A New RBAC Based Access Control Model for Cloud Computing

Zhuo Tang[1], Juan Wei[1], Ahmed Sallam[1], Kenli Li[1], and Ruixuan Li[2]

[1] College of Information Science and Engineering, Hunan University,
Changsha 410082, China
hust_tz@126.com
[2] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China

Abstract. Access Control is an important component of Cloud Computing; specially, User access control management; however, Access Control in Cloud environment is different from traditional access environment and using general access control model can't cover all entities within Cloud Computing, noting that Cloud environment includes different entities such as data owner, end user, and service provider. In this paper, we propose a new access control based on Role-based access control (RBAC) model. This model includes two kind of roles, user role (UR) and owner role (OR); such that, Users get credential from owners to communicate with service provider and to get access permissions of resources. We also discuss the aspects of user access control management, such as authentication, privilege management, and deprovisioning. Moreover, we use administrative scope to update hierarchy when there is a role added or revoked to simplify the user access control management. By applying the model in Cloud environment the results shows that it can reduce the security problems to two classes in the RT [$\longleftarrow$, $\cap$] role-based trust-management language with a test-paper system.

Keywords: Cloud computing, access control, user access control management, security analysis.

1 Introduction

Over the passed few decades, many access control models have been proposed to specify the different access control policies. These models primarily include DAC, MAC and RBAC. In DAC [1] the creator controls all the objects and the object owner has rights to give the access permissions to others; however, it is difficult to control the permission hierarchy. MAC [2] on the other side cannot implement distributed management. A better alternative is the RBAC [3] that is a widely used access control mechanism, which associates permissions and roles; such that, users get the corresponding permissions by playing roles in RBAC.

In the RBAC family which was proposed by Sandhu in 1996 [7], users' privileges are attached to their roles, where the users can acquire when needed. A role is a

R. Li, J. Cao, and J. Bourgeois (Eds.): GPC 2012, LNCS 7296, pp. 279–288, 2012.

permission set for a special work station. When the users' privileges need to be changed, we have to revoke the user role or re-distributing the roles.

James B.D. Joshi et al. [4, 5] extended RBAC theory. In many practical scenarios, users may be restricted to assumed roles only at predefined time periods. GTRBAC allow expressing the role hierarchies and separating of duty (SoD) constraints [6] for specifying fine-grained temporal semantics.

Li et al. [8] proposed the notion of security analysis and studied security analysis in the context of RT[←,∩], a role-based trust-management language. They showed that a security analysis instance in RT[←,∩] involving only semi-static queries can be solved efficiently. The work by Koch [9] analyzes the safety in RBAC with the RBAC state and state-change rules posed as graph formalism.

Li and Tripunitara [10] performed security analysis on two restricted versions of administrative RBAC. These are known as AATU and AAR. They proposed two reduction algorithms and studied complexity results for various analysis problems such as safety, availability and containment.

Jason Crampton et al. [11] and Koch et Al. [12] have introduced the administrative scope in a role hierarchy and develop a family of models for role hierarchy administration. The administrative role could update the role hierarchy dynamically when some roles added or deleted. Youngmin Jung Et al. [13] have proposed an adaptive security model for Cloud Computing environment. The model based on the improved RBAC model and adapts a role switching model. Weichao Wang et al [14] have discussed the outsourced data security. But this paper only considered the data update, and they did not consider the entities update.

2 CARBAC: The RBAC for Cloud Computing Environment

2.1 Models and Definitions

RBAC is a very useful access control model, despite its benefits it is fundamentally limited by its subject-centric nature. The CARBAC model proposed in this section addresses these situations by incorporating support for environmental state and object state in access control policies. Because the resources stored in Cloud Computing servers are very huge, we have to divide resources to data blocks. Our model focuses on the access control in the SaaS delivery model [15] where applications are delivered as a service to end users. In the following we list a set of important definitions.

User Role (UR): A user role in our model is similar to RBAC role. The user is an individual within an organization gets his/her access permission through the role assigned to him.

Owner Role (OR): It is the set of permissions that data owner has to give credential to users and update Cloud resources. Data owner processing users' request through OR.

Request: User's access request is denoted as Req=(UR,OR, request index, data block,op), where "request index" is to label user's access request., "data block" is to label the data block the user requests. And "op" is the operation that the user will apply on resources.

