COMPOSABLE TRUSTED SYSTEMS

E. S. Lee
P. I. P. Boulton
B. W. Thomson
R. E. Soper

Technical Report CSRI-272
May 31, 1992

Computer Systems Research Institute
University of Toronto
Toronto, Ontario, Canada M5S 1A4

The Computer Systems Research Institute (CSRI) is an interdisciplinary institute formed to conduct research and development relevant to computer systems and their applications. It is supported in part by the Natural Sciences and Engineering Research council of Canada.

This document is the final report of Contract W2213-9-6351/01-SZ issued by the Department of Supply and Services of the Government of Canada. The funding was administered by the Communications Security Establishment of Canada. Milan S. Kuchta was the scientific authority for the contract.

[Wittbold90] Wittbold, J.T., and D.M. Johnson, ''Information Flow in Nondeterministic Systems'', *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, Oakland, Ca., pp. 144-161

[Yu88] Yu, C.-F., and V.D. Gligor, ''A formal specification and verification method for the prevention of denial of service'', *Proc. 1988 IEEE Symposium on Security and Privacy*, IEEE 1988, pp. 187-202.

Throw this page away.

Throw this page away.

Throw this page away.

TABLE OF CONTENTS

PREFACE

The ability to derive the properties of an assembly of components from the properties of its individual constituents is central to good engineering design practice. It is also an underlying axiom in the modern discipline of computer system engineering. A system for which this capability exists is said to be a *composable* system.

In the design of a system, it is desirable to start at quite a high level of abstraction with the overall functional requirements of the system and to step-by-step work towards the details of the individual sub-systems that will be needed. Composability is essential for this successive-refinement design methodology.

Classical successive-refinement design methodologies begin with the creation of an abstract, high-level specification of the intended functional behaviour of a system. This specification is intentionally vague about non-critical details, so that constraints are applied only where required by overall system goals. Step-by-step refinement is applied to the abstract specification to evolve it into a suitable implementation. Details that were not visible or not fully constrained at the abstract level are determined during this process. More and more details of the system behaviour are determined at each step of this refinement process.

The design process starts at the highest level of abstraction, with a full system view. The design is then successively refined to determine the specifications of the components of the system. The system will then meet its overall requirements because each component is designed specifically to do just that.

For many years engineers have been designing systems using standard, pre-designed components to achieve an economy of design effort and multiple other benefits. The system is composed out of sub-systems that have been designed individually and separately. This means that the designer need not start with a blank sheet every time a system is to be composed, because a repetoire of components exist that can be incorporated into a new system.

What is required is that these sub-systems be composable. This means that as long as some simple rules are followed, they can be connected to each other, or to purpose-designed interface components, and the result will be a system that meets some overall criteria determined by the composability properties of the components.

Composability is a desirable goal. We are interested in composing secure information processing systems out of secure components that have been shown to be composable. The individual parts have been evaluated and found to be composable in certain ways. If the proper rules are known to have been followed, and if any purpose-built components that have been included in critical sections of the system have been evaluated and shown to be composable, the system must be secure.

The thrust of this project was to perform research to attempt to find ways to build, categorize, describe, and assemble composable components of secure information processing systems. The successive-refinement methodology may be at odds with the preservation of security. If we think of constructing a secure system out of a collection of parts, each of which has been independently evaluated to meet some security requirements, we can easily arrive at a resultant system that is not secure.

In order to deal with the issue of the level of confidence that can be placed in a purportedly secure system or sub-system, it must possible to measure the assurance that the system or sub-system meets the requirements of a set of criteria. To develop such a measure, there must be some set of values that the abstract notion of assurance can assume, and some way to gauge or quantify the assurance of the system. We want to be able

- to define the security policy that a system must meet, and then
- to derive the functionality that is required of the system, and then
- to develop a measure of the assurance that we require that these goals are met.

We want to do these things without confusing the notion of assurance with the notion that we think of as a guarantee. Additional mechanism may be necessary to provide the function demanded by a security policy, and thus to guarantee that it is fulfilled. Assurance should be a measure that is independent of this, a measure of the level of confidence in the validity of the guarantee.

A guarantee asserts that given security functionality is sufficient to satisfy the requirements of a security policy. Usually, there will be levels of guarantee that can be given, corresponding to levels of functionality provided by the system.

Assurance is associated with any guarantee. It is the level of confidence that a guarantee of security is valid and effective. It is associated with both the quality and the durability of the guarantee.

Suppose that some system is to provide appropriate user functions, and that it must satisfy some known security policy. Some security functions will be implied by the security policy. It must be the case that suitable mechanisms will exist in the system to provide these security functions. Thus, since it provides the requisite function, the overall system gives a guarantee that the security policy is obeyed. The details of what specific functions must be guaranteed to be present, or to be absent, is a significant aspect of the specification of the system. The guarantee must address the intended use or possible misuse of the system, with the risks of security failure and its attendant damage considered. In practice there would undoubtedly be standard guarantees that would suffice for the majority of systems that could be implemented.

The assurance associated with the guarantee, and thence with the system, is a measure of the confidence that the system will enforce the policy without exception or qualification. The assurance is the level of trust that might be placed in the guarantee. The assurance will be determined from many factors.

This research delves into the notions of guarantee and assurance of composable systems. Thinking clearly about the nature of guarantee and assurance is necessary to study the composition of systems. Composing two systems that have a different guarantee is considerably different than composing two systems with a different assurance. We will show that drawing the distinction between guarantee and assurance will help the composition problem.

This is the final report on work performed on DSS Contract W2213-9-6351/01-SZ for the Communications Security Establishment of the Department of National Defence, Ottawa. Much of work under this contract has been reported previously in a series of four-monthly progress reports beginning October 16, 1990. This report includes the results from the progress reports, so that it can be read without reference to them. The work that is included was performed by a team of four research workers. They are:

E. Stewart Lee	Principal Investigator
Peter I.P. Boulton	
Robert E. Soper	
Brian W. Thomson	

Brian Thomson and Bob Soper did most of the day-to-day research. Much of what is of value in the results described in this report is directly attributable to one or both of them. Stewart Lee was responsible for defining the job to be done, for many of the original ideas behind the work, and for the management of the contract that resulted. Peter Boulton participated with the others in the research, in the contract management, and particularly in preparing this final report.

E. Stewart Lee
Toronto, May 31, 1992

1 INTRODUCTION

A little monograph on
the ashes of one hundred and forty
different varieties of pipe, cigar, and cigarette tobacco.
Sir Aurther Conan Doyle,
The Adventures of Sherlock Holmes,
The Boscombe Valley Mystery
1891.

To see so far as one may,
and to feel the great forces that are behind every detail ...
to hammer out as compact and solid a piece of work as one can,
to try to make it first rate ...
Oliver Wendell Holmes, Jr.,
The Class of '61. From Speeches,
1913.

1.1. INTRODUCTION

This document reports on our research to develop measures and means for the composition of secure computer systems from components known to be secure. It is a fundamental principle of this work that the security issues that affect the trustworthiness of subsystems and of composite systems may be related to *guarantee* , which refers to properties of the system specification, and related to *assurance* , which refers our level of confidence that the implemented system conforms to the guaranteed specification. Consequently, the scope of this work includes the composition of guarantee - that is, how to define the functionality of components so that their composition retains desired (security) characteristics - and the composition of assurance - appropriate measures for assurance and techniques for combining these measures in a composite system.

1.1.1. Guarantee and Assurance

In order to deal with the issue of composition of guarantee and assurance, we must first categorize security concerns as being related either to one or the other. In distinguishing the two categories we need to apply a consistent set of definitions. The statement of work issued for this research contained definitions of these terms, and required investigation of the concepts embodied in the definitions. A guarantee was defined as

> an assertion that a given security functionality is sufficient to satisfy the requirements of a specified security policy

while assurance was

> the level of confidence that a guarantee associated with a specified security policy is both valid and effective and that assurance is associated with both the quality and durability of the guarantee.

In these definitions, the term *security policy* is considered to designate

> the set of laws, rules, and practices defining the protection of a system and the information it contains from unauthorized disclosure, modification, destruction, or removal and protection from the interruption of services that the system provides.

Our objectives therefore include developing methods of measuring guarantee and assurance, and methods for deriving the guarantee and assurance of a composed system from the guarantees and assurances of its components.

1.1.2. National Security Criteria

In approaching this research, we believed it important to relate our work both to the formalism of computer science and the practicalities of the published national security criteria documents of several countries. It was our hope that the latter effort would provide guidance for our modelling effort in the form of a target based on official policy statements.

1.2. OVERVIEW

Chapter 2 sets the stage for our work by providing a review and summary of the national computer security criteria published by the United States of America and Germany, and as expressed in the European harmonized security criteria.

In chapter 3, we begin our examination of guarantee and assurance. Guarantee must be composed first, since an assurance without a guarantee is meaningless. Therefore, chapter 3 concentrates on the composition of specifications, and leaves until later the issue of conformity with specification. This is equivalent to making the assumption that all components are utterly reliable in the sense that their behaviour is fully in agreement with their specification (their ‘guarantee’), and then asking what extra security specifications must be applied to the components such that their composition will continue to behave appropriately.

Assurance is taken up in chapter 4. The published security criteria have little to say on the issue of assurance, so much of this chapter deals with the concept of assurance as expressed in other standards documents.

In chapter 5, we begin a survey of the formal models for security to be found in the literature of computer science and engineering. We concentrate on the much researched policies relating to confidentiality. We extend our critique of formal modelling to issues of integrity and denial of service in chapter 6.

Nondeterminism has been the spark for many new models and modelling methods since its introduction to the security scene in 1986. In chapter 7, we review this work and present and support our thesis that nondeterminism is neither necessary nor desirable, and that existing nondeterministic models fail to adequately address the security problems cited as evidence supporting the need for nondeterminism.

Our main results are presented in chapters 8 and 9. We propose that the problem in generating composable security properties results from the attempt to define security for isolated components, and subsequently retrofit composition into the model. We call these ‘‘open’’ models, in that they are based on the analysis of an individual component operating in isolation, without regard to what might actually be producing or consuming the signals that enter or exit it. This chapter suggests the use of ‘‘closed’’ modelling as a basis for the definition of security properties, where a closed model is one in which the component under scrutiny is assumed from the start to be interconnected with an assortment of information sources, sinks, and processors.

In chapter 9, we present a formal logical modelling method for security properties, and translate existing confidentiality models to it. We obtain the expected immediate result that in general, these confidentiality models are not composable. We conclude with a new flow model for confidentiality, the Possibilistic Flow Model. This is a closed model of strong confidentiality with the non-probabilistic simplicity of nondeterminism, but the composability of flow models.

2 GUARANTEE IN NATIONAL COMPUTER SECURITY POLICIES

My history has been composed
to be an everlasting possession,
not the showpiece of an hour.
Thucydides,
The History of the Peloponnesian War,
431-413 BC.,
Book I, section 22.

Guarantees which are not worth the paper they are written on.
Johann Bernhard,
Graf von Rechberg,
In a dispatch concerning the recognition of Italy,
1861.

2.1. INTRODUCTION

Defining a ''measure'' of guarantee is an elusive task. Just what is guarantee, and how can it be quantified? We must develop a quantitative notion of guarantee if we expect to compose trusted systems and deduce the resulting composite guarantee.

Since guarantee is fundamentally a measure of a system's security functionality, we examine here some widely-held notions of trustworthiness assessment. Later in this report we will survey the various notions of guarantee and guarantee measure suggested in the literature by formal security modellers. In this chapter, we direct our attention to another set of documents that indicate the perspectives of those who set security policy. These are the ''national computer security policies'' that various governments are promoting as standards for evaluating trusted systems. We hope to gain from these considerations a set of principles which together make up our intuitive notion of guarantee, