\mathcal{R}$ of n roles as

$$Hist_{\mathcal{R}}(O) = \{H_1^O, H_2^O, \cdots, H_n^O\}$$

Each entry H_i^O in $Hist(O)$ is defined as a pair of parameters, $H_i^O = \langle R_i, v_{i,O} \rangle$ where $\boldsymbol{v}_{i,O} = (h, s)$ is a trust vector that represents the trust record of the owner O on the resources that she or he has assigned to the role R_i. h is the total number of O's resources that has been assigned to R_i, and s is the number of bad resources assigned by O.

We assume that an owner O has a resource and wants to assign it to a role R_k. When this resource is assigned to the role R_k, R_k updates the trust record of the owner by increasing the value h in the trust vector H_k^O of O by 1. Now assume that a user has found the resource to be invalid, and then she or he reports to the role of this resource. If the role has confirmed that the user's complaint is true after verifying the resource, R_k will find out that it is O who uploaded this resource, and R_k will increase the value s in trust vectors H_k^O for this owner by 1.

A user that belongs to a role has the permission to access resources of the descendant roles of the role. When the user reports a bad resource from its descendant role, this role may not be able to identify the owner of the resource as the resource is not assigned to this role directly. Hence the role cannot update the trust records of the owner. In this case, the role can notify all its descendant roles about this bad resource, and the role to which the resource is assigned to will update the trust record of the owner who owns the resource.

Assume that w is the weight that the role R_k assigns to the feedback from other roles. Taking as input the interaction history of an owner, the trust value of the owner can be computed as follows:

$$T_{\mathcal{R}}(O) = \mathcal{T}(\boldsymbol{v}_{k,O}^{T}), \quad \boldsymbol{v}_{k,O}^{T} = \boldsymbol{v}_{k,O} + w \cdot \sum_{i=1, i \neq k}^{n} \boldsymbol{v}_{i,O}$$

This trust value is evaluated based on a combination of all trust records in $Hist_{\mathcal{R}}(O)$ considering the weighting for the trust records from other roles.

5 Architecture for User's Trust Models

In this section, we present the design of a secure cloud storage system by combining the user's trust models for RBAC proposed in Sect. 4 with a cryptographic RBAC system. This architecture provides a practical solution for building a reliable and trusted RBAC system while retaining the use of cryptographic techniques. We have implemented a prototype of this architecture and have been conducting a range of experiments.

5.1 System Overview

Consider the system architecture shown in Fig. 3. Each solid line in the figure shows the communication channel set by the system between two components

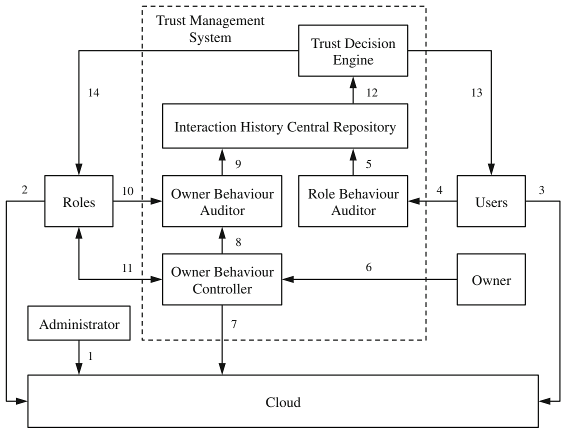


Fig. 3. Architecture for using user's RBAC trust models in a cryptographic RBAC system.

joined together by the line, and the arrows indicate the direction in which the information flows. Since our trust models are based on cryptographic RBAC schemes, our system contains all the entities that a cryptographic RBAC scheme has, including an administrator, roles, users, and owners. The administrator is the certificate authority of the RBAC system, and it generates the system parameters and issues all the necessary credentials. In addition, the administrator manages the role hierarchy of the system. To put a role into the role hierarchy, the administrator needs to compute the parameters for that role. These parameters represent the position of the role in the role hierarchy. They are stored in the cloud, and are available publicly. Roles are the entities that associate users and owners together. Each role has its own role parameters which define the user membership. These role parameters are stored in the cloud, and a role needs to update them in the cloud when updating the user membership of the role. Owners are the parties who possess the data and want to store the encrypted data in the cloud for other users to access, and they specify the roles who can access the data. In the RBAC model, they are the parties who manage the relationship between permissions and roles. Users are the parties who wish to acquire certain data from the cloud. When a user wishes to access stored data in the cloud, she or he first sends the request to the cloud, and decrypts the data upon receiving the response from the cloud.