Role hierarchies are almost inevitably included whenever roles are discussed in the literature [16, 17]. In the following, we define role hierarchies and describe their semantics.

We define a partial order set $< R, \leq >$ and R is the set of roles. We denote x covers y as $y \leq x$, and if there is an edge between the role x and the role y, x is called the parent role of y. And the role x can inherit the permission of role y .

We use the administrative scope definition introduced in Jason Crampton et al. [12]. When a role is added to or deleted from the hierarchy, the administrative role will dynamically change the administrative scope. The model can be seen from figure1. UR and OR can update the role hierarchy by themselves. User role hierarchy and owner role hierarchy has its administrator. The administrator of owner role hierarchy administrates the relationship between OR and UR while OR has the responsibility to manage users' information and the credential of user. Roles assigned to user with the following constraints:

1) Only one role can be assigned to one user in one session, and one user cannot be signed to two roles at the same time.

2) Every user has a tag to note the role he/she assigned to.

2.2 User Access Control Management

This paper discusses the user access control management in universal cloud computing environment which includes authentication, privilege management, and deprovisioning. The authentication control is among end users, data owner, and service provider. The privilege management is for some special users.

Authorization and Authentication

In an access session S, as Figure 2 shows, the process can be divided into two parts: <UR, OR, Req> and <UR,SP, P>, in which UR is the set of user roles, OR is the set of data owner roles, SP is the service provider, P is the set of permissions, and Req represents the set of user's access requests where Req=(UR,OR, request index, data block,op).

The authorization procedure works as follows.

1. UR sends a data access request to OR. $UR \rightarrow OR$: {UR, OR, Req}.The request includes the index number of data block that the user wants to access.
2. When data owner receives the request; it will check the integrity and lateness of the message, Then OR needs to check whether UR has the right to access the data blocks which it request to access. If UR passes the check, the OR will send a reply to UR. $OR \rightarrow UR$: {OR, UR, Req, Cred}.Where the Cred is a Credential to the service.
3. UR will send the packet {UR, SP, request index, Cred} to the service provider., then the service provider firstly authenticate Cred, and that the Cred is generated by OR, because the Cred is encrypted by a secret key which is only known by OR and SP; moreover, The Cred has the information of data user and date owner. Next, S will send the permission to UR. So UR gets the access rights to the data blocks.

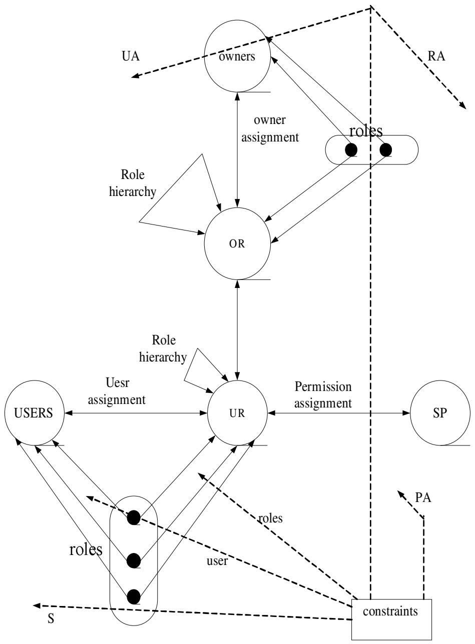


Fig. 1. An access control based RBAC for cloud computing

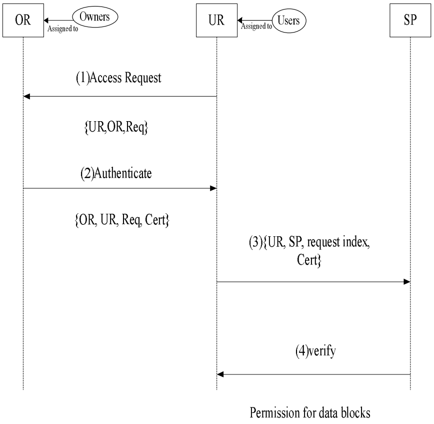


Fig. 2. Authorize Process

The request sent by the end user who has the privilege to send access requests to the data owner is noted as ReqP, and it has the following form.
ReqP=(UR,OR, ID, request index, data block, op).

