Trust Enhanced Cryptographic Role-Based Access Control for Secure Cloud Data Storage

Lan Zhou, Vijay Varadharajan, and Michael Hitchens

Abstract— **Cloud data storage has provided significant benefits by allowing users to store massive amount of data on demand in a cost-effective manner. To protect the privacy of data stored in the cloud, cryptographic role-based access control (RBAC) schemes have been developed to ensure that the data can only be accessed by those who are allowed by access policies. However, these cryptographic approaches do not address the issues of trust. In this paper, we propose trust models to reason about and to improve the security for stored data in cloud storage systems that use cryptographic RBAC schemes. The trust models provide an approach for the owners and roles to determine the trustworthiness of individual roles and users, respectively, in the RBAC system. The proposed trust models consider role inheritance and hierarchy in the evaluation of trustworthiness of roles. We present a design of a trust-based cloud storage system, which shows how the trust models can be integrated into a system that uses cryptographic RBAC schemes. We have also considered practical application scenarios and illustrated how the trust evaluations can be used to reduce the risks and to enhance the quality of decision making by data owners and roles of cloud storage service.**

Index Terms— **Role-based access control, trust model, cryptographic RBAC, secure cloud data storage.**

I. Introduction

THERE has been a rapid growing trend in the recent times in using online services. A major benefit of using online services is that users can store their data online and access it from anywhere. However, many online service providers do not have the capacity to store large amount of users' data due to high maintenance cost and complexity. Cloud services such as cloud storage services are providing solutions to address these issues with the ability to store and manage increasing amount of users' data stored online. Online service providers can outsource users' data to the public cloud while focusing on the service quality. Since a public cloud is an open platform, and can be subjected to malicious attacks from both insiders and outsiders, this has raised several security issues such as how to control and prevent unauthorised access to data stored in the cloud; this also applies to cloud providers themselves as the owners of the data may not wish the cloud providers to view or access its content.

One approach to protect the privacy of the data stored in the cloud is using access controls. Many access control models have been proposed over the years in the literature. In this context, 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 [1], 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 [9], and the RBAC standard was proposed in 2000 [10].

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. However in a distributed system such as a cloud, there may not exist such a central authority as the data may be stored in distributed data centres which cannot be under the control of a single authority. 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 specify correctly the access control policies and enforce them properly.

In a cloud data storage system, the data owners would wish to specify the policies as to who can access their data and the cloud providers are required to correctly enforce the policies that the data owners have specified. In order to enforce the specified access control policies before putting the data onto the cloud, the data owners can encrypt the data in the way that only users that the owners wished to allow as specified in the access control policies are able to decrypt and access the data. Several cryptographic schemes, such as the schemes in [2]–[6], have been developed to enforce access policies on outsourced data. These schemes combine cryptographic techniques and access control to protect the privacy of the data in an outsourced environment. The paper [6] specifically addressed the security issue of RBAC in cloud systems and proposes a new scheme called Role-based Encryption (RBE). It is worth noting that the security of a RBAC system using one of these

Manuscript received February 8, 2015; revised June 10, 2015; accepted June 30, 2015. Date of publication July 13, 2015; date of current version September 15, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. H. Vicky Zhao.

The authors are with the Advanced Cyber Security Research Centre, Department of Computing, Macquarie University, Sydney, NSW 2109, Australia (e-mail: lan.zhou@mq.edu.au; vijay.varadharajan@mq.edu.au; michael.hitchens@mq.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2015.2455952

schemes is under the assumption that the authorised users and roles behave in a trusted manner so they do not breach the RBAC policies. However, in a cloud storage system that uses RBAC to control the access to the data, an authorised user of the system may leak the data in the cloud to unauthorised users; or an authorised user may be excluded from accessing the permissions of the role that have been legitimately assigned to the user by a malicious administrator of the system. Such issues rely on trust aspects in these systems.

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 a data owner wants to know if a RBAC system is secure, she or he only needs to determine the trustworthiness of the administrator of the system. However large-scale RBAC systems may have hundred or even thousands of roles and hundreds of thousands of users and permissions. In such cases, 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. In the RBE scheme proposed in the paper [6], the users management can be decentralised to individual roles; that is, the administrators only manage the roles and the relationship among them while the roles 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 the RBE scheme, where each individual role 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, data owners will need to consider the trust of the roles with whom they wish to interact instead of the administrator of the system.

In such a cloud storage system using cryptographic RBAC schemes, it would be helpful if a data owner could determine whether or not a role in the system is trusted before interacting with it in order to prevent malicious roles or users from accessing his or her private data. When the data owner evaluates the trust value of a role, she or he will only proceed with encrypting data to the role if the trust value of the role is above a certain trust threshold (this threshold being set by the data owner). Though the bad behaviour of a role will result in its low trust value, malicious users in the role could also bring down the trust value of the role. Therefore, roles will also need to consider the trust of users so that only users with good behaviour will be granted the role membership if the roles wish to keep their reputations.

A. Contributions of This Paper

The main contributions of this paper are trust models for securing data storage in cloud storage systems that are using cryptographic RBAC schemes. Though there exists many works on trust models in RBAC, none of these works consider the trust for users on the RBAC system itself. The proposed trust models address the missing aspect of trust in cryptographic RBAC schemes to secure data storage in the cloud, and can provide better protection of stored data than using cryptographic approaches alone. The paper proposes trust models to assist (i) the data owners to evaluate the trust on 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 (ii) the roles to evaluate the trust on the users in the RBAC system and use this trust in the decision to grant the membership to a user. We refer to these trust models as Owner-Role RBAC and Role-User RBAC trust models respectively.

These trust models can not only prevent the owners from interacting with roles which have bad historical behaviour in terms of poor track record in carrying out their functions properly, but also assist the roles to identify the malicious users who caused bad impacts on the roles' trustworthiness. This can in turn be used to reduce the risks associated with interacting with the RBAC system for the owners and help roles to keep the RBAC system authentic.

Another important contribution of this paper is that the proposed trust models take into account role inheritance. Since our trust models are for cloud storage systems dealing with hierarchical RBAC schemes, the trustworthiness of a role is also affected by the historical behaviour of its ancestor roles and/or descendent roles (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). Similarly the trustworthiness of a user is also affected by his or her historical behaviour in other roles in the RBAC system. Hence in our trust evaluation, we take into account the impact of role hierarchy and inheritance on the trustworthiness of the roles and users. As far as we are aware, this is the first time such a trust model for RBAC system taking into account role inheritance has been proposed. We also present the architecture of a trust-based cloud storage system which integrates the trust models in a cryptographic RBAC system. Moreover, we describe the relevance of the trust models by considering practical application scenarios and illustrating how the trust evaluations can be used to enhance the quality of decision making by data owners and roles of cloud storage service.

The paper is organised as follows. Section II reviews relevant preliminary knowledge that is needed for the design of our trust models. Section III describes the trust issues in a cryptographic RBAC system and discusses the trust requirements for data owners and roles (and role managers). The formal Owner-Role and Role-User RBAC trust models are presented in Section IV and Section V respectively. The architecture for our secure cloud storage system is presented in Section VI. In Section VII, we illustrate how our trust models can be used in a cloud service application to enhance the quality of security decision making. Section VIII discusses some relevant related works and compares them with our proposed trust models. Section IX concludes the paper.