In addition to these four entities in a basic cryptographic RBAC scheme, our trust models enhanced the cryptographic RBAC system by integrating an extra trust management system, which consists of five components.

Central Repository. In our trust models, all the interaction histories and trust records related to roles and users are stored in a central repository. The central repository is used to keep the records of all these interaction histories and trust records which are used by the Trust Decision Engine (described below) in evaluating the trust value of roles and owners. Any entity that is residing outside the trust management system is not able to access the central repository.

Role Behaviour Auditor. In order to protect the integrity of the feedback on roles, a role behaviour auditor collects the feedback on roles from users. The role behaviour auditor needs to ensure that a user who uploads feedback against a role has been granted the membership of the role or an ancestor role of that role. All the valid feedback will be forwarded to the central repository, and invalid feedback will be discarded.

Owner Behaviour Auditor. An owner behaviour auditor is an entity to collect the feedback on owners' behaviour. However, unlike the role behaviour auditor, the owner behaviour auditor listens for feedback on two channels. One is from the roles who may report the invalid data, and another is from the owner behaviour controller which reports the ownership of the stored data in the cloud. This auditor will determine whether an owner has uploaded any malicious or invalid data to the cloud, and can update the central repository.

Owner Behaviour Controller. Owner behaviour controller acts as a proxy server between owners and the cloud. It controls and forwards the owners' encrypted data to the cloud. The controller can decide whether to store data in the cloud based on the decision from the role to which the data is assigned. The controller will inform the owner behaviour auditor about which owner the uploaded data belongs to.

Trust Decision Engine. The trust decision engine is the entity which evaluates the trust of the roles for users and the trust of the owners for roles. The trust decision engine takes as input the interaction histories or trust records stored in the central repository, and outputs the trust value of a particular role or owner.

5.2 System Workflow

All the entities in the system are connected through different communication channels which are labelled with numbers in Fig. 3. We explain how the system works by describing the information flow through these channels.

First, the administrator initialises the system and specifies the role hierarchy of the system. The generated system parameters are uploaded to the cloud via (1). Roles grant the membership to users, and upload role parameters to the cloud via (2). Users download and decrypt data from the cloud via (3). When an owner wants to encrypt and store data in the cloud to a particular role, she or he first encrypts the data and sends a request to the owner behaviour controller via (6). Then the owner behaviour controller notifies the role via (11) and forwards the request to the cloud through (7) if the role agrees to accept the data from this owner. The cloud then communicates with the owner as in a normal cryptographic RBAC scheme. The controller also sends the owner behaviour auditor the information about the owner's identity and the resource's identity via (8).

When a user wants to access a resource in the RBAC system, the system first returns a list of roles who claim to have this resource. Then the user requests the trust evaluation on these roles from the trust management system. The trust value of the roles will be returned to the user through the (13). The user may choose a role who has the highest trust value to send the join request. When a user has found that the data she or he has accessed from the role is malicious or invalid, she or he then provides feedback on the role to whom the resource is encrypted to the role behaviour auditor through (4). Once the role behaviour auditor verifies that the feedback is from an authorised user, it will forward the feedback to the central repository.

When a negative feedback of a role has been raised by a user because of an invalid resource, the role will send the identity of the resource to the owner behaviour auditor via (10) if it believes that the resource was invalid when the owner uploaded the resource. The auditor then updates the trust records of the owner of this resource to the central repository via (9). When an owner wants to assign a resource to a role, the role can ask the trust management system about the trust evaluation for an owner, and the trust value will be returned by the trust decision engine through (14). Upon receiving the trust values for the owner from the trust decision engine, the role can inform the owner behaviour controller via (11) whether to accept the data. Moreover, this trust evaluation process can be made automatically by connecting the owner behaviour controller to the trust decision engine directly. Roles can pre-determine a trust threshold for accepting data from owners. Every time an owner wants to upload a resource, the owner behaviour controller can check the trust value of the owner from the trust decision engine directly, and decide whether to accept the resource by comparing the trust value with the role's threshold.