Deprovisioning and Insertion

When a new role node is added to the system or a role is deleted from the role hierarchy, we use the administrative scope model to update the role hierarchy; this procedure includes operations on roles and edges. These operations can be denoted as AddRole(a ,r, $\Delta r, \nabla r$), DeleteRole(a ,r), AddEdge(a ,c,p), DeleteEdge(a ,c,p) respectively., where a is an administrative role, r is the new role, Δr is the immediate child role of role r , ∇r is the immediate preceding role of role r , c is the child role, and p is the parent role. Mentioning that, OR give credential to UR according to the first come first served with high priority first principles.

Suppose a new role is inserted to the hierarchy. When it wants to access data blocks, first it needs to get access authorization, so it gains access rights in the same way as described in section 3.1. And the service provider and the data owner do not change to adapt to the update. On the other side, if a user role is removed from the hierarchy, besides the UR hierarchy update, the OR also needs to update its Credential management. It labels the data block which the UR was accessing to note that the UR does not have further access to that data block. If a user needs to access the data block again, it is necessary to gain new access rights from the data owner and service provider and if a user needs to change the accessed data block, it sends access request to the data owner, the data owner delete its access rights of the last block and sends a new Credential to the UR. Also note that, the data block which the user does not access is also labeled.

2.3 Instances

In a typical Cloud Computing environment, suppose there are three users: Alice, Bob and Carol. Alice and Bob are assigned to the same role E., see UR and OR hierarchies in Figure 3. Alice requests to access the data block DB1, Bob requests to access the data block DB2 and Carol request to access the data block DB3. Following the proposed model.

(1) Alice sends access request to the owner role OR1 through role E, consider that Alice has higher priority than Bob and Bob cannot get edit permission from E. this request can formally be written as following:

$$\text{Alice} \rightarrow \text{OR1: \{E, OR1, Req\}} \qquad (1)$$

Req = { UR=E,OR=OR1 , request index=1, data block=DB1,permission=edit}

Alice is the first time for Alice to send request, the request index is “1”. And permission =edit means that the access rights Alice needs to gain is to edit the data block.

(2) The role OR1 authenticates the message and replies with the Credential. At the same time, it label the data block in service.

$$\text{OR1} \rightarrow \text{E: \{OR1, E, Req, Cred\}} \qquad (2)$$

Req={UR=E,OR=OR1 , request index=1, data block=DB1,permission=edit} and Cred is a Credential for Alice.

(3)When E receives the message, it first checks the freshness of the message and then sends it to the service provider, and waits to receive the access permission.

$$\text{E} \rightarrow \text{SP: \{E, SP, request index, Cred\}} \qquad (3)$$

The SP receives the request, and checks whether the Cred matches role E. then it sends the Edit permission to role E; consequently, user Alice gets the edit permission to DB1.

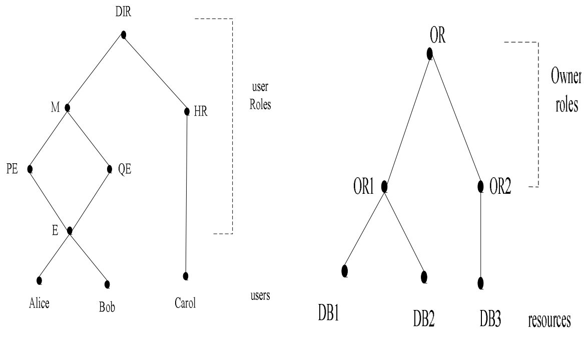
When the role PE is deleted, its administrator will send a notification to the data owner OR. The notification contains the ID of the role PE and data block. Next, OR labels the data block to show that the role’s rights have been revoked and the notification of the information is as following:

$$\text{M} \rightarrow \text{OR: \{PE, OR, ID, data index\}.}$$

When the user wants to access the same data block again, it will need to send an access request to OR.

node	legend
DIR	Director
M	Manager
PE	Production Engineer
QE	Quality manager
E	Engineer
HR	Human Resources
OR	Data Owner
OR1	Data Owner 1
OR2	Data Owner 2
DB1	Data Block 1
DB2	Data Block2

(a) Legend



(b) User role hierarchy (c) Owner role hierarchy

Fig. 3. Application of access control model for Cloud Computing

3 The Security Analysis

In a multi-users environment, access control is usually used to control and protect resources. When an access control policy is made, some security problems should be considered, for example, in the given authorization status and policy description, whether a subject can access the object safely? Security analysis techniques answer two questions: whether an undesirable state is reachable and whether every reachable state satisfies some safety or availability properties.

