RZ 3017 (# 93063) 04/20/98
Computer Science/Mathematics 15 pages

Research Report

Quality in the Access to Information

José L. Abad Peiro*

IBM Research Division
Zurich Research Laboratory
8803 Rüschlikon
Switzerland
jla@zurich.ibm.com

Jean-Jacques Quisquater

UCL Crypto Group
Dép. d'Électricité
Université de Louvain-la-Neuve
Belgium
jjq@dice.ucl.ac.be

*Also affiliated with *Université de Louvain-la-Neuve, Belgium*

Research Division
Almaden · Austin · Beijing · Haifa · T.J. Watson · Tokyo · Zurich

Quality in the Access to Information

José L. Abad Peiro[1]

IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Jean-Jacques Quisquater

UCL Crypto Group, Dép. d'Électricité, Université de Louvain-la-Neuve, Belgium

Abstract

In this paper we define access-control properties and metrics that are useful in evaluating the risk of using alternative authorization systems. We define a formal notation to express access-control models, and apply it to a subset of the most representative ones. This notation is used to evaluate analytically a set of properties on authorization systems, e.g., lower bounds of the minimum amount of information required to control access to information sources, performance of access-control schemes, and quality-of-access measurements. We define possible threats to access-control services, and describe how these attacks reflect the quality of an access function. We discuss the classical concept of suitability in the definition of access rights, claiming that it could be considered too conservative under certain circumstances of attacks. Finally, we prove that it is possible to evaluate analytically certain measurements of the performance and effectiveness of particular access-control models in specific threatening environments, rather than simply to rely on empirical results that are difficult to compare. These measurements provide an idea of the quality of an access-control model, at least comparatively.

[1] Also affiliated with *Université de Louvain-la-Neuve, Belgium*

1 Introduction

It is possible to evaluate analytically measurements of the performance and effectiveness of particular access-control models in certain threatening environments, instead of merely relying on empirical results that are difficult to compare. Such measurements provide an idea of the quality of an access-control model, at least comparatively.

Having defined two systems, A and B, in a common formal model, we analyzed the possible situations in which A and B could be considered threatened. Can we give a meaningful answer to the question: is system A more secure than system B? This gives rise to further questions: how much more secure is one system than the other, and in which circumstances?

The first question is addressed mainly by qualitative analysis techniques based on formal security models [LBF+93]. Qualitative analysis techniques examine *how* a system is influenced operationally by certain events. The second question, *how much*, is approached by quantitative analysis techniques, e.g., by measuring distances between states within a Petri net [Dac93], by weighing the risk in the evolution of privilege paths using a graph [DD94], by defining a probability distribution on "levels of threat" that a system can resist [Lee89], or by performing a statistical analysis to show a relationship between security and reliability issues [LBF+93].

In this paper we focus on a subset of the original problems by addressing questions such as: is the access to resources in system A better protected than that to resources in system B? Obviously, such these questions are "easier" to solve than the original one about the comparative security of two systems because the security of a system includes the control of the access to its resources, among other properties [For94].

Our quantitative analysis of authorization systems is based on the level in which security policies for access control are enforced. Figure 1 displays the various levels on which authorization requirements can be implemented. The first level (or *authoritative specification level*) contains security properties that should always be granted in the system, such as: "no access is allowed outside of working hours." The second level (or *level of enforcement of authoritative specifications*) describes states in the system that should never take place. This definition of states can be carried out using a wide range of possibilities, e.g., a simple access matrix [Lam71], a graph of relationships among entities [HMT+90], or a complex set of rules based on the trust that users have in each other [BFL96]. The third level (or *level of implementation*) represents how the enforcement mechanism is really implemented in an operating system, network, or environment in general. Our measurements are taken *only* on the second level of enforcement of authorization requirements.

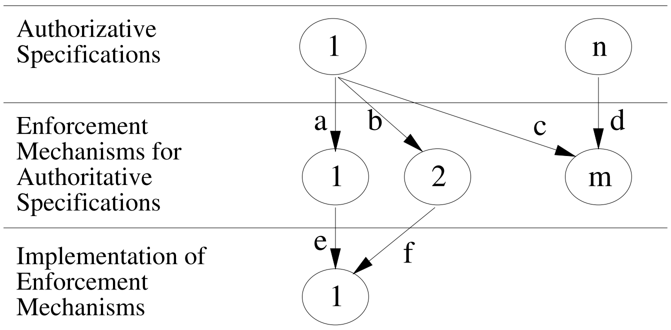

Figure 1: Security requirements for authorization and implementations

Section 2 of this paper defines a formal model of a system in which an access-control function distinguishes legitimate from unauthorized access. The access-control function is defined in

Section 3. We define in Section 4 a set of absolute magnitudes for the access-control function, e.g., the values of functions for *quality of access* and *theoretical pseudo-performance*. We then apply these functions to different access-control schemes. In Section 5 we define situations that threaten the security of a system, and observe the influence of these threats on the evaluation of the quality-of-access function. Finally we compare in Section 6 our theoretical results with those gained from statistical analysis.

2 Definitions

Authorization and access-control concepts are understood differently, depending on the context in which these issues are discussed. We define first informally and then formally certain aspects of access control to outline the ideas treated in this paper.

We regard access control as the security property of a system in charge of allowing legitimate access to its components and denying unauthorized attempts. Thus, access control protects access not only to files and networks, but also to encrypted information, anonymous identities, payment receipts, and locked doors. We define in this paper generic metrics concerning access-control services that are useful for evaluating the security risks of alternative access-control methods.

Informally, let $\mathcal{E}$ be the set of elements, entities, or components of a system, e.g., the set of all processes, files, users, etc.; and let $\mathcal{A}$ be the set of functions on $\mathcal{E}$ called *actions*, e.g., read(A,B), write(A,B), encrypt(A,B), $A, B \in \mathcal{E}$. Each $A \in \mathcal{A}$ is an internal composition law without the associative, commutative, and idempotent properties, and without a neutral element e.