II. Preliminaries

A. Experience-Based Trust

Trust has played a foundational role in security for a long period of time. Most experience-based trust systems derive the trustworthiness of an entity from both its own

experience and the feedback on the transactions provided by other entities which have had interactions with the entity concerned in the past. Let us consider a simple example of such a system. When a client c finishes a transaction with a service provider p, c gives a feedback as either "positive" or "negative" depending on whether or not c is satisfied with the transaction. The feedback record is of the form $f = (c, p, b, t)$ where b represents the binary value of the feedback and t is the timestamp when the transaction took place. This record f is uploaded by the client to a trust central repository. When another client wants to evaluate the trustworthiness of the service provider p (assuming this client does not have any previous experience with the provider), first it obtains the collection of feedback records $Hist(p) = \{f_1, \ldots, f_n\}$ from the central repository, where n is the total number of the feedbacks about p that have been uploaded to the central repository; $Hist(p)$ represents the feedbacks that all the clients have made to the service provider p. By adding up the total number of each different type of feedback, the client gets an evidence tuple (p, r, s) where r represents the total number of "positive" feedbacks appeared in the collection $Hist(p)$, and s is the total number of "negative" feedbacks in $Hist(p)$. Then the client makes the decision whether or not to continue the transaction with the service provider p based on whether this tuple exceeds a certain threshold; this threshold is dependent on the context of the application at hand.

Many approaches have been proposed that use probabilistic models to evaluate the trust based on the evidence tuple which contains the number of "positive" and "negative" transactions in which the given entity has been involved. Perhaps the most common probabilistic model is the one based on Bayesian trust using a beta probability distribution function [22]–[24]. 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 $0 \leq \theta \leq 1,\ \alpha > 0,\ \beta > 0$. Assume $X = \{x_1, \ldots, x_n\}$ is the collection of the feedback 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:

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

Using Equation 1, the client can derive the probability that the next transaction with the provider will be positive from the transaction history of the provider. Most Bayesian trust systems assume that the parameters $\alpha = \beta = 1$. Some other approaches allows the parameters α and β to be chosen depending on the system context.

B. Role-Based Encryption

A cryptographic RBAC scheme integrates encryption scheme with RBAC model to enforce the access control policies in an untrusted environment. This approach allows data to be encrypted in the way that the ciphertext can only be decrypted by those which are allowed by the access policies. A hierarchical cryptographic access control scheme [11] 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 [12]. Several cryptographic access control approaches have been investigated in [2], [13], and [14] to address the problem of secure data access and cost effective key management in distributed environments. Among the cryptographic RBAC schemes in the literature, role-based encryption (RBE) schemes [6], [8] have achieved many superior characteristics compared to other solutions in terms of efficiency and flexibility.[1]

In this section, we review several concepts in RBE schemes and briefly describe how it works to assist understanding our proposed trust models. We first describe four types of entities which are involved in a RBE scheme.

- SA, the system administrator of the system. It generates the system parameters and issues all the necessary credentials. In addition, this administrator manages the role hierarchy structure for the RBAC system.
- RM is a role manager who manages the user membership of a role. In systems where there are a small number of users, the SA can act as the role manager to manage the user membership of each role to keep the systems compact. However, in large-scale systems, it is almost

[1]There are also attempts of using attribute-based encryption (ABE) schemes to enforce RBAC policies in outsourcing environment. However, those approaches are less flexible in user management compared to RBE schemes, such as that the revocation of a user may result in a key update of all the other users of the same role.

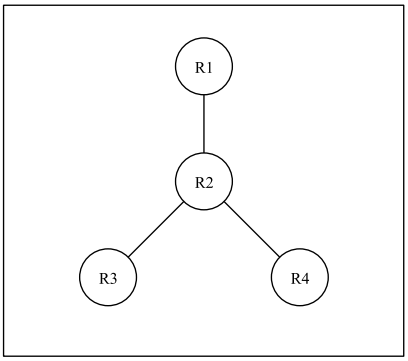


Fig. 1. RBAC Example.

impractical for a single party to manage all the users and permissions. Therefore, having separate role managers can make the user management tasks more flexible and efficient.

- *Users* are the parties who want to access and decrypt the stored data. Each user is required to be authenticated by the SA and issued a credential, which is associated with the identity of the user, upon successful authentication.
- *Owners* are the parties who possess data and want to store the encrypted data in the cloud for other users to access. Owners define role-based access policies to specify who can access their data.

Now we use a general RBAC example to illustrate a RBE scheme and explain how it supports the role inheritance in decryption.

Let us first look at the RBAC example shown in Fig. 1. Four roles are created in a hierarchical structure. The role R_2 inherits from R_3 and R_4, and R_1 inherits from R_2. Assume that all the required algorithms in the RBE scheme have been executed properly to setup the system parameters. We first look at the case where an owner wants to encrypt a message M to the role R_3. The inputs of the RBE encryption are the system public keys pk and the role public parameters pub_{R_3} of R_3, and the output of the algorithm is the ciphertext tuple C.

Assume that the role R_1 has a set of user members $\{U_1, U_2, U_3\}$, and the user U_1 wants to access the message M. Since R_1 inherits from R_2, and hence inherits from R_3, the user U_1 is allowed by the policy to access M. Then U_1 can execute RBE decryption algorithm to recover the message M, and the inputs of the algorithm are pk, the role public parameters pub_{R_1}, the user decryption key dk_{U_1} and the ciphertext C. The algorithm outputs the message M if the decryption key dk_{U_1} that U_1 holds is valid.

Note that users of any role in the set of R_3's ancestor role, $\{R_1, R_2, R_3\}$, can decrypt M, and the users do not need to know the role to which the message was encrypted. Only the role public parameters of the role to which they belong are required in the decryption. From the above briefly described example, we can see how a RBE scheme supports role inheritance in the data decryption.

As mentioned above, security of a RBAC system using a cryptographic approach such as a RBE scheme to protect data privacy is under the assumption that all the entities behave in a trusted manner so they do not breach the RBAC policies, which is not always true in real world systems. For example, in a cloud storage system that uses RBAC to control the access to the data, an authorised user of the system may leak data to unauthorised users; or an authorised user may be excluded by a malicious administrator of the system from accessing the permissions of the role that have been legitimately assigned to the user. Such issues relate to aspects of trust in these systems.

III. Trust Issues in Using Cryptographic RBAC Schemes in Secure Cloud Storage

By 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 the membership to the role or the 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 issues to consider are how the data owners can decide whether or not to trust the role managers in the system and how the role managers can decide whether and how much to trust the users in the system. Owners consider the trust of role managers in order to ensure that their data is secure after being assigned to the roles, and role managers 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 the data owners and role managers of a cryptographic RBAC system.

A. 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, 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 expected by a data owner is to prevent unqualified users 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.

B. 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 leaking is detected. A user who has been involved in the leaking event m times will be considered less trusted than the user who has been involved in the leaking n times if $m > n$.
- *Requirement 2 (The User Should Be Considered as Trusted 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 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.

IV. Owner-Role RBAC Trust Model

In this section, we consider the owner trust models for RBAC systems. We define three entities in our models, namely *Owner*, *User* and *Role*. *Owner* is the entity who owns the data and stores it in an encrypted form in the cloud, and *User* is the entity who wishes to access the data from the cloud. Role is the entity that associates users with the access to owners' data, and each role manages the user membership of itself. When we refer to *Role* in such a context we imply role managers. Then we present an example to illustrate how the behaviour histories of roles in the RBAC system affect the trust of a particular role that a user wants to interact with.