In Li et al. [8], a version of security analysis is defined in the context of trust management, and Li et al [10], introduced solutions to two classes of security analysis problems, which are assignment and trusted users (AATU) and assignment and revocation (AAR).

3.1 Reduction for AATU

Given an AATU security instance in Cloud Computing $<\gamma=<UA, PA, RA>$, $q = u_1 \sqsupseteq u_2$, $\varphi =< can\_assign, T >$, $\Pi \in \{\exists, \forall\}>$. The can_assign relation determines who can assign users to roles and which preconditions the users should satisfy, where can_assign = {<DA, GT, {Expertss}>, <SA, true, {GU, DA}>}. DA means the department administrator, GT is a general teacher, SA standards the school administrator, and GU is general users. DA, GT, SA, and GU are all end users in the Cloud. Let q reduction results take the form of $<\gamma^T, q^T, \varphi^T>$. q^T is $Sys.Experts \sqsupseteq \{Alice\}$.

We use the principal Sys to represent Cloud Computing RBAC system. The RT [$\longleftarrow$, $\cap$] role Sys.ur represents the user role r in Cloud Computing, and the RT [$\longleftarrow$, $\cap$] role Sys.p represents the cloud computing RBAC system permission p. Each $(u, ur) \in UA$ is translated into the RT [$\longleftarrow$, $\cap$] statement Sys.ur $\longleftarrow$ u, and Each $r_1 \le r_2$ means that a member of r_1 is also a member of r_2.

According to Figure 4, the following RT statements in γ^T result from UA:

- Sys.Students $\leftarrow$ Alice Sys.GT $\leftarrow$ Bob Sys.DA $\leftarrow$ Carol

the following statements in γ^T result from PA:

- Sys.View $\leftarrow$ Sys.DA Sys.Edit $\leftarrow$ Sys.GT

and the following statements in γ^T result from RA:

- Sys.GU $\leftarrow$ Sys.Students Sys.GU $\leftarrow$ Sys.GT Sys.GT $\leftarrow$ Sys.Experts
- Sys.DA $\leftarrow$ Sys.SA Sys.Access $\leftarrow$ Sys.Students

In Figure 3, GT stand for General Teachers, GU stand for General Users, DA represents Department Administrator, and SA represents the School Administrator. Roles are shown in solid boxes, permissions in dashed boxes, and users in ovals. A line segment represents a role–role relationship, the assignment of permission to a role, or the assignment of a user to a role. The following statements in γ^T result from can_assign; such that, the first two statements reflect the ability of a member of Department Administrator to assign users to Students and General Teachers with no prediction, then the remaining statements will reflect the ability of a member of School Administrator to assign users to Experts provided that they are already members of Department Administrator and General Teachers.

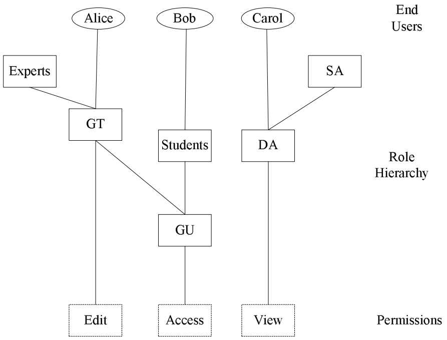
- Sys.Students $\leftarrow$ Sys.DA.Sdtudents Sys.GT $\leftarrow$ Sys.DA.GT
- Sys.NewRole1 $\leftarrow$ Sys.DA $\cap$ Sys.GT Sys.NewRole2 $\leftarrow$ Sys.Experts
- Sys.Experts $\leftarrow$ Sys.NewRole1 $\cap$ NewRole2

$\gamma^T = < G, S >$, where G is the [10] set of growth-restricted roles and S is the [10] set of shrink-restricted roles. Statements defining roles in G can't be added, and statements defining roles in S cannot be removed. Now, it is easy to see that the security analysis instance $< \gamma^T, q^T, \varphi^T, \exists >$ is false, as Alice is not a member of the Experts.