6 Owner's Trust Models for Cryptographic RBAC

In this section, we consider the remaining two trust relationships in cryptographic RBAC systems, data owners' trust in role managers and role managers' trust in users. We refer to these trust models as Owner-Role RBAC and Role-User RBAC trust models respectively. The Owner-Role RBAC trust model assists the data owners to evaluate the trust in role managers in a RBAC system and use this trust evaluation to decide whether to store their encrypted data in the cloud for particular roles. The Role-User RBAC trust model helps the role managers to evaluate the trust in users in the RBAC system and use this trust in deciding whether to grant membership to the users. These trust models can not only prevent the owners from interacting with role managers which have a poor track record in terms of carrying out their functions properly, but also assist the role managers to identify the malicious users who caused the negative impacts on the role managers' trustworthiness. This can in turn be used to reduce the risks associated with interacting with the RBAC system for the data owners and help role managers to keep the RBAC system authentic. The Owner-Role RBAC and Role-User RBAC trust models are independent of each other and serve different purposes.

6.1 Owner-Role RBAC Trust Model

An important feature of the proposed trust models is that they take role inheritance into account. Since our trust models are for cloud storage systems dealing with hierarchical RBAC schemes, the trustworthiness of a role manager is also affected by the historical behaviour of the role managers of its ancestor roles and/or descendent roles. Hence in our trust evaluation, we take into account the impact of role hierarchy and inheritance on the trustworthiness of the role managers and users.

In Sect. 3.3, we discussed the trust requirements that an owner should consider when deciding whether or not to trust a role. From that discussion, we see that the factors which can affect the owners' decision come from the interaction history of the role with whom owners have interacted as well as its ancestor roles and descendant roles. When an owner evaluates the trust of a role R, the owner needs to consider the interaction history of other roles that have inheritance relationships with the role R.

First we consider the inheritance trust where only the interaction history of the descendant roles is included. When an owner detects that a user of a descendant role R_d of a role R leaked data to unqualified users, the feedback that the owner provided should not only be applied to that descendant role R_d, but should also affect the trust of R as users belonging to the role R also have the access to owner's data assigned to R_d and hence are under suspicion of causing an unsuccessful interaction. Therefore, while evaluating the trust of the role R, the interaction history from all its descendant roles including R_d needs to be considered.

Then we look at the interaction history of ancestor roles' impacts on the trust of the role. Consider the scenario where the trust score of a role is higher than the trust score of one of its ancestor roles. Then the owners will trust this role at the same level as its ancestor role which has a lower trust score, as the users of its ancestor role have the same level of access as the users in this role. So the trust score of the role will be the minimum value of the trust of this role and the trust of all its ancestor roles.

6.2 Role-User RBAC Trust Model

Since the trustworthiness of a role is primarily determined by the behaviour of users of the role, it is important for the role to ensure that only users with good behaviour are granted membership. If roles do not have a way to evaluate the trust of their users, it would be difficult for them to distinguish the malicious users from those with good behaviours.

When a role wishes to evaluate the trust value of a user, the role first considers the user's trust record rated by the role itself. Then the role needs to consider the trust records of the user from other roles in the system. Therefore, the trust score of the user is evaluated based on all the trust records of this user including the trust records from other roles.

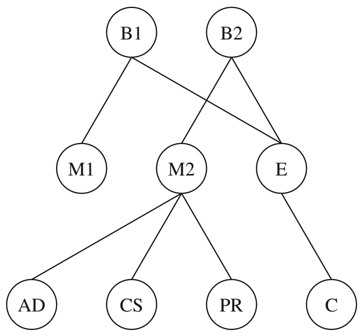
This trust model can either work independently or work together with the Owner-Role RBAC trust model. Roles can use this model to periodically check the trust value of the existing users in the roles, and revoke the memberships from users whose trust values are below the preset threshold. This trust model can also be used by roles to determine the trust value of a new user requesting to join; the request from the users whose trust values are below the threshold will be rejected.

7 Application Scenario

In this section, we use application examples to illustrate how the trust models in the framework can be used in a cloud service application to enhance the quality of security decision making.

7.1 Application for User's RBAC Trust Models

In this subsection, we describe a digital library system which uses our proposed user's trust models to illustrate how the trust models can assist security decision making in this system. The digital library system uses an external public cloud



B1	Business 1
B2	Business 2
M1	Marketing 1
M2	Marketing 2
E	Economics
AD	Advertising
CS	Customer Service
PR	Public Relations
C	Commerce

Fig. 4. Digital library system example for user's RBAC trust models.

to store all the digital format resources such as books, papers, theses, and other types of publications. There are many distributors who use the platform provided by the digital library system to share digital resources. Each distributor can get the authorisations for sharing the digital resources from the publishers directly. A party who subscribes to a distributor can access all the resources of the distributor. Assume that the distributors have two types of subscription licenses; personal licenses that allow only the subscribed user to access the resources, and business licenses that allow another distributor to resell the resources to other users or distributors.