A. Trust Model

Now we give the formal definition of the Owner-Role RBAC Trust Model.

Definition 1 (Interaction): From an owner's perspective, an interaction is a transaction whereby an owner encrypts data to a role, and the role gives the access to the data to qualified users.

A successful interaction is an interaction where only qualified users in the role to which data is encrypted or users in the ancestor roles of the role have accessed the data. An unsuccessful interaction is an interaction where an unqualified user has accessed the data. We define two types of unsuccessful interactions.

User Management Failure: User management failure is an unsuccessful interaction caused by a role who did not manage the user membership properly; that is, the role did not grant the membership to users even when the users have qualified for the role, or the role manager has granted the membership to unqualified users.

User Behaviour Failure: User behaviour failure is an unsuccessful interaction where the data is leaked to unqualified users. When an owner detects that its data has been accessed by unqualified users, the owner may or may not know which qualified user(s) has leaked the data. Here we define *User Behaviour Failure* as such an unsuccessful interaction where the owner does not know which user has leaked the data. If the owner knows who has leaked the data, we consider this unsuccessful interaction as *User Management Failure*.

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

$$\boldsymbol{v} = (r, s_M, s_B)$$

In this trust vector, r is the value related to successful interactions, s_M is the value related to the *User Management Failure* of the role, and s_B is the value related to *User Behaviour Failure*.

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

$$\mathcal{T}(\boldsymbol{v}) = \mathcal{E}(r, s_M + s_B)$$

In order to assist owners to collect feedbacks from other owners, we assume that there exists a central repository in the system to collect the ratings on all the interactions between owners and roles. The feedbacks are available to the owners.

Definition 3 (Interaction History): We define the interaction history derived from these ratings of a role R as

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

Each entry H_i^R in $Hist_{\mathcal{O}}(R)$ is defined as a pair of parameters $H_i^R = \langle ID_i, \boldsymbol{v}_{i,R} \rangle$, where $\boldsymbol{v}_{i,R} = (r, s_M, s_B)$ is a trust vector that represents the trust record of interactions that the owner ID_i has had with the role R. r is the number of ID_i's positive feedbacks on the interactions with R, s_M is the number of negative feedbacks on the interactions with R due to the *User Management Failure*, and s_B is the number of negative feedbacks due to the *User Behaviour Failure*.

Assume that an owner ID_i has assigned a resource to the role R. The central repository will increase r in $\boldsymbol{v}_{i,R} = (r, s_M, s_B)$ by 1. However, if the owner later reports a leak of this resource, the central repository will decrease r by 1 first. Then depending on the failure type, the central repository will increase s_M by 1 if the owner knows who has leaked the resource, or increase s_B by 1 if the owner does not know.

In section III-A, we have 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 following different trust classes.

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

When computing the trust of the role R, an owner obtains the interaction history $Hist_{\mathcal{O}}(R)$ of the role R from the central repository. Assume w_o is the weight that the owner ID_k assigns to the feedbacks from other owners. The individual trust value of the role R is computed as

$$T_{\mathcal{O}}(R)^D = \mathcal{T}(\boldsymbol{v}_{k,R}^D), \quad \boldsymbol{v}_{k,R}^D = \boldsymbol{v}_{k,R} + w_o \sum\nolimits_{i=1, i\neq k}^{n} \boldsymbol{v}_{i,R}$$

The trust vector $\boldsymbol{v}_{k,R}^D$ in this equation is a combination of all the trust vectors in $Hist_{\mathcal{O}}(R)$ with regard to the role R considering the weighting for the ones from other owners.

The weight w_o used in calculating the trust is a positive number indicating the importance of other owners' interaction history in the trust evaluation. Depending on the design of the system, the weight can be chosen from the range [0, 1] where the number 1 means that the trust records from all the owners will be considered as the same important and the number 0 means that other owners' feedback will not be considered in evaluating the trust of a role. Although it is possible to choose a weight $w_o > 1$, it makes little sense to an owner that others' opinions are more important than that of herself or himself.

Inheritance Trust: Inheritance trust is a belief that is derived from the interaction history of other roles that have inheritance relationships with the role.

First we consider the inheritance trust where only the interaction history of the descendant roles is included. When an owner detects a *User Behaviour Failure* with a descendant role R_d of a role R, 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 R, the interaction history from all its descendant roles including R_d needs to be considered.

Assume a role R has m immediate descendant roles $\{R_1, \cdots, R_m\}$, and we define a weight vector $\boldsymbol{w}_{R_i} = (w_{R_i}^R, 0, w_{R_i}^R)$ is the system specified weight between R and R_i. We define the second element of $\boldsymbol{w}_{R_i}$ as zero because the *User Management Failure* is not considered in inheritance trust. We denote the number of users that have been included in the role R_i as n_{R_i}, and the total number of users that have been included in the role R_i and all its ancestor roles as N_{R_i}. Here we assume that the probability that each user violates the trust requirement is the same. The inheritance trust value derived from the descendant roles is computed as

$$T_{\mathcal{O}}(R)^I = \mathcal{T}(\boldsymbol{v}_{k,R}^I),$$
$$\boldsymbol{v}_{k,R}^I = n_R \sum\nolimits_{i=1}^{m} [(\frac{\boldsymbol{v}_{k,R_i}^D}{N_{R_i}} + \frac{\boldsymbol{v}_{k,R_i}^I}{n_{R_i}}), \boldsymbol{w}_{R_i}]$$

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

The weight $\boldsymbol{w}_{R_i}$ in computing the **Inheritance Trust** is a positive number indicating the importance of trust values inherited from other roles. In most cases, the system may use the same weight for all the inheritance relationships. If a system wishes to consider the inheritance relationships of some roles more important than those of other roles,

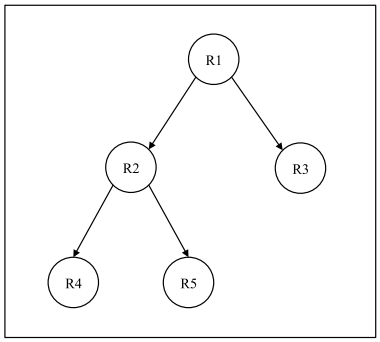


Fig. 2. Hierarchical RBAC Example I.

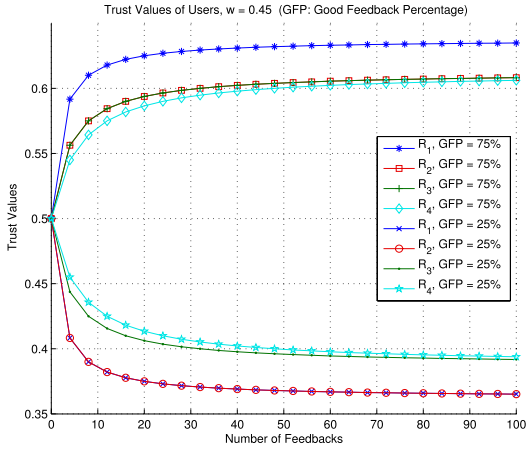


Fig. 3. Trust Values with Feedback on Different Roles.

our trust model can support that by allowing the system to specify different weight for individual inheritance. Similarly to the weight used in the **Individual Trust**, a system can choose the weight $w_{R_i}^R \in [0, 1]$. Choosing the weight 1 means that all the trust values derived from role inheritance will have the same importance. Choosing the weight 0 means that the inheritance will not be considered in the trust evaluation.

Combination Trust: To combine these two types of trusts together, we define a combination trust function for a role R as $T_{\mathcal{O}}(R)$. Assume that $w \in [0, 1]$ is the weight of the inheritance trust. The trust is first computed as follows:

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

In this equation, the smaller the weight w is, the less important the inheritance trust is considered. If a system does not want to consider the inheritance trust, the weight w will be chosen as 0.

Consider the scenario where the combination trust of a role is higher than the trust value 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 value, as the users of its ancestor role have the same level of access as the users in this role. So the combination trust of the role will be the minimum value of the trust of this role and the trust of all its ancestor roles. Assume the role R has m immediate ancestor roles $\{R_1, \cdots, R_m\}$. Then the combination trust is amended to be the following:

$$T_{\mathcal{O}}(R) = min(T_{\mathcal{O}}(R)^C, T_{\mathcal{O}}(R_1), \cdots, T_{\mathcal{O}}(R_m))$$

B. Example

In this section, we look at a few examples of trust evaluation based on feedback from different sources. From these examples, we discuss how the trust value of a particular role is affected by different components in the system.

1) Feedback of Different Roles: First we use an example to show how the owners' trust in a role is affected by the feedbacks for different roles in a RBAC system. In this example, we consider all the bad feedbacks as *User Behaviour 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. 2.

In Fig. 2, the role R_1 inherits from role R_2 and role R_3, and the role R_2 inherits from R_4 and R_5. For simplicity, let us assume that the number of users in all these roles are the same. We set the weight between every two roles and the weight of other owners' feedbacks to 1; that is, the weight vector for each role is defined as $\boldsymbol{w}_{R_{i+1}}^{R_i} = (1, 0, 1), i \in [1, 5)$, and $w_o = (1, 1, 1)$. When an owner wants to encrypt data to the role R_2, she or he will need to evaluate the trust value of R_2 to decide whether it is safe to give access to the data to R_2. In Fig. 3, we show the trust values of R_2 when only different individual roles in the RBAC system have feedbacks. 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 feedbacks, and 75% of these feedbacks are positive.

When the good feedbacks percentage is 75%, the trust value for R_2 goes up with increasing number of feedbacks. When the feedbacks are only given for R_1, the increase in the trust value is the fastest. This is because all the feedbacks are used in the calculation of the individual trust of R_1, and the combination trust of R_1 is not affected by other roles as there is no feedback for others. Since R_2 has neither individual trust nor inheritance trust, its combination trust is the minimal value among the set which contains the trust value of R_1 only. Therefore, it has the highest value as it is calculated based on all the feedbacks in which the positive ones are in the majority. When the feedbacks are only provided for R_4, the increase in the trust value is the slowest. This is because the feedbacks used in the calculation of the inheritance trust of R_2 has been averaged over three roles, R_1, R_2, and R_4, and only 1/3 of the feedbacks are considered in calculating the trust value of R_2. We see that the trust value of R_2 increases slightly faster when the feedbacks are only provided for R_3. This is because the population in two roles is less than that in three roles, and 1/2 of the feedbacks are considered in the calculation of R_2's trust value. This is analogous to the case where R_2 has more feedbacks where the positive ones are in the majority. When the feedbacks are only provided for R_2, we see that the curve overlaps with the trust value where the feedbacks are only for R_3. Since R_2 and R_3 have the same population in this example, the feedbacks for R_2

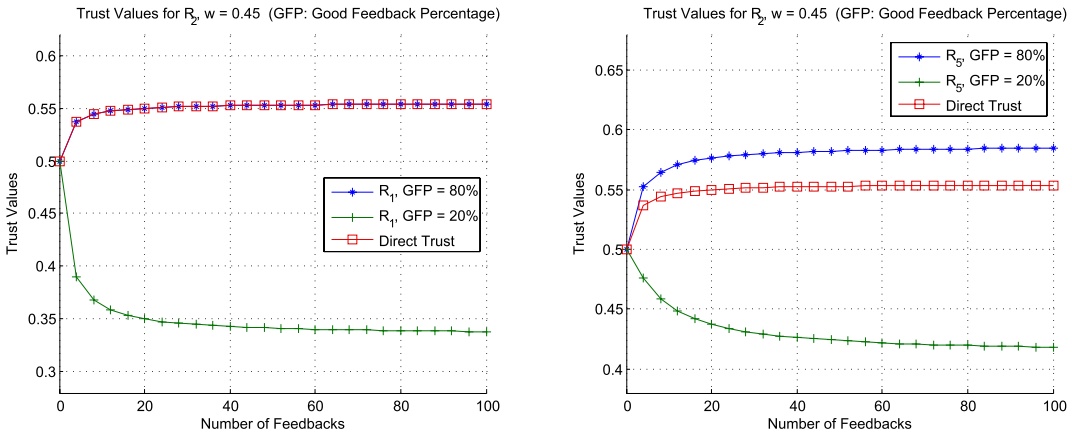


Fig. 4. Trust Values for R_2 with Role Inheritance. (a) Feedback on R_1. (b) Feedback on R_5.

and those for R_3 have the same impact on the trust value of R_1. When not considering R_1, R_2 has a higher trust value as the value is based on all the feedbacks. However, taking the lowest trust value among R_2 and all its ancestor roles makes R_2's trust value the same as the one when feedbacks are only for R_3.

2) Role Inheritance: When the good feedbacks percentage is 25%, the trust value for R_2 goes down with the increasing number of feedbacks. When the feedbacks were only provided for R_1, the decrease in the trust value is the fastest. Similar to the above, this is because all the feedbacks have been used in the calculation of R_1's trust value which in turn becomes the trust value of R_2, and the majority of these feedbacks are negative. When the feedbacks are only provided for R_4, the decrease in the trust value is the slowest. This is because the feedbacks used in the calculation of the inheritance trust has been averaged by three roles, R_1, R_2, and R_4. We also see that the trust value decreases slightly faster when the feedbacks are only for R_3 because more feedbacks whose majority are negative are used in calculating R_2's trust value. When the feedbacks are only provided for R_2, we see that the curve overlaps with the trust value when feedbacks are only for R_1 instead of R_3. Since the feedbacks for R_2 and those for R_3 have the same impact on the trust value of R_1, the trust value of R_1 is the same as when the feedbacks are for R_3. However, the unamended combination value of R_2 is the same as R_1's trust value (the same as the trust value of R_2) when the feedbacks are only for R_1 as they both are calculated based on the same amount of feedbacks. This trust value is lower than that of the role R_1, and is used as the amended trust value of R_2.

From Fig. 3, we see that the feedbacks for different roles in the system have different impact on the trust value of R_2. Firstly, the feedbacks on ancestor roles have the most significant impacts on the trust of a role. Secondly, the more users have access to a role's data, the less impact the feedbacks for the role will have on each role that the users belong to.

Next we use an example to illustrate how the owners' trust in a role is affected by the inheritance relationship in a RBAC system. In this example, we use the same role hierarchy example shown in Fig. 2. We assume that each role has the same amount of *User Behaviour Failure* as the *User Management Failure* and each of R_1, R_2, R_5 has 100 users. We set the GFP of R_2 to 60%.