3.2 Reduction for AAR

In Cloud Systems, we use five special principles: Sys, RSys, ASys, HSys, BSys. The Sys roles simulate RBAC permissions, the roles of RSys contain all the initial roles memberships in UA, ASys roles represent every user that exercises the user-role assignment operation, Hsys.r maintains the history of the user role r. And BSys is similar to ASys, but it is used to construct the HSys roles.

An example of a state-change rule in AAR is $\varphi = < can\_assign, can\_revoke >$, where can_assign is the same as in AATU and $can\_revoke \subseteq R \times 2^R$ determines who can remove users from roles. can_revoke consists of two tuples <DA, {GT, Experts}> and <SA, {GU, DA}>.



RA = {(Experts, GT),(GT,GU),(students, GU),(SA,DA)}
PA = {(Edit, GT),(Access, GU),(View, DA)}
UA = {(Alice, GT),(Bob, Students),(Carol, DA)}

Fig. 4. An example RBAC state for cloud computing

Let γ be the same as in AATU and q is $Experts \sqsupseteq Alice$. The output of AAR is $<\gamma^T, q^T, \varphi^T>$. q^T is $Sys.Expert \sqsupseteq \{Alice\}$.

The following RT statements in γ^T result from UA:

- HSys.Students ← Alice RSys.Students ← Alice HSys.GT ← Bob
- RSys.GT ← Bob HSys.DA ← Carol RSys.DA ← Carol
- Sys.Students ← RSys.Students Sys.GT ← RSys.GT Sys.DA ← RSys.DA,

the following statements in γ^T result from RA:

- Sys.GT ← Sys.Experts HSys.GT ← HSys.Experts Sys.GU ← Sys.Students
- HSys.GU ← HSys.Student Sys.GU ← Sys.GT HSys.GU ← HSys.GT
- Sys.DA ← Sys.SA HSys.DA ← HSys.SA,

the following statements in γ^T result from PA:

Sys.Edit ← Sys.GU Sys.Access ← Sys.Students Sys.View ← Sys.Administrator,

and the following statements in γ^T result from can_assign:

- HSys.GU ← BSys.GU Sys.GU ← ASys.GU HSys.DA ← BSys.DA
- Sys.DA ← ASys.DA Sys. NewRole1 ← HSys.GU ∩ HSys.DA
- HSys.Experts ← BSys.Experts ∩ Sys. NewRole1
- Sys.Experts ← ASys.Experts ∩ Sys.NewRole1

$\varphi^T = <G, S>$,where G is the set of growth-restricted roles and S is the set of shrink-restricted roles. Moreover, there exists a state γ_1^T that is reachable from γ^T and has the following statements in addition to the ones in γ^T.

- BSys.D ← Alice ASys.Experts ← Alice

We can now infer that in γ_1^T, HSys.DA $\longleftarrow$ Alice and, therefore, HSys.NewRole1 $\longleftarrow$ Alice, and Sys.Experts $\longleftarrow$ Alice. Thus the security analysis instance $< \gamma_1^T, q^T, \varphi^T, \exists >$ is true.

If we consider instead, the query q_1^T, which is $Sys.Experts \sqsupseteq Alice$, then as RSys.GT is a shrink-unrestricted role, there exists a state γ_2^T that is reachable from γ^T in which the statement RSys.GT $\longleftarrow$ Alice is absent. Therefore, we would conclude that Sys.Experts does not include Alice. So the analysis instance $< \varphi^T, q_1^T, \gamma^T, \forall >$ is false.

We need to consider if only GT has edit right to a test paper and if David has right to access the Testpaper.

Given a state γ,state-change rule

φ={<Students,Access, Test paper>,<GT, Edit and Access, Test paper>}

is authorized to user assigned to GT has right to access and edit the Testpaper. And users assigned to SA has right to assign Students, GT, Experts, DA, or SA to users. For the query $q = \{David\} \sqsupseteq Access$ and $\gamma \to_{\phi} \gamma_1$, the instance $< \gamma, q, \phi, \exists >$ asks whether there exists a reachable state γ_1 such that end user David can access the Testpaper. It is clear that the instance is true. For the query $q = Edit \sqsupseteq \{David\}$, and $\gamma \to_{\phi} \gamma_1$, the instance $< \gamma, q, \phi, \exists >$ asks whether there exists a reachable state γ_1 such that end user David can edit the Testpaper. It is clear that the instance is false.