Now let us consider the example of a distributors' network for this digital library system. The hierarchical relationship of the distributors is shown in Fig. 4. In this system, distributors choose the resources to share by their categories. The distributors AD, CS, PR, C, and M_1 get the authorisations for selling digital resources in the categories *Advertising*, *Customer Service*, *Public Relations*, *Commerce* and *Marketing* respectively from the publishers. Distributors M_2 and E sell a wider ranger of resources which cover all the categories in *Marketing* and *Economics* respectively, and these two distributors get authorisations from the distributors of sub-categories instead of the publishers directly. Note that the categories of resources sold by M_1 and M_2 are overlapped. The difference is the channels they get the resources from: M_1 from publishers, and M_2 from sub-distributors. Similarly, distributors B_1 and B_2 get authorisations from M_1, E, and M_2, E respectively, and their resources both cover the categories *Business.*

To use cryptographic RBAC schemes to protect the resources so that only the authorised users can access them, the administrator of the digital library system first sets up the system parameters based on the relationships of the distributors. Then the publishers can encrypt their resources to the distributors whom they authorised to sell the resources. Here we consider the distributors as roles in the RBAC, and publishers as owners of the resources. When a user subscribes to a distributor, the distributor simply adds the user to the role. Then the user can use the key given by the system administrator to decrypt the resources of the role. Because the cryptographic RBAC schemes support role hierarchy, in this example, users who subscribed to the role M_2 can also access the resources of the role AD, CS and PR, and users subscribed to B_1 can access the resources of all the roles M_1, E, and C.

First let us consider how the trust model can assist the users. Assume that the distributor M_2 also gets some resources, which the distributors AD, CS, and PR do not have directly from the publishers. To save the cost of storing resources in the cloud, M_2 chooses to reprint some resources in a lower quality to reduce the file size. Users subscribed to M_2 may give negative feedbacks on M_2 because they have difficulties in reading some of the resources. Later on, when a user want to access marketing resources, she or he evaluates the trust of M_1 and M_2, and the trust model will output a higher trust value for M_1 than for M_2 because of the negative feedbacks of M_2. Then the user will know the quality of resources from M_1 is better than those from M_2. However, the distributors AD, CS, and PR will not be affected because the poor quality resources are not coming from them. When a user wants to subscribe to a distributor for *Business*, B_2 will have lower trust value than B_1 as resource the user would get from B_1 may come from M_2.

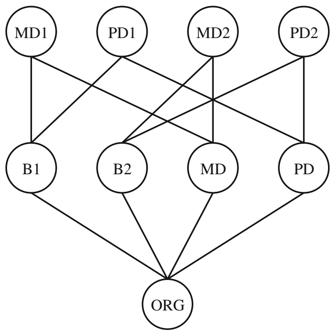
Now let us look at the trust model for roles' trust in owners. Assume that publishers want to promote their digital resources, and they actively assign their resources to distributors. The resources that have come from some publishers may be of poor quality or alternatively some resources are not what the publishers claim them to be. The distributors may not be able to verify each individual resource due to the lack of expertise in certain areas. When users complain about a bad resource, the role can give a negative feedback on the publisher who owns the resource, after confirming that the users' complaints is valid. The feedback of the publisher can be accessed by all the distributors so they can avoid using this publisher in the future.

7.2 Application for Owner's RBAC Trust Models

Finally, we consider another application scenario based on a digital library system. Similarly, assume that the digital library system uses an external cloud storage platform to store all the resources. A party can subscribe to the publisher for particular resources in order to access the resources stored in the cloud, and the subscription to a publisher needs to be authorised by the publisher. The publisher may reject the subscription request for reasons such as the party is not reliable in paying the subscription or the party has the potential to leak the resources to unauthorised parties.

Now assume that there is an organisation with several branches in different geographical locations and that each branch consists of several departments. When employees of the organisation need to access the digital resources stored in the cloud, the relevant department or the branch (where the employee works) can subscribe to the publisher. Let us assume that the organisation uses a RBAC system to control the access to resources, and the role hierarchy is shown in Fig. 5.

In this example, the organisation consists of two branches B_1 and B_2, and each branch has two departments MD_1 and PD_1, MD_2 and PD_2 respectively.



ORG	Organisation
B1	Branch 1
B2	Branch 2
MD	Marketing Department
PD	Production Department
MD1	Marketing Department 1
PD1	Production Department 1
MD2	Marketing Department 2
PD2	Production Department 2

Fig. 5. Digital library system example for owner's RBAC trust models.

Assume that the head office has two head departments MD and PD which manage the relevant departments in both branches. Recall that a role can inherit from other roles in the RBAC system. For example, when PD has subscribed to a resource from a publisher, both PD_1 and PD_2 will have access to the resource. Similarly, a resource subscribed by the role ORG can be accessed by all the roles in the system.

By using a cryptographic RBAC scheme in this system, a publisher is able to encrypt the resource to the branch or department (who subscribes to the resource) and store it in the cloud so that the employees who work in the branch or department can access it. There is an assumption that these employees are trusted and will not redistribute resources of the publisher to employees who are not in that branch or department. However, it is possible that an employee leaks the content of a resource to others. Therefore, the publishers will need a trust system to assist them in identifying the roles who have malicious users, and hence avoid accepting the subscriptions from them.

Let us now consider how our trust model can be used in this system to assist the publishers (owners). Assume that no publisher has ever interacted with the role PD_1, PD_2, and now a publisher wishes to evaluate the trust of these two roles. We also assume that B_1, MD_1 and PD_1 are the same as B_2, MD_2 and PD_2 in terms of the number of employees and percentage of good feedbacks for B_1 is higher than that for B_2. Since the trust of the role is affected by descendant roles in our model, the publisher will get the result where the role PD_1 is more trusted than PD_2. This result aligns to the fact that if the branch B_1 is more trusted than the branch B_2, then the department PD_1 of the branch B_1 will also be considered more trusted than the department PD_2 of the branch B_2.

Now assume that the role ORG only has good feedbacks; that is, the resources the role ORG has subscribed have never been leaked. Since the trust value of a role is taken from the minimum value of the trust value for all its ancestor roles, when a publisher evaluates the trust of the role ORG, its trust value may be low if the good feedback percentage for B_2 is low. This is because employees in the role B_2 inherit permissions from the role ORG, and employees in this branch could potentially leak the resources.

When the role B_2 realises that its trust value is low, it may decide to warn or even exclude some potential malicious users. Then our Role-User RBAC trust model can be useful to assist roles in identifying the potential malicious users. Our trust model allows the trust for employees in B_2 to be evaluated based on the feedbacks from all the roles in the organisation. That is, if an employee was working in the branch B_1 and relocated to B_2 recently, the feedbacks on the user from B_1 when the user was working in B_1 is also taken into account when B_2 determines the trustworthiness of the user.

From this digital library system example, we see that our owner's trust model can be used in the cloud storage system using cryptographic RBAC schemes where role managers themselves have the flexibility in managing the user membership.

8 Related Works

There have been some related works which have addressed only trust on users in RBAC systems. Reference [2] proposed a trust model for RBAC system which considers users' trust by assigning trust levels to users. These trust levels are based on a number of factors such as user credentials, user behaviour history and recommendations from other users. Trust levels are then mapped to roles. Another trust model for RBAC was proposed in [13] to assist roles with the decision of user-role assignment based on a wide range of criteria of users, including behaviour history and reputation. In [4], a trust model for RBAC was introduced which evaluates the trust in the users based on user behaviours and context, in a context-aware access control model. Another trust model was discussed in [14] which also uses trust level to determine the access privileges of users. All these trust models only consider the trust on users in a RBAC system. None of these works address the other trust relationships in the RBAC system. The trust for role managers is critical in cloud storage systems which has been addressed in this paper. Our trust models have also addressed the roles's trust on data owners and users. Another difference between our model and the previously proposed ones is that our trust models work in RBAC systems which use cryptographic RBAC schemes. That is, our models take into account cryptographic operations and the access privileges to decrypt the data stored in the cloud, which none of the previous works address.

9 Conclusions and Future Work

In this paper, we have addressed trust issues in cryptographic role-based access control systems for securing data storage in a cloud environment. We have proposed trust models for different entities in RBAC systems which are using cryptographic RBAC schemes to secure stored data. These trust models assist the users and role managers to determine the trustworthiness of individual role managers and data owners, and data owners and role managers to evaluate the trustworthiness of individual role managers and users, in the RBAC system respectively. They allow the users and the data owners to perform the trust evaluation to decide whether or not to access/share a resource from/to a particular role. Our trust model takes into account role inheritance and hierarchy in the evaluation of trustworthiness of roles. The models also enable the roles to use the trust evaluation in their decision to accept the resources from a particular owner and grant the role membership to a user. We have given the design of an architecture of a trust-based cloud storage system which has integrated the user's trust models with the cryptographic RBAC schemes. We have also described the application of the proposed trust models by considering a practical scenario and illustrating how the trust evaluations can be used to reduce the risks and enhance the quality of security decision making by different entities of the cloud storage service.

The proposed trust models used a centralised trust management system to assist entities with their trust evaluations. Though these entities in the system still need to trust the centralised trust management components, we believe that this approach has much improved the cases where entities need to trust every other individual entity in the system. We note that the auditing components in our designed architecture need to collect all the provided feedback. In large-scale systems, the load of these auditing components could be high. One solution to this issue is using decentralised auditing components which will be considered in our future work. In addition, we only considered two types of feedbacks in our trust models, positive and negative. However, a user who has unsatisfactory experiences with roles may want to provide varying levels of negative feedback. For example, the user may have retrieved a malware instead of valid data from a role and a poor quality data instead of a good quality one from the same role. It is clear that some cases are more harmful than others, and the user may want to rate the role less untrusted when it is more harmful. We will also consider this issue in our future work.

References

1. Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. ACM Trans. Comput. Syst. **1**(3), 239–248 (1983)
2. Chakraborty, S., Ray, I.: TrustBAC - integrating trust relationships into the RBAC model for access control in open systems. In: 11th ACM Symposium on Access Control Models and Technologies, SACMAT 2006, pp. 49–58. ACM, 7–9 June 2006
3. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Encryption policies for regulating access to outsourced data. ACM Trans. Database Syst. **35**(2), 12:1–12:46 (2010)
4. Feng, F., Lin, C., Peng, D., Li, J.: A trust and context based access control model for distributed systems. In: 10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008, pp. 629–634. IEEE, 25–27 September 2008

5. Ferraiolo, D.F., Kuhn, D.R.: Role-based access controls. In: Proceedings of the 15th NIST-NCSC National Computer Security Conference, pp. 554–563. National Institute of Standards and Technology, National Computer Security Center, 10–13 October 1992
6. Jøsang, A., Ismail, R.: The beta reputation system. In: Proceedings of the 15th Bled Conference on Electronic Commerce (2002)
7. Miklau, G., Suciu, D.: Controlling access to published data using cryptography. In: Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, pp. 898–909, 9–12 September 2003
8. Mui, L., Mohtashemi, M., Ang, C., Szolovits, P., Halberstadt, A.: Ratings in distributed systems: a bayesian approach. In: Workshop on Information Technologies and Systems (2001)
9. Mui, L., Mohtashemi, M., Halberstadt, A.: A computational model of trust and reputation for e-businesses. In: HICSS, p. 188 (2002)
10. Samarati, P., De Capitani di Vimercati, S.: Data protection in outsourcing scenarios: issues and directions. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, pp. 1–14. ACM, 13–16 April 2010
11. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Comput. **29**(2), 38–47 (1996)
12. Sandhu, R.S., Ferraiolo, D.F., Kuhn, D.R.: The NIST model for role-based access control: towards a unified standard. In: ACM Workshop on Role-Based Access Control, RBAC00, pp. 47–63 (2000)
13. Takabi, H., Amini, M., Jalili, R.: Trust-based user-role assignment in role-based access control. In: AICCSA, pp. 807–814. IEEE, 13–16 May 2007
14. Toahchoodee, M., Abdunabi, R., Ray, I., Ray, I.: A trust-based access control model for pervasive computing applications. In: Gudes, E., Vaidya, J. (eds.) Data and Applications Security XXIII. LNCS, vol. 5645, pp. 307–314. Springer, Heidelberg (2009)
15. Zhou, L., Varadharajan, V., Hitchens, M.: Enforcing role-based access control for secure data storage in the cloud. Comput. J. **54**(13), 1675–1687 (2011)
16. Zhu, Y., Hu, H., Ahn, G.-J., Wang, H., Wang, S.-B.: Provably secure role-based encryption with revocation mechanism. J. Comput. Sci. Technol. **26**(4), 697–710 (2011)