In Fig. 4a we show the trust value for R_2 when R_1 has different feedback from owners. For example, the curve for $R_1, GFP = 80\%$ shows the trust values of R_2 when R_1 has 80% positive feedback, while R_2 having 60% positive feedback. The curve for $Direct\ Trust$ shows the trust values for R_2 when the trust records of R_1 is not considered when evaluating the trust of R_2.

When the good feedback percentage of R_1 is 80%, the trust values for R_2 remain the same, whether the feedback of R_1 is considered or not. The result shows that the feedback of ancestor roles will not have impact on the trust of the role if the ancestor roles are considered more trusted. However, when the good feedback percentage of R_1 is 20%, the trust values of R_2 drop rapidly less than that for $Direct\ Trust$. This is because the users of R_1 have the same level of access to the data assigned to R_2 as the users in R_2, and a malicious user in R_1 can leak the data assigned to R_2. Therefore the owners will trust R_2 at the same level as R_1 if R_1 is less trusted than R_2. This would not be captured if the role inheritance is not considered.

Then we look at the impact of R_5's feedback on the trust of R_2. In Fig. 4b we show the trust value for R_2 when R_5 has different feedback from owners. For example, the curve for $R_5, GFP = 80\%$ shows the trust values of R_2 when R_5 has 80% positive feedback, while R_2 having 60% positive feedback. The curve for $Direct\ Trust$ still shows the trust values for R_2 when the trust records of R_5 is not considered when evaluating the trust of R_2.

When the good feedback percentage of R_5 is 80%, the trust values for R_2 are slightly more than that of $Direct\ Trust$. This is because the *User Behaviour Failure* of R_5 is related to R_2; the more trusted R_2 is, the less *User Behaviour Failure* R_4 and R_5 will have. So higher GFP of R_5, that is less *User Behaviour Failure*, implicitly indicates that R_2 is more trusted. Therefore the trust values for R_2 increase slightly when the

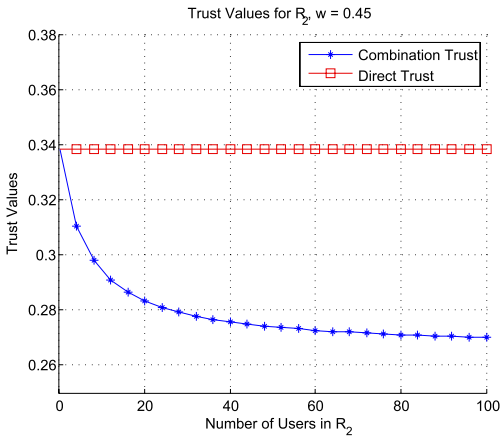


Fig. 5. Trust Values for Different Number of Users.

feedback of R_5 is taken into account when evaluating the trust of R_2. When the good feedback percentage of R_5 is 20%, the trust values for R_2 decrease gradually as the number of feedback increases. The result shows that the *User Behaviour Failure* of R_5 results in suspicion of data leakage on users in R_2, and hence cause low trust values of R_2. But the inheritance trust has limited impact on the trust of R_2 as that is only a portion of the trust evaluation which is controlled by the weight w. In this example, the weight w is set to 0.45. This is different from the example with feedback on R_1 where the trust value for R_2 is dominated by the trust records of R_1 when GFP of R_1 is lower than that of R_2.

In the examples of role inheritance, we see the importance of using the trust model that considering the role inheritance relationship when evaluating the trust. Suppose the trust threshold of the system is set to 0.5, without considering the trust records of other roles, the role R_2 would be evaluated as trusted as its $Direct\ Trust$ is always above the threshold. However, when R_1 or R_5 are untrusted as shown in the examples, still evaluating R_2 as trusted will put owners in risk when they want to assigned data to the role R_2. The result of the examples shows that this problem can be addressed by rectifying the trust values for R_2 using our trust model.

3) Number of Users: Last we use an example to show how the owners' trust in a role is affected by the number of users of the role. We use the same role hierarchy example shown in Fig. 2. We assume that R_1 does not have any feedback and R_2, R_5 has the same amount of *User Behaviour Failure* as the *User Management Failure*. We set the GFP of R_2 to 60% and each of R_1, R_5 has 100 users.

In Fig. 5 we show the trust value for R_2 when the number of users in R_2 changes. The curve for $Combination\ Trust$ shows the trust values of R_2 when inheritance trust is taken into account. The curve for $Direct\ Trust$ shows the trust values for R_2 when only the trust records of R_2 is used in evaluating the trust of R_2, and it is used for comparison purpose in this example.

The result shows that the more users that R_2 has, the more impact it will have from the inheritance trust. This is because the trust model assumes that all the users will have equal suspicion when the system could not track the user who has caused the leak of data. Then the suspicion of each user will be aggregated for each role to reflect the trustworthiness of the role as a whole. Therefore, the more users in R_2 implies the more suspicion that R_2 has when there are *User Behaviour Failure* in its descendant roles.

For all the above examples, the results show that our owners' trust model is useful in assisting owners to determine properly the trust of roles in RBAC systems.

V. 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. In this section, we present a trust model for roles' trust in users as an extension of the owners' trust model on roles. This trust model aims to assist a role to determine the trust of users who belong to the role or want to join the role.

Roles can use this model to periodically check 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.

A. Trust Model

In this subsection, we give the formal definition of the roles' trust in users.

Definition 4 (Trust Vector): Since there is no interaction between the roles and users, we define a trust vector to represent directly the behaviour history of a user as follows:

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

In the trust vector, h is the total number of resources that have been assigned to the role, and s is the value related to the leaking events that the user was involved in. Recall that we say that a user is involved in a leaking event of a resource if the resource has been found to be leaked, and this user has accessed the resource before the leaking was detected.

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

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

Definition 5 (Trust Records): We assume that there exists a central repository in the system that collects and stores the behaviour histories of users provided by roles of which the user was a member. We define the trust record provided by a set $\mathcal{R}$ of n roles as

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

Each entry H_i^U in $Hist(U)$ is defined as a pair of parameters, $H_i^U = \langle R_i, v_{i,U} \rangle$ where $\boldsymbol{v}_{i,U} = (h, s)$ is a trust vector that represents the trust record of the user U when she or he is the member of the role R_i. h is the number of resources

that have been assigned to R_i when U is the member of the role R_i, and s is the number of leaks related to R_i that U was involved in when U is the member of R_i.

We assume that a role R_k currently has a set $\mathcal{U}$ of n users. When a new user joins, R_k will create a trust vector for the user in the central repository. The vector is initialised as $(h, 0)$ where h is the current number of resources that have been assigned to the role.

When a new resource is assigned to R_k, R_k will update the trust records $(H_k^{U_1}, H_k^{U_2}, \cdots, H_k^{U_n})$ in the central repository by increasing h in each vector by 1. When a leak is detected by or reported to the role R_k, R_k tracks the set of users who have accessed the leaked resource, and increases the value s in trust vectors of this set of users by 1.

Since a role can inherit from another role in RBAC systems, besides updating the trust records maintained by R_k, R_k will need to notify all its ancestor roles about the leak, and all the ancestor roles of R_k will update their trust records for their users in the same way that R_k does.

Next we define the trust function used in the model.