The sets of entities $\mathcal{E}$ and actions $\mathcal{A}$ existing in the system change with time. Furthermore, we distinguish between the set of subject entities, $\mathcal{S}^t = \{S_1 \dots S_n\} \subseteq \mathcal{E}$, and the set of object entities, $\mathcal{O}^t = \{O_1 \dots O_m\} \subseteq \mathcal{E}$, to designate, respectively, the set of entities exclusively performing an action and the set of entities on which actions are exclusively performed at an instant t. $\mathcal{A}^t$ is the set of actions that can be performed in t

$$O_i = A_j(S_k),\ S_k \in \mathcal{S}^t, O_i \in \mathcal{O}^t, A_j \in \mathcal{A}^t \quad .$$

Definition 1 *The context of an entity $E \in \mathcal{E}$, represented by $\mathcal{C}(E)$, is a list of attribute-value pairs defined for the entity E that distinguishes E uniquely from the rest of the entities in $\mathcal{E}$*

$$\mathcal{C}(E) = (v_1^E, \dots, v_p^E), p > 0, \forall E \in \mathcal{E}$$
$$\forall E_1, E_2 \in \mathcal{E},\ \mathcal{C}(E_1) = \mathcal{C}(E_2) \longrightarrow E_1 = E_2 \quad .$$

Similarly the context of an action $A \in \mathcal{A}$ is explained in the following definition.

Definition 2 *The context of an action $A \in \mathcal{A}$ represented by $\mathcal{C}(A)$, is a list of attribute-value pairs defined for the action A that distinguishes A uniquely from the rest of the actions in $\mathcal{A}$.*

$$\mathcal{C}(A) = (v_1^A, \dots, v_q^A), q > 0, \forall A \in \mathcal{A}$$
$$\forall A_1, A_2 \in \mathcal{A},\ \mathcal{C}(A_1) = \mathcal{C}(A_2) \longrightarrow A_1 = A_2 \quad .$$

All attributes $v^E \in \mathcal{C}(E)$ and $v^A \in \mathcal{C}(A)$ have a meaning in a well-defined language $\mathcal{L}$, which we do not describe here. We address only the evaluation and enforcement of access rights, not their definition. We represent by $\mathcal{C}^t(E)$ the context of entity E at the instant t.

3 The Access-Control Function

Based on the definitions above, we formulate a corresponding formal definition of the access-control concept:

Definition 3 *The function of access control,* AC, *of a system with $\mathcal{E}$ components and $\mathcal{A}$ actions is the composition of two function applications AC_{ev} and AC_{en}, such that $AC = (AC_{en} \circ AC_{ev})$, in which*[2]

$$AC_{ev} : \mathcal{C}(S) \times \mathcal{C}(O) \times \mathcal{C}(A) \rightarrow \{T, F\} \times \mathcal{C}(S) \times \mathcal{C}(O) \tag{1}$$

$$AC_{en} : \{T, F\} \times \mathcal{C}(S) \times \mathcal{C}(O) \rightarrow \mathcal{C}(O) \quad . \tag{2}$$

The AC_{ev} application function (1) is responsible for deciding whether a subject $S \in \mathcal{S}$ is allowed to access an object $O \in \mathcal{O}$. For simplicity, we will use the notation $AC(S, O, A)$ instead of $AC_{en}(AC_{ev}(S, O, A))$, $S \in \mathcal{S}$, $O \in \mathcal{O}$, $A \in \mathcal{A}$. For example, a *read* access to the file *f.txt* would be represented by the function $AC(jla, f.txt, read) = f.txt$ in a UNIX operating system.

The AC_{en} application function (2) is defined in a straightforward manner[3] and is similar for most known authorization models, i.e., the subject entity S "performs" application A on target O if the AC_{ev} returns $\{T\}$, otherwise the subject is denied access to the object.

$$AC_{en}(x, S, O) = \begin{cases} \emptyset & \text{if } x = \{F\} \\ A(S) = O & \text{otherwise} \end{cases} \quad . \tag{3}$$

The AC_{ev} function is more difficult to calculate in our model. It differs widely from one access-control model to another. In the rest of this section we present a paradigm of an AC_{ev} function corresponding to discretionary, mandatory, role-based, and crypto-based authorization models.

- In the *access-matrix model* (HRU[4]) for authorizations [Lam71], a matrix contains access rights indexed by users and resources. For example, if a user U would pretend to *read* the file F, the AC_{ev} would only have to check the access matrix M whether $\{read\} \in M[U, F]$. In this model, access of subjects to objects is controlled by means of their unique "identity" represented by an element v^{E}_{ident} of their context $\mathcal{C}(E)$

$$M[i, j] \in \emptyset \cup \{v^{A_i}_{ident}\}, \quad v^{A}_{ident} \in \mathcal{C}(A), \forall A \in \mathcal{A},$$

$$AC^{DAC}_{ev}(S, O, A) = \begin{cases} (\{T\}, S, O) & \text{if } v^{A}_{ident} \in M[v^{S}_{ident}, v^{O}_{ident}] \\ (\{F\}, S, O) & \text{otherwise} \end{cases} \quad . \tag{4}$$

  For example in a UNIX file system, $AC_{ev}(jla, f.txt, read)$ will return $(\{T\}, S, O)$ if the attribute *ownership* in the context of the file *f.txt* is v^{jla}_{ident} and the attribute *access rights* of the file *f.txt* is a list containing the action *read*.

[2] To simplify the notation, we consider time as an element of the context of action A, removing all t superscripts from the formulas.