4 Conclusion

In this paper we introduce a new access control model for Cloud Computing based on RBAC Where both the user and the data owner have a corresponding role hierarchy. Each role hierarchy has administrative roles, so that it can be updated dynamically if a new role is inserted or deleted from the role hierarchy; moreover, the role hierarchy can update roles relationship by administrator scope. We discuss the authentication, privilege management, and deprovisioning of the user access control.

We also analyze the security of the access control system in a test-paper Cloud Computing system and reduced the security problem to two classes of AAU and AAR using the RT[$\longleftarrow$, $\cap$] role-based trust-management language. The results indicate that our proposed model can satisfy the security needs in Cloud Systems.

Acknowledgments. This work is supported by the National Postdoctor Science Foundation of China (20100480936).

References

1. Osborn, S., Sandhu, R., Munawer, Q.: Configuring Role-Based Access Control toEnforce Mandatory and Discretionary Access Control Policies. ACM Transactions on Information and System Security 3(2), 85–106 (2000)
2. Jiang, Y., Lin, C., Yin, H., Tan, Z.: Security Analysis of Maindatory Access Control Model, Systems, Man and Cybernetics 6, 5013–5018 (2004)
3. Ferraiolo, D., Kuhn, R.: Role-based access controls. In: 15th NIST-NISC National Computer Security Conference, October 13-16, pp. 554–563. Baltimore, MD (1992)
4. Joshi, J.B.D., Bertino, E., Ghafoor, A.: A Generalized Temporal Role-Based Access Control Model. reference IEEECS (accepted December 9, 2003. Published online November 18, 2004)
5. Joshi, J.B.D., Bertino, E., Ghafoor, A.: Temporal hierarchies and inheritance semantics for gtrbac. In: SACMAT 2002: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, pp. 74–83. ACM, New York (2002)
6. Li, N., Tripunitara, M.V., Bizri, Z.: On mutually exclusive roles and separation of duty. ACM Transactions on Information and System Security 10(2) (May 2007)
7. Sandhu, R., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role Based Access Control Models. Computer 29(2) (February 1996)
8. Li, N., Mitchell, J.C., Winsborough, W.H.: Beyond proof-of-compliance: Security analysis in trust management. Journal of the ACM 52(3), 474–514 (2005)
9. Koch, M., Mancini, L.V., Parisi-Presicce, F.: Decidability of Safety in Graph-Based Models for Access Control. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 229–243. Springer, Heidelberg (2002)
10. Li, N., Tripunitara, M.V.: Security Analysis in Role-Based Access Control. ACM Transactions on Information and System Security 9(4), 391–420 (2006)
11. Crampton, J., Loizou, G.: Administrative Scope: A Foundation for Role-Based Administrative Models. ACM Transactions on Information and System Security 6(2), 201–231 (2003)
12. Koch, M., Mancini, L.V., Parisi-Presicce, F.: Administrative scope in the graph-based framework. In: Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies (SACMAT 2004), pp. 97–104 (2004)
13. Jung, Y., Chung, M.: Adaptive Security Management Model in the Cloud Computing Environment. In: 2010 the 12th International Conference on Advanced Communication Technology (ICACT), vol. 2, pp. 1664–1669 (2010)
14. Wang, W., Li, Z., Owens, R., Bhargava, B.: Secure and Efficient Access to Outsourced Data. In: CCSW 2009, Chicago, Illinois, USA, November 13 (2009)
15. Mather, T., Kumaraswamy, S., Latif, S.: Cloud Security and Privacy, pp. 18–19. O'Reilly Media, Inc. (2009)
16. Ferraiolo, D., Kuhn, R.: Role-based access controls. In: 15th NISTNCSC National Computer Security Conference, Baltimore, MD, October 13-16, pp. 554–563 (1992)
17. Nyanchama, M., Osborn, S.: Access rights administration in role-based security systems. In: Biskup, J., Morgernstern, M., Landwehr, C. (eds.) Database Security VIII: Status and Prospects. North-Holland (1995)