Direct Trust: Direct trust of a role on a user U is the belief that is derived directly from trust records of the user U from the role itself.

When a role R_k wishes to evaluate the trust value of a user U, it first obtains the trust record $Hist_{\mathcal{R}}(U)$ of the user from the central repository. Taking as input the trust record H_k^U maintained by R_k itself, the direct trust value can be computed as follows:

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

Recommended Trust: Recommended trust is the belief that is derived from the trust records of the user from other roles in the system.

Assume there are n roles $\{R_1, \cdots, R_n\}$ who have provided the trust records for the user. The recommended trust value of the user from the perspective of the role R_k is computed as

$$T_{\mathcal{R}}(U)^R = \mathcal{T}(\boldsymbol{v}_{k,U}^R), \quad \boldsymbol{v}_{k,U}^R = \sum\nolimits_{i=1, i\neq k}^{n} \boldsymbol{v}_{i,U}$$

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

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

In this equation, the smaller the weight w is, the less important the recommended trust is considered. If a system does not want to consider the recommended trust, the weight w will be chosen as 0.

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

B. Example

In this section, we look at a few examples of trust evaluation based on feedback from different sources. From these examples, we discuss how the trust value of a particular role is affected by different components in the system.

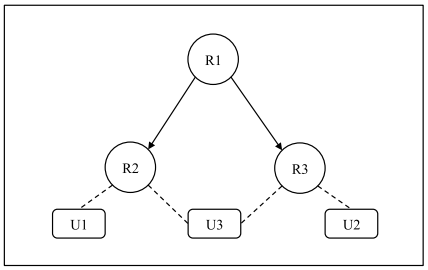


Fig. 6. Hierarchical RBAC Example II.

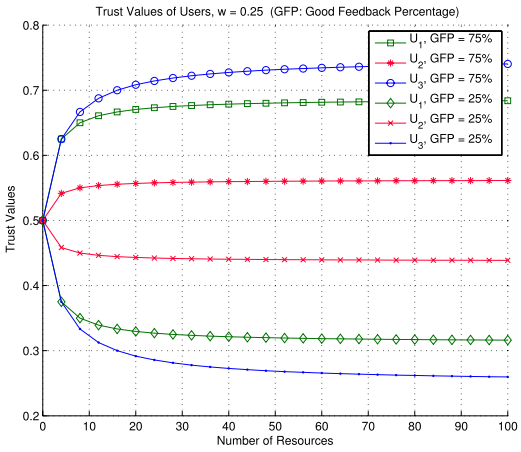


Fig. 7. Trust Values for Users Evaluated by R_2.

1) Trust in Different Users: First let us consider an example to illustrate how a role's trust in users is related to role(s) that users belong to in a RBAC system. In this example, we assume that we do not know whether these users are the causes of the bad feedbacks. Consider the role hierarchy example shown in Fig. 6.

In Fig. 6, the role R_1 inherits from role R_2 and role R_3, the users U_1 and U_2 are the members of the role R_2 and R_3 respectively, and the user U_3 is the member of both the role R_2 and R_3. For simplicity, let us assume that the number of resources assigned to all the roles are the same, and users' feedbacks are from the role(s) to which they belong. For example, U_2's feedback is provided only by R_3 while U_3 is getting feedbacks from both R_2 and R_3. In Fig. 7, we show the trust values for these users U_1, U_2, and U_3 evaluated from the role R_2's perspective.

When the users' good feedbacks percentage is 75%, the trust values for the users are increasing. The increase in the trust value for U_2 is slowest because the weight of feedbacks from other roles is low where $w = 0.25$. The trust value for U_1 increases faster as the weight for the direct trust in this example is more than that of the recommended trust. The trust value for U_3 increases the fastest as this user receives good feedbacks from both roles R_2 and R_3 and the combined value is higher than any one of them. When the users' good feedbacks percentage is 25%, the trust values for the users are decreasing. The decrease in the trust value for U_2 is slowest because the weight for feedbacks from other roles is low. The trust value for U_1 decreases faster as the weight for the direct

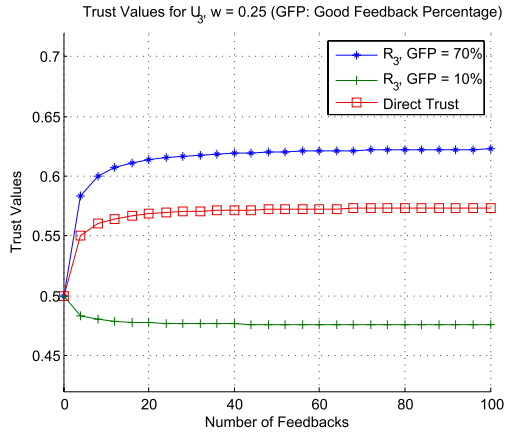


Fig. 8. Trust Values for U_3 Evaluated by R_2.

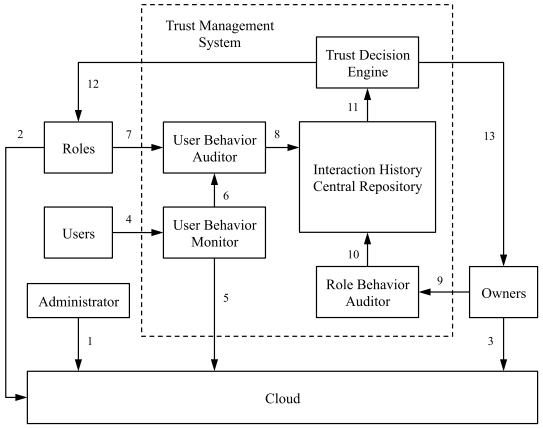


Fig. 9. Architecture for Using Owners' Trust Models in a Cryptographic RBAC System.

trust is more than that for the recommended trust. The trust value for U_3 decreases the fastest as this user receives bad feedbacks from both roles R_2 and R_3.

From Fig. 7, we see that the feedbacks for users in different roles result in a different trust value when R_2 evaluates the trust of these users. When a user has good trust records in other roles only, the role will trust the user less than another user who has the same trust record in the role itself. A user who has good trust records in both this role and other roles will be trusted the most among the three users.

2) Combination Trust: Then we use an example to show how the roles' trust in a user is affected by the feedback from other roles in the system. We use the same role hierarchy example shown in Fig. 6. We set the GFP of R_2 to 60%.

In Fig. 8 we show the trust value for U_3 evaluated by R_2 when the GFP from R_3 on U_3 changes. For example, the curve for R_3, $GFP = 70\%$ shows the trust values of U_3 when the GFP from R_3 is 70% and the GFP from R_2 is 60%. The curve for $Direct\ Trust$ shows the trust values for U_3 when only the trust records from R_2 is used in evaluating the trust of U_3, and it is used for comparison purpose in this example.

As shown in Fig. 8, when the GFP for the feedback from R_3 is higher than that from R_2, that is R_3 has more positive feedback on U_3 than R_2 has, the trust values for U_3 are higher than that evaluated using the feedback from R_2 alone. When the GFP from R_3 is 10%, the trust values for U_3 are less than that for $Direct\ Trust$. Assume that the system has set the threshold to 0.5, without taking into account the feedback from R_3, R_2 would consider U_3 as trusted. However, by using feedback from both R_2 and R_3 in evaluating the trust of U_3, U_3 will be rated as untrusted.

These results show that our roles' trust model is intuitive. Hence it is useful in assisting roles to determine properly the trust of users in RBAC systems.

VI. Architecture

In this section, we present the design of a secure cloud storage system combining the trust models for RBAC proposed in section IV and V with a cryptographic RBAC system. This architecture provides a practical solution in building a reliable and trusted RBAC system while retaining the use of cryptographic techniques. We have developed a prototype implementation of this architecture and used it in carrying out our experimental analysis.

A. System Overview

Consider the system architecture shown in Fig. 9. Since our trust models are based on cryptographic RBAC schemes, our system contains all the entities that a cryptographic RBAC scheme has, which include an administrator, roles, users, and owners. The administrator is the certificate authority of the RBAC system. The administrator generates the system parameters and issues all the necessary credentials. In addition, the administrator manages the role hierarchical structure of the system. To put a role into the role hierarchical structure, the administrator needs to compute the parameters for the 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 defines 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. Owners specify who can access the data in terms of role-based policies. In the RBAC model, they are the parties who manage the relationship between permissions and roles. An owner can be a user within the organisation or an external party who wants to send data to users in the organisation. In this architecture, we consider an owner to be a logically separate component even though a user can be an owner and vice versa. 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-enhanced cryptographic RBAC system integrates an extra trust management system, which consists of five components. Next, we describe the details of these components.

1) 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 records of all these interaction histories and trust records which are used by the Trust Decision Engine (described as below) in evaluating the trust value of roles and users. Any entity that is residing outside the trust management system is not able to access the central repository.

2) Role Behaviour Auditor: In order to protect the integrity of the feedbacks on roles, a role behaviour auditor collects the feedbacks for roles from owners. The role behaviour auditor needs to ensure that an owner who uploads feedback is authorised. All the valid feedbacks will be forwarded to the central repository, and invalid feedbacks will be discarded. Besides the feedbacks from owners, the role behaviour auditor also collects information about data assignment to roles. Owners need to inform the role behaviour auditor when they encrypt data to roles. The auditor will update the central repository with the number of resources that have been assigned to the roles.

3) User Behaviour Auditor: A user behaviour auditor is an entity to collect the feedbacks on users' behaviour. However, unlike the role behaviour auditor, the user behaviour auditor listens on two channels for feedbacks. One is from the roles who may report the leakage of data, and another is from the user behaviour monitor which reports the access histories of users to the stored data in the cloud. This auditor will determine whether a user is involved in the leakage of data, and update the user trust records in the central repository if the user has accessed the leaked data.

4) User Behaviour Monitor: A user behaviour monitor acts as a proxy server between users and the cloud. It only monitors and forwards the users' requests to access stored data in cloud. When a user wants to access a resource, she or he does not send the request to the cloud directly. Instead, the request is sent to the user behaviour monitor, and the user behaviour monitor will forward the request to the cloud. The monitor will inform the user behaviour auditor the information about which user has accessed which resources.

5) Trust Decision Engine: The trust decision engine is the entity which evaluates the trust of roles for owners and the trust of users 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 user.

B. System Workflow

All the entities in the system are connected through different communication channels which are labelled with numbers in Fig. 9. 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 channel 1. Roles grant the membership to users, and upload role parameters to the cloud via channel 2. Owners encrypt and upload data to the cloud via channel 3. When a user wants to access a resource stored in the cloud, she or he first sends the access request to the user behaviour monitor via channel 4, and the user behaviour monitor forwards the request to the cloud through channel 5. The cloud then communicates with the user as in a normal cryptographic RBAC scheme. The monitor also sends the user behaviour auditor the information about the user identity and the resource identity via channel 6.

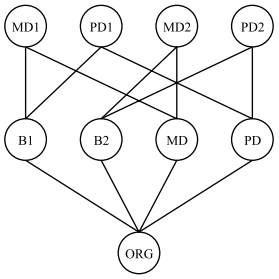
When an owner wants to encrypt a resource to a role in the RBAC system, she or he requests for the trust evaluation on the role to the trust management system. Then the trust value of the role will be returned to the owner through the channel 13. If the owner believes that the role is trusted, she or he then encrypts and uploads the resource to the cloud via channel 3. The owner also notifies the role behaviour auditor about the identity of the resource in the cloud and the role to whom the resource is encrypted. The auditor then updates the number of the resources that have been assigned to this role in the central repository via channel 10. When an owner has found a leak in her or his resource to unauthorised users, she or he then provides feedback on the role to whom the resource is encrypted to the role behaviour auditor through channel 9. Once the role behaviour auditor verifies that the feedback is from an authorised owner, it will forward the feedback to the central repository.

When a negative feedback of a role has been raised by an owner because of the leak of a resource, the role will send the identity of the resource to the user behaviour auditor via channel 7. The auditor then updates the trust records of users, who have accessed this resource, in the central repository via channel 8. The role can ask the trust management system about the trust evaluation for a user at any time, and the trust value will be returned by the trust decision engine through channel 12. Upon receiving the trust values for users from the trust decision engine, a role can update the role parameters that represent the user membership in the cloud via channel 2 if there exist malicious users whose role memberships need to be revoked.

VII. Application Scenario

Finally, we consider an application scenario based on a digital library system to illustrate how our proposed trust models can enhance the quality of decision making. Assume that the digital library system uses an external cloud storage platform to store all the resources, and publishers are allowed to share their digital resources such as books, magazines, and other types of publications on this platform. 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 each branch consists of several departments. When employees 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



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. 10. Digital Library System Example.

uses a RBAC system to control the access to resources, and the role hierarchy is shown in Fig. 10.

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. 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 them. 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 feedback; 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. Without using the trust model, the role ORG may be considered as trusted as it has no negative feedback. However, if the role ORG is allowed to subscribed to a resource, the resource owner has the risk that the users in B_2 may leak the resource to unauthorised users.

Again we assume that the role $MD1$ has only good feedback; that is, the resources the role $MD1$ has subscribed have never been leaked. Consider the extreme case where all the feedback that the role B_1 has is *User Behaviour Failure* which means that the system could not determine who has caused the leakage of the resources that role B_1 has subscribed. Without using the trust model, the role $MD1$ may be rated as trusted as it has no negative feedback. However, because the system does not know which user(s) leaked the resources B_1 has subscribed, it can be the user(s) in the role B_1, or it can be the user(s) in the role $MD1$ as all the users in $MD1$ have access to the leaked resources. There is a risk if the malicious user is in $MD1$, allowing $MD1$ to subscribe additional resources may cause the leakage of more data.

On the other hand, when the role B_2 realises that its trust value is low, it may decide to warn or exclude potential malicious users. Then our roles' 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 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.

VIII. Related Works

There have only been some related works which have addressed only trust on users in RBAC systems. Chakraborty and Ray [25] 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. In [26], a trust model for RBAC was introduced which evaluates the trust in users based on user behaviours and context, in a context-aware access control model. Another trust model was discussed in [27] which also uses trust level to determine the access privileges of users. All these trust models only consider the trust of users in a RBAC system. None of these works address the trust for data owners on the RBAC system itself thereby determining the trust of the roles in the RBAC system with which they want to interact. The trust for data owners is critical in cloud storage systems which has been addressed in this paper.