[3] We do not consider the enforcement function to be the implementation of a specification, but rather the natural continuation of the protocol after an access request. In this way, AC_{ev} would not define an authorization policy, but it would implement one.

[4] The HRU model is named after the safety analysis formalized by Harrison, Ruzzo, and Ullman on the basic access-matrix model [HRU76].

- In a *mandatory access-control model* (MAC) such as the one defined by Bell and La-Padula [BL73], subjects can access objects[5] if they satisfy a specific security policy[6] on the system. The policy determines how any subject must relate to objects in valid accesses.

  Subjects and objects have defined a *security level* in their context. The security level is composed of two attributes, a security classification and a set of categories. The classification attribute takes a value from a fully ordered set; values are for example "Top Secret", "Secret", "Confidential", and "Unclassified." The set of categories is a partially ordered set of classes in which data can be categorized, e.g., classes may be "Research", "Personnel", or "Staff." Usually, the security level of a subject fixes its level of trustiness, whereas for an object it determines its relevance.

  To define security levels formally, we use a binary, partially ordered relationship[7] ($\leq$) in $\mathcal{C}(E)$, such that

  $$\leq \subseteq \mathcal{C}(E) \times \mathcal{C}(E)$$

  $$\forall E_1, E_2 \in \mathcal{E},\ \mathcal{C}(E_1) = \{v_{class}^{E_1}, v_{categ}^{E_1}\}, \mathcal{C}(E_2) = \{v_{class}^{E_2}, v_{categ}^{E_2}\},$$

  $$\mathcal{C}(E_1) \leq \mathcal{C}(E_2) \text{ if } v_{class}^{E_1} \leq v_{class}^{E_2} \text{ and } v_{categ}^{E_1} \subseteq v_{class}^{E_2} \quad .$$

  We represent by $f_t(E)$ the function that associates a security level to each entity E at an instant of time t. The security level of entities may change during their lifetime, but the inequality

  $$f_{t_1}(S) \leq f_{t_0}(S),\ \forall S \in \mathcal{S},\ \forall t_1 \geq t_0$$

  must remain true if t_0 is the instant of time at which the subject S is created. Security levels of subjects and objects will be related with each other by an authorization policy.

  The authorization policy will be different in each mandatory-access-control model. For example in the Bell–LaPadula model the access policy can be defined in two parts as follows:

  - The so-called *simple property* for which a subject can only read objects that have a lower security level, i.e.,

    $$AC_{ev}^{MAC}(S, O, read) = \begin{cases} (\{T\}, S, O) & \text{if } f(O) \leq f(S) \\ (\{F\}, S, O) & \text{otherwise} \quad . \end{cases} \tag{5}$$

  - The **-property* for which a subject can only write[8] objects that have a higher security level, i.e.,

    $$AC_{ev}^{MAC}(S, O, write) = \begin{cases} (\{T\}, S, O) & \text{if } f(S) \leq f(O) \\ (\{F\}, S, O) & \text{otherwise} \quad . \end{cases} \tag{6}$$

---

[5] Note that entities in the model must be of fixed type: subject or object.

[6] Policies would be formally defined to a greater extent, but at this point we can consider them simply as rules.

[7] The relationship is antisymmetric and transitive, but it is not fully connected by the set of categories.

[8] This is a simplification of the *-property, but sufficient for this purpose.

- In a *role-based access control*, subjects and objects are not fixed types of entities. In this system there are several groups of entities, *roles*, each of which is defined as having a particular purpose. Access to entities is defined in terms of roles, i.e., each object[9] is defined as having a set of valid roles that subjects must have for each particular action. This set of subject roles valid to access an object O by an action A is represented by $v^{object}_{A-roles}$. Each subject has a contextual attribute defining an active role, $v^{subject}_{active-role}$, and a subset of the roles in the system from which the entity can change its active role, $v^{subject}_{valid-roles}$. Only subjects having one of the active roles specified by the object for the action can perform such an action on the object. Formally,

  $$AC_{ev}(S,O,A) = \begin{cases} (\{T\},S,O) & \text{if } v^S_{active-role} \cap v^O_{A-roles} \neq \emptyset \\ (\{F\},S,O) & \text{otherwise} \quad . \end{cases} \tag{7}$$

  For example, user *jla* wants to *read* the file *f.txt* from a database controlled by a role-based authorization model. If the contexts are

  $$\mathcal{C}(jla) = \{v^{jla}_{active-role} = \{user\}, v^{jla}_{valid-roles} = \{user, guest\}\}$$
  $$\mathcal{C}(f.txt) = \{v^{f.txt}_{read-roles} = \{admin\}\} \quad ,$$

  the AC_{ev} function would return $\{F\}$ because user *jla* cannot have the *admin* role.

- Let us suppose that an encrypted file system has been designed with the following authorization policy: "only entities that can decrypt a file can access it." Such a policy would be useful, for example, in pay-TV networks. In such systems, there would be at least two alternatives to separate evaluation and enforcement: either the evaluation function is the decryption function itself and the enforcement function simply returns decrypted files, or the enforcement function decrypts files when the evaluation function authorizes it.

  The decryption process is more convenient if it takes place in the evaluation function, the reason being that determining *a priori* the success of decrypting a file is as complex as the decryption process itself.[10] In addition, a simple enforcement function facilitates the comparison of all previously described access-control models.

  To formulate the AC_{ev} function we use an application function defined in the context of entities, $verify : \mathcal{C}(E) \rightarrow \mathcal{E}$, which returns the entity's identity from the cryptographic signatures in the context, $verify(v^E_{credential}) = E$.

  $$AC^{Pay-TV}_{ev}(S,O,A) = \begin{cases} (\{T\},S,O) & \text{if } (v^S_{acReqKey} = v^O_{pubKey}) \\ & \wedge S = verify(v^S_{credential}) \\ (\{F\},S,O) & \text{otherwise} \end{cases} \tag{8}$$
  $$\forall S,O \in \mathcal{E}, v^S_{acReqKey} \in \mathcal{C}(S), v^O_{pubKey} \in \mathcal{C}(O) \quad .$$

  Subjects will encode the access request message using an access request key to access an object securely. The access request key, $v^S_{acReqKey}$, in this case must be the public key of the object. From the subject's point of view, only the object that owns the corresponding private key will be able to decrypt the access request. Authentication of the subject is required by the object itself in order to control who is using its public key.

[9] We retain here the terminology subject-object to distinguish between entities with a role of subject and entities with a role of object.

[10] This assumes that the parameters in the function call are correct. The problem is reduced to determine whether decryption keys are valid.

4 Magnitudes in the Access-Control Space

We develop with this formal access-control notation a set of magnitudes and functions that analytically evaluate the behavior of various access-control models against the attacks defined in Section 5.

Definition 4 *The amount of information needed to control access to data is the minimum number of bits needed to distinguish whether an access is legitimate.*

An example is an access-matrix model that uses an access matrix containing actions that subjects are allowed to perform in objects, i.e., subjects and objects are indexes in the matrix. The amount of information required to control the access would be at most

$$Am(HRU) \leq \mid \mathcal{E} \mid^2 \cdot \lceil \log_2 \mid \mathcal{A} \mid \rceil + 2\lceil \log_2(\mid \mathcal{E} \mid) \rceil + \lceil \log_2(\mid \mathcal{A} \mid) \rceil \quad .$$

But if we use contextual information to index the access matrix, the lower bound on the amount of information to control access would be instead (without request and response):

$$Am(HRU) \leq (\sum_{\forall E \in \mathcal{E}, v_i^E \in \mathcal{C}(E)} \lceil \log_2(v_i^E) \rceil)^2 \times \sum_{\forall A \in \mathcal{A}, v_j^A \in \mathcal{C}(A)} \lceil \log_2(v_j^A) \rceil \quad . \tag{9}$$

Implementations of the access-matrix model usually handle a nonfixed number of entities and actions. In addition, subjects and objects may dynamically change their types. Sandhu proved that if this happens, the access-evaluation function during a period of time is not granted to be convergent [San92]. These two conditions make it difficult to use pure access-matrix models in modern complex OS, specially when using contexts for authorizations. But how good are other models, e.g., role-based access schemes, in terms of resources required?

Theorem 1 *The amount of information required to control a system with an access matrix (in inequality) 9 may be used as a function of time* t *as a lower bound for the amount of information at the instant* t *needed to control the access to any system with* $\mathcal{E}$ *entities and* $\mathcal{A}$ *actions.*

For example, in a system using a mandatory-access-control scheme, the minimum amount of information needed to control access would come from the size of the access matrix and at least two contextual attributes to represent the security level of each entity, one for the security classification and another for the set of categories. The value of the Am function would be

$$Am(MAC) \geq (\sum_{\forall E \in \mathcal{E}} \mid 1 + \log_2(n) \mid)^2 \times \log_2(\sum_{\forall A \in \mathcal{A}} \mid \mathcal{C}(A) \mid) \quad , \tag{10}$$

where $n > 0$ is the number of security levels defined in the system. There will be $|E|/n$ entities per security level.

Definition 5 *The complete evaluation time,* Cet, *of an access-control model is the minimum time that would be needed among all possible systems to implement that access-control model to let each subject in the system access each object executing all possible actions.*

Definition 6 *The average access-evaluation time,* Aet, *of an access-control model is the minimum average time taken by any system to implement that access-control model for evaluating whether a particular subject is authorized to access a particular object.*

Definition 7 *Based on the two previous definitions we can describe a pseudo-performance magnitude $\mathcal{P}$ for an access-control model. Supposing the rate of access requests is uniformly distributed for all systems implementing that model, we can measure the pseudo-performance of the model as the minimum amount of information required to evaluate an access over the average access evaluation time*

$$\mathcal{P} = \frac{Am}{Aet} \quad . \tag{11}$$

All previous definitions are useful for estimating the performance of various access-control models. However we are also interested in finding a magnitude to measure the quality of the service provided.

Definition 8 *Quality of access, $\mathcal{Q}_{AC}$, in an access-control system is the accumulated differences between the rights in the access matrix* M, *at t_0, and the* actual *rights that entities in an access-control system may gain at t_1:*

$$\mathcal{Q}_{AC}(t_0) = M(t_1) - M(t_0),\ t_1 > t_0 \quad .$$

To calculate accurately the function $\mathcal{Q}_{AC}$ we must either "predict the future" at t_1 or calculate t_0 when we are at t_1 by empirically analyzing historical information on the audit files of the system. Whereas the first approach is impossible, the second one is addressed by intrusion-detection systems. We propose in the following section a method to estimate at t_0 the values of the access matrix at t_1.

Obviously, two systems that express the same access rights for a user on an object offer equally good service. Nevertheless, the quality of an access-control model as defined here would indicate the sensitivity of the mode to fraud. The "goodness" of an access-control model in that sense is in terms of its resistance to particular attacks. In the following section we discuss possible attacks on access control and the relationship with the quality-of-access magnitude.

5 Threats to Access Control

Let us suppose that an access-control scenario is placed in a theater like the one shown in Fig. 2. In this example, the AC_{ev} function is located in the ticket office where decisions are taken depending on the money that people have at the time of their access request. The access-enforcement function is implemented by the ticket controller posted at the door of the theater. The controller must verify that everybody entering the theater owns a ticket, i.e., the evaluation function has given them access. However, is this a complete secure system? In other words, is it possible to access the theater without paying?

The following possibilities present threats to an access-control system:

1. Compromise the AC_{ev} function, i.e., the ticket office, to obtain a valid access decision response without authorization, i.e., obtain a ticket without paying.
2. Compromise the AC_{en} function, i.e., the gatekeeper, to access resources for which the subject has not been authorized, i.e., enter the theater without a ticket.
3. Compromise the authorization definition service [11] to obtain permissive access decisions from the AC_{ev} function, i.e., forge a theater ticket.

[11] The access-control function as defined in Section 3 does not include the mechanism to fill in the access matrix with rights, i.e., the authorization definition service.

4. Compromise innocent subjects from which valid access responses are stolen, i.e., steal a theater ticket.

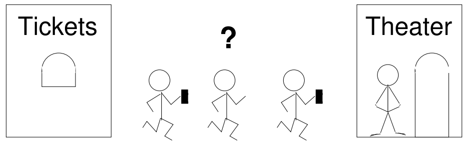

Figure 2: Example of an access-control scenario

To prevent attacks of type 3 we need to fill in carefully the access matrix. Jones depicts the optimal case for access rights *definition* when entities obtain only the resources that they need [Jon73]. This is equivalent to saying that only those who enter the theater need a ticket. However, we consider no threat presented by those who paid but did not attend. It should be sufficient to verify that the number of sold tickets is greater than the number of persons in the theater. Of course we also need to check the transfer of rights (tickets in this case[12]).

It is important to realize the evolution in time of the access rights matrix M. In the lifetime of each entity E there are at least three characteristic instants of time: t_0, t_1, t_2, $t_0 < t_1 < t_2$, as we see in Fig. 3. In t_0 an entity E is created with some fixed access rights, either inherited from another "creator" entity or assigned by a default policy in the system. At t_1, entity E gains a particular set of access rights[13] to access another entity. At t_2, the access takes place.

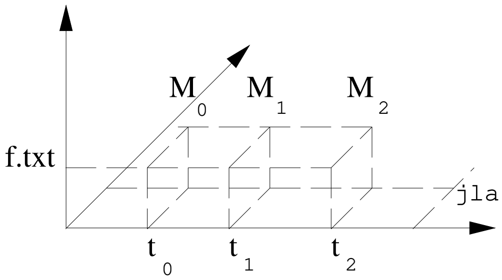

Figure 3: Evolution in time of an entity's access rights

Attacks of type 2 to the AC_{en} function could be minimized by using intrusion-detection techniques, in what we call *a posteriori* access control. The quality of access[14] of Definition (8) would then be measured in t_2 by the formula

$$Q_{AC_{en}}(t_2) = \sum_{\forall E_1, E_2 \in \mathcal{E}, \forall A \in \mathcal{A}} \mid AC_{ev}^{t_2}(E_1, E_2, A) - AC_{ev}^{t_1}(E_1, E_2, A) \mid \quad . \tag{12}$$

The access rights of entities whose state belongs to a t_0 situation will not differ from the access-matrix rights, therefore they do not add value to the quality measurement. The quality of access as defined in Definition (8) is a value to be minimized, in an optimal case $Q_{AC_{en}}^{t_2} = 0$, meaning that no intrusions in the system take place.

[12]We do not disregard the importance of authentication, but we treat it separately from the definition of access control.

[13]Referred to in certain architectures as a ticket.

[14]Note that we use AC_{ev} instead of the entry in the access matrix itself. In this way we study differences among the access-control models.

Attacks of type 1 to the AC_{ev} function are the ones in which it is possible to obtain a ticket for the theater without paying. Unfortunately at this point, it is not possible to apply the quality-of-access measurement, i.e., access has not yet taken place. To solve this problem we need either to predict t_2, or to study the transfer of rights that takes place in t_1 for each entity. We choose the second option.

To study the transfer of rights at t_1 we define the function $create{:}\mathcal{E} \rightarrow \mathcal{E}$ such that $create(E_1) = E_2,\ E_1, E_2 \in \mathcal{E}$ means that the entity E_1 creates the entity E_2. In UNIX-like operating systems when one process creates another, both are initially equal.[15] After some time, $t_1 - t_0$, the new process must derive its own rights. For example, a process such as the *login daemon* running with *root* privileges creates after user authentication a new process at t_0. At t_1, the new process must obtain the privileges accorded to the user. How do we know whether those privileges have been granted properly? We claim that measuring the difference of privileges at the instant t_1 from a new process with respect to its creator is a valid correlation to the quality of the access magnitude, e.g., *in Multics*

$$Q_{AC_{ev}}(t_1) = (\sum_{\forall E_1 \in \mathcal{E}} (\sum_{\forall E \in \mathcal{E}, \forall A \in \mathcal{A}} \mid AC_{ev}^{t_1}(E_1, E, A) - AC_{ev}^{t_1}(E_0, E, A) \mid +1))^{-1} \quad . \tag{13}$$

$$E_1 = create(E_0)\ in\ t_0 \quad .$$

In the next section we describe an alternative way to measure the quality of an authorization system using a statistical model of queuing networks. The analytical correlation between both quality measures, the one in the past chapter and the one to be described, is beyond the scope of this paper and currently a subject of interest for the authors.

6 Statistical Analysis

The model based on a network of $M/M/1$ queues, Fig. 4, represents an access-control service. The attacks are described in Section 5. In this model, access-control requests for service are represented by jobs that arrive at the queuing network. Ordinary access requests follow the path with probability p_2 to reach the queue Q_1, where the access-evaluation function is executed. The subnet of parallel servers with service time μ_2 models the process of construction of a normal access request. The delays[16] $\overline{D_2}$ and $\overline{D_4}$ are, respectively, the time needed to prepare a normal access-control request and the time needed to evaluate it. As access requests are prepared locally to the client, jobs do not have to wait. Therefore, there is no queue associated with the delay $\overline{D_2}$. Once an access request has been evaluated, the access-control enforcement function is computed in the queue Q_2. After Q_2, the access-control service is considered completed.