As an extension of the owners' RBAC trust model, our trust models have also addressed the roles' trust on users. The existing works [25], [27] control the access privileges of a user depending on his or her trust level. The differences

between our model and the existing ones is that our roles' trust model works in the RBAC systems which use cryptographic RBAC schemes. That is, our models take into account cryptographic operations and the access privilege to decrypt the data stored in the cloud, which none of the existing works address.

IX. Conclusion

In this paper, we have addressed trust issues in cryptographic RBAC systems for securing data storage in a cloud environment. The paper has proposed trust models for owners and roles in RBAC systems which are using cryptographic RBAC schemes to secure stored data. These trust models assist owners and roles to create flexible access policies, and cryptographic RBAC schemes ensure that these policies are enforced in the cloud. The trust models enable the owners and roles to determine the trustworthiness of individual roles and users in the RBAC system respectively. They allow the data owners to use the trust evaluation to decide whether or not to store their encrypted data in the cloud for a particular role. The models also enable the role managers to use the trust evaluation in their decision to grant the membership to a particular user. Another significant contribution of this paper is that the proposed trust models take into account role inheritance and hierarchy in the evaluation of trustworthiness of roles. As far as we are aware, this is the first time such a trust model for role-based access control system taking into account role inheritance has been proposed. We designed the architecture of a trust-based cloud storage system which has shown how the trust models can be integrated into a system that uses cryptographic RBAC schemes. We have also described the application of the 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 decision making by data owners and role managers of the cloud storage service.

References

[1] D. F. Ferraiolo and D. R. Kuhn, "Role-based access controls," in *Proc. 15th NIST-NCSC Nat. Comput. Secur. Conf.*, Oct. 1992, pp. 554–563.

[2] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "A data outsourcing architecture combining cryptography and access control," in *Proc. CSAW*, Nov. 2007, pp. 63–69.

[3] S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proc. VLDB*, Sep. 2007, pp. 123–134.

[4] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[5] Y. Zhu, H.-X. Hu, G.-J. Ahn, H.-X. Wang, and S.-B. Wang, "Provably secure role-based encryption with revocation mechanism," *J. Comput. Sci. Technol.*, vol. 26, no. 4, pp. 697–710, Jul. 2011.

[6] L. Zhou, V. Varadharajan, and M. Hitchens, "Enforcing role-based access control for secure data storage in the cloud," *Comput. J.*, vol. 54, no. 10, pp. 1675–1687, Oct. 2011.

[7] L. Zhou, V. Varadharajan, and M. Hitchens, "Integrating trust with cryptographic role-based access control for secure cloud data storage," in *Proc. IEEE TrustCom*, Jul. 2013, pp. 560–569.

[8] L. Zhou, V. Varadharajan, and M. Hitchens, "Achieving secure role-based access control on encrypted data in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp. 1947–1960, Dec. 2013.

[9] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Comput.*, vol. 29, no. 2, pp. 38–47, Feb. 1996.

[10] R. Sandhu, D. Ferraiolo, and D. Kuhn, "The NIST model for role-based access control: Towards a unified standard," in *Proc. RBAC*, 2000, pp. 47–63.

[11] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 239–248, Aug. 1983.

[12] G. Miklau and D. Suciu, "Controlling access to published data using cryptography," in *Proc. VLDB*, 2003, pp. 898–909.

[13] M. J. Atallah, K. B. Frikken, and M. Blanton, "Dynamic and efficient key management for access hierarchies," in *Proc. CCS*, Nov. 2005, pp. 190–202.

[14] H. R. Hassen, A. Bouabdallah, H. Bettahar, and Y. Challal, "Key management for content access control in a hierarchy," *Comput. Netw.*, vol. 51, no. 11, pp. 3197–3219, Aug. 2007.

[15] *Trusted Computer System Evaluation Criteria*, Dept. Defense, Alexandria, VA, USA, Dec. 1985.

[16] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proc. IEEE Symp. Secur. Privacy*, May 1996, pp. 164–173.

[17] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis, "The keynote trust-management system version 2," Internet Soc., Network Working Group, Tech. Rep. RFC2704, 1999.

[18] R. Yahalom, B. Klein, and T. Beth, "Trust relationships in secure systems—A distributed authentication perspective," in *Proc. IEEE Symp. Res. Secur. Privacy*, May 1993, pp. 150–164.

[19] B. Yu and M. P. Singh, "A social mechanism of reputation management in electronic communities," in *Proc. CIA Workshop Cooperat. Inf. Agents*, 2000, pp. 154–165.

[20] A. Jøsang and S. L. Presti, "Analysing the relationship between risk and trust," in *Proc. iTrust*, 2004, pp. 135–145.

[21] A. Jøsang, "A logic for uncertain probabilities," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 9, no. 3, pp. 279–311, Jun. 2001.

[22] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt, "Ratings in distributed systems: A Bayesian approach," in *Proc. Workshop Inf. Technol. Syst.*, 2001, pp. 1–7.

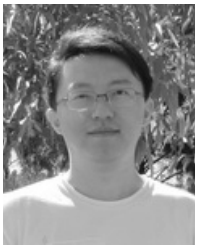
[23] A. Jøsang and R. Ismail, "The beta reputation system," in *Proc. 15th Bled Conf. Electron. Commerce*, 2002, pp. 2502–2511.

[24] L. Mui, M. Mohtashemi, and A. Halberstadt, "A computational model of trust and reputation for E-businesses," in *Proc. HICSS*, 2002, p. 188.

[25] S. Chakraborty and I. Ray, "TrustBAC: Integrating trust relationships into the RBAC model for access control in open systems," in *Proc. SACMAT*, Jun. 2006, pp. 49–58.

[26] F. Feng, C. Lin, D. Peng, and J. Li, "A trust and context based access control model for distributed systems," in *Proc. HPCC*, Sep. 2008, pp. 629–634.

[27] M. Toahchoodee, R. Abdunabi, I. Ray, and I. Ray, "A trust-based access control model for pervasive computing applications," in *Proc. DBSec*, vol. 5645., Jul. 2009, pp. 307–314.



Lan Zhou received the Ph.D. degree from Macquarie University, and the M.Sc. degree in computer science from the University of Wollongong. He is currently an Honorary Associate with the Department of Computing, Macquarie University. His research interests include role-based access control, secure cloud storage, trust management systems, secure credential systems, and key agreement protocols.

Vijay Varadharajan is currently a Microsoft Chair Professor of Innovation in Computing with Macquarie University. He is also the Director of Advanced Cyber Security Research Centre. He has authored over 380 papers in international journals and conferences, has coauthored and edited ten books on security, networks and distributed systems, and holds three patents. His current areas of research interest include secure distributed systems, trusted computing, Internet security, cloud computing, and mobile and wireless security. He is a fellow of the British Computer Society, the IEE, U.K. (FIEE), the Institute of Mathematics and Applications, U.K. (FIMA), the Australian Institute of Engineers (FIEAust), and the Australian Computer Society (FACS). He has been on the Editorial Board of several journals, including the IEEE TRANSACTIONS IN DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and the *ACM Transactions in Information Systems Security*.

Michael Hitchens received the bachelor's (Hons.) degree in mathematics and the Ph.D. degree in computer science. He is currently the Associate Dean of Quality and Standards with the Faculty of Science, Macquarie University. He is part of the Department of Computing, where he is a member of the Advanced Cyber Security Research Centre and the Virtual and Interactive Simulations of Reality Research Group. His research interests include access control, security protocols, trust in distributed systems, and game design.