In this model there is a probability p_1 that a job is a malicious request trying to fake an access-control evaluation request (attack 1 in preceding section), e.g., using a false identity.[17] Attacks of type 2 to the access-control enforcement part, e.g., using a false AFS token, may happen with a probability of p_3. Normal access requests happen with a probability of $p_2 = 1 - p_1 - p_3$. It is very intuitive that $p_2 \gg p_1$ and $p_2 \gg p_3$. The subnet of queues with μ_5 represents the effort required to highjack a valid access-evaluation response, e.g., spuffing network packets. This type of attack happens with a probability of p_4. Intuitively, $p_4 \ll p_5$, where p_5 is the probability of a job being a normal enforcement request after the evaluation.

[15] $AC_{ev}^{t_1}(E_1, *, *) = AC_{ev}^{t_1}(E_0, *, *)$ at t_0 if $E_1 = create(E_0)$.

[16] The Appendix A describes the formulas used to calculate delays.

[17] How actually to estimate real values for probabilities and processing times is discussed below.

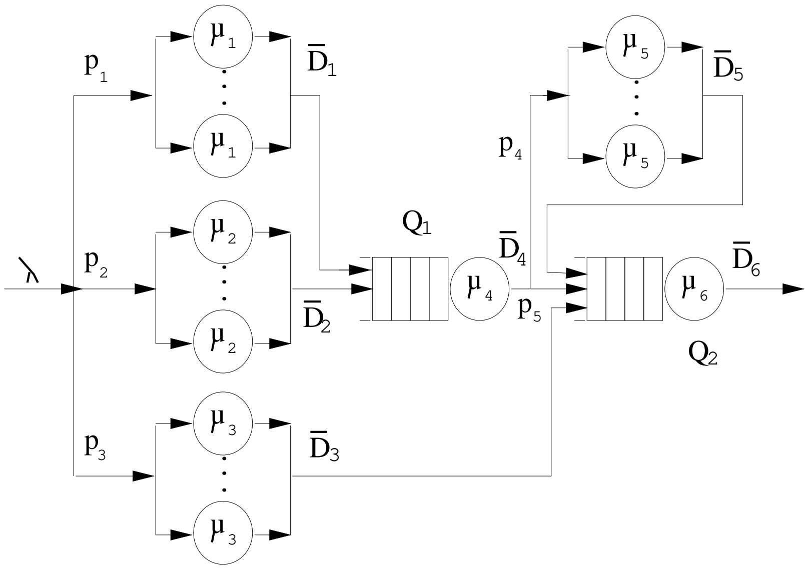

Figure 4: Queuing network for an access-control service with attacks

This statistical model is useful for the comparison of different access-control models under a set of common assumptions,[18] e.g., fixed probabilities for attack occurrences. It is clear that the difficulty in building this model lies in correctly estimating the appropriate processing times and routing probabilities. Nevertheless, one could compare two access-control models and obtain valid comparative results, e.g., by comparing a crypto-based access control using DES with one using a one-time pad algorithm. In this case and assuming that both implementations are perfect, the delay required to break a one-time pad is almost infinite, but the global average delay for normal access requests would be too high for a specific load of the system. With our model, saturation loads could be calculated.

An access-control system in our statistical model is defined by evaluation and enforcement functions, i.e., μ_4 and μ_6. For simplicity, we assume that all other processing times are proportional to μ_4 and μ_6. A reason for this is that the time required to perform an attack depends on the type of access-control model used, and we are only interested in comparative values for alternative access-control systems, not in the absolute values of the functions. For example in a public key-based access-control system, the time required to fake an evaluation response could be assumed to be equivalent to the minimum time required to find a collision in the encryption algorithm. Other valid assumptions would be to use the minimum time required to break the integrity property of the messages (replacing signatures), or to find the actual secret key. Any claim to estimate such times beforehand is probably apt to fail. However, we can still establish relationships among times required for different types of attacks, e.g., stealing a packet from the network would appear to be independent from the crypto algorithm, and thus could be considered fixed in the evaluation of two different crypto-based access-control systems.

Definition 9 *Following this model, a "good" access-control system would maximize the delays suffered when performing an attack, and minimize the delays in serving valid access-control requests. Thus, we define statistically the quality of an access-control system as the rate*

$$Q_{AC} = \frac{\overline{D}_1 + \overline{D}_3 + \overline{D}_5}{\overline{D}_6} \quad .$$

[18]See Appendix A for a complete list of assumptions.

Let us now consider an example. We assume the existence of two access-control systems based on two encryption algorithms DES and IDEA. Both access-control systems are equal in every detail except for the crypto algorithm used, i.e., IDEA and DES (obviously the consequences of using these algorithms involve differences in the protocols, but this is not relevant for our analysis). Requests for access in these scenarios consist of the ACev process, which verifies users' identities after decrypting the access matrix with a private key. For simplicity, we neglect the signature verification step (identities).

To calculate probabilities, we consider a monthly average number of access to be 16200 requests,[19] and a monthly average rate of 2.25 "serious" attacks.[20] For simplicity and for lack of a better estimate, we assume that 50% percent of the attacks are to the evaluation request process, and the other 50% to the enforcement request process. We assume for simplicity too that the probability of a packet being stolen is the same as the other attacks. With these numbers we estimate $p_1 = p_3 = p_4 = 6.94 \cdot 10^{-5}$ and $p_2 = 0.999\hat{8}$.

The processing time μ_4 represents the time required to evaluate an access request with IDEA and DES. The function μ_4 is an exponential function depending on the crypto-algorithm used:

$$\mu_4 = \exp\left(\frac{r \cdot o + bs + ks}{s}\right) \quad ,$$

where r is the number of rounds of the crypto algorithm, o is the number of operations per round, bs is the block size, ks is the key size and s is the comparative speed of both algorithms. For simplicity, we assume the time of evaluation and enforcement to be equal, $\mu_4 = \mu_6$.

The processing time of a normal request to reach the evaluation function, μ_2, is simply an exponential function of a positive constant K_2. The time required to attack the evaluation function depends on the crypto algorithm used.[21] We know nothing else about this function, except that its values should be much higher than those for the function μ_2. We select for this reason $\mu_1 = \mu_4 \cdot \exp(K_1)$, $K_1 > 0$. For simplicity, we assume attacks to the enforcement function to be computationally as difficult as attacks to the evaluation function, $\mu_3 = \mu_1$. Attacks based on "stealing" or highjacking access-evaluation tickets are independent of the crypto algorithm used, $\mu_5 = \exp(K_3)$, $K_3 > 0$. For the graphics we have considered[22] $r = 8, o = 16, bs = 64, ks = 128, s = K_4$ for IDEA, $r = 16, o = 8, bs = 64, ks = 56, s = 2 \cdot K_4$ for DES, and the constant values $K_1 = 100, K_2 = 1, K_3 = 100$, and $K_4 = 10$.

We represent the quality function Q_{AC} and the average global delay $\overline{D}_6$ with respect to the number of access requests per second in the following figures. The curves show that in fast implementations with $K_4 > 1000$, both algorithms have a tendency to perform similarly in terms of the quality function (Fig. 5). For slow implementations, $K_4 > 33.47099$, the quality of DES as defined is higher than that of IDEA because the average delay $\overline{D}_6$ is much greater for IDEA than for DES (Fig. 6). For very slow implementations with $K_4 < 33.47099$, the average delay $\overline{D}_6$ is approximately the same for both algorithms but the quality of DES is still better than that of IDEA. Whereas in the former it is constant, in the latter it grows linearly. The rationale behind this is that the faster the algorithm, the shorter the delays associated with attacks.

[19]This figure corresponds to the average monthly amount of http access requests to one of our servers.

[20]This figure is based on a study of Internet attacks in [Cor97]. The analysis takes place in a "wild" and open context for e-commerce. Hackers have an economic goal for breaking into a system. Our environment differs from the one in the analysis, but we are only interested in an estimation of our probabilities.

[21]We assume a perfect implementation of both systems.

[22]We assume that all hardware and software implementations of IDEA are twice as fast as implementations of DES when using the same amount of resource (memory, computing power, etc.).

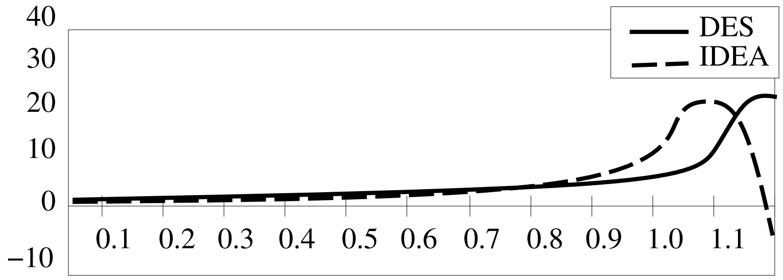

Figure 5: Quality function plot for fast implementations of DES and IDEA

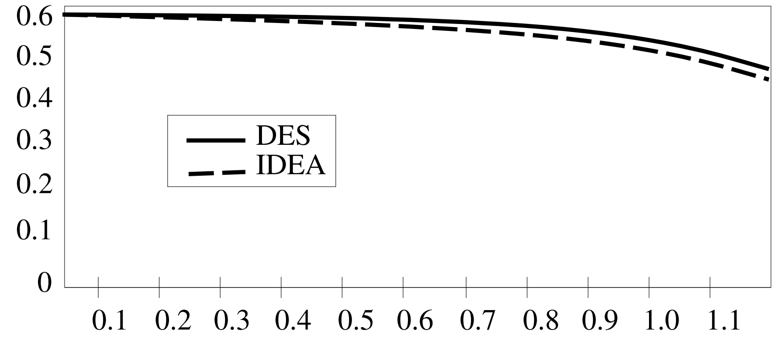

Figure 6: Average global delay plot for fast implementations of DES and IDEA

7 Conclusions

We define an access-control function as a composition of applications, one for the evaluation of rights and the other for access enforcement. We have intentionally omitted the definition of rights as we pursued a quality-of-access measurement independent of how the matrix is filled in. However we have referred to the risks involved in misplacing privileges in the access matrix, and the fact that the criteria proposed by Jones to fill it are too conservative to accommodate the models of possible attacks.

The given definition of access control may apply to different types of access. In particular we consider access to information contents as a point of view from which cryptographic tools may best suit requirements of authorization services.

Based on our definition of access control we have characterized lower bounds in the minimum amount of information required to determine the legitimacy of accesses to information, and have run a performance measurement to compare different access-control schemes as a function of the number of entities, actions, and the size of their contexts. We have also provided analytical tools to compare alternative access-control methods quantitatively as well as qualitatively.

The quality of the information magnitude reflects the "goodness" of an authorization system by analyzing the behavior of entities at different instants of time. Unexpected behaviors are due either to implementation faults or to attacks, but we do not make this differentiation as is done in [LBF$^+$93] and take their influence on the quality-of-access function into consideration. The ways in which access-control systems react against different types of attacks is reflected in the quality measurements.

Another scenario in which the statistical model presented in Section 6 would be useful is to calculate the optimal number of security classes for a Bell–LaPadula system. In a matrix-based access-control system such as UNIX, the number of objects in the system would correlate with the number of possible implementation faults. This assumption would be useful to estimate the exponential functions μ_1 and μ_3.

Further subjects of research can be established by seeking correlations between our measurements and specific characteristics regarding requirements of the first level, for example when specifications are composed [AL90] (arrows a to d in Fig. 1), when the enforcement of different specifications is managed by single implementations [LTBS92] (arrows e and f), or when properties such as the "CCS information security property" are defined to follow flows of information between different levels of security [FG93].

8 Acknowledgments

Earlier versions of this document were read and commented on by Günter Karjoth, Michael Waidner, Uta Wille, and Mehdi Nassehi. We are grateful for their constructive observations and suggestions.

References

[AL90] Martin Abadi and Leslie Lampor. Composing specifications. Technical report, DEC SRC, 130, Lytton Av, Palo Alto, Ca 94301, October 1990.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Research in Security and Privacy, Oakland, CA, May 1996. IEEE Computer Society, Technical Committee on Security and Privacy, The Institute of Electrical and Electronics Engineering IEEE Inc.: Computer Society Press.

[BL73] D. Bell and L. LaPadula. Secure computer systems: Mathematical foundations. Technical Report MTR-2547, Vol 1, MITRE Corp., Bedford, MA, November 1973.

[Cor97] WheelGroup Corporation. Analysis of internet attacks - prowatch secure network security survey. Published in Internet, May-September 1997. http://www.wheelgroup.com/netrangr/PWS_survey.html.

[Dac93] Marc Dacier. A petri net representation of the take-grant model. In *Proc. of the Computer Security Foundations Workshop VI*, pages 99–108, Franconia, New Hampshire, USA, June 1993. IEEE Computer Society Press.

[DD94] M. Dacier and Y. Deswarte. Privilege graph: an extension to the typed access matrix model. In *ESORICS-94*, 1994.

[FG93] R. Focardi and R. Gorrieri. An information flow security property for CCS. In *Second North American Process Algebra Workshop (NAPAW '93)*, 1993. TR 93-1369, Cornell (Ithaca).

[For94] W. Ford. *Computer Communications Security*. Prentice-Hall, 1994.

[HMT+90] A. Heydon, M.W. Maimone, J.D. Tygar, J.M. Wing, and A.M. Zaremski. Miró: Visual specification of security. *IEEE Transactions on Software Engineering*, 16(10):1185–1197, October 1990.

[HRU76] M. Harrison, W. Ruzzo, and J. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, August 1976.

[Jon73] Anita Katherine Jones. *Protection in Programmed Systems*. PhD thesis, Dep. of Computer Science of Carnegie-Mellon University, Pittsburgh, Pa 15213, June 1973.

[Lam71] B. W. Lampson. Protection. In *5th Princeton Symposium on Information Sciences and Systems,*. Princeton University, March 1971. Reprinted in Operating Systems Review 8,1 January 74.

[LBF+93] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2:211–229, 1993.

[Lee89] T. Lee. Statistical models of trust: TCBs vs. people. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Research in Security and Privacy, pages 10–19, Oakland, CA, May 1989. IEEE, The Institute of Electrical and Electronics Engineering IEEE Inc.: Computer Society Press.

[LTBS92] E. S. Lee, B. W. Thomson, P. I. P. Boulton, and R. E. Soper. Composable trusted systems. Technical Report CSRI-272, CSRI, University of Toronto, May 1992.

[San92] Ravi S. Sandhu. The typed access matrix model. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Research in Security and Privacy, pages 122–136, Oakland, CA, May 1992. The Institute of Electrical and Electronics Engineering IEEE Inc.: Computer Society Press.

Appendix A

The following is a list of formulas used to calculate delays for the model shown in Figure 4:

$$\overline{D}_1 = \frac{1}{\mu_1}$$
$$\overline{D}_2 = \frac{1}{\mu_2}$$
$$\overline{D}_3 = \frac{1}{\mu_3}$$
$$\overline{D}_4 = \frac{p_1}{p_1 + p_2} \cdot \overline{D}_1 + \frac{p_2}{p_1 + p_2} \cdot \overline{D}_2 + \frac{1}{\mu_4 - \lambda \cdot (p_1 + p_2)}$$
$$\overline{D}_5 = \overline{D}_4 + \frac{p_4}{\mu_5}$$
$$\overline{D}_6 = (p_1 + p_2) \cdot \overline{D}_5 + p_3 \cdot \overline{D}_3 + \frac{1}{\mu_6 - \lambda}$$

$$p_1 + p_2 + p_3 = 1$$
$$p_4 + p_5 = 1$$

with the following assumptions:

1. The rate of arrival of access requests follows an exponential distribution.
2. The processing times of all elements of the queuing network follow an exponential distribution.
3. μ_4 models the processing time of an access-evaluation request, $\mu_4 = AC_{ev}$.
4. μ_6 models the processing time of an access-enforcement request, $\mu_6 = AC_{en}$.
5. For simplicity, we assume in the example that $\mu_6 = \mu_4 = \exp[(r \cdot o + bs + ks)/s]$.
6. μ_1 models how difficult it is to fake an AC_{ev} request, $\mu_1 = f(\mu_4, \mu_6)$ but we do not know *f*.
7. μ_3 models how difficult it is to fake an AC_{en} request, $\mu_3 = f'(\mu_4, \mu_6)$ but we do not know *f'*. For simplicity, in the example, $\mu_3 = \mu_1 = \mu_4 \cdot \exp(K_1),\ K_1 > 0$.
8. μ_2 models processing times for normal access requests. For simplicity in the example, $\exp(K_2),\ K_2 > 0$.
9. μ_5 models how difficult it is to "steal" an AC ticket (or AC_{ev} response). For simplicity, in the example, $\exp(K_3),\ K_3 > 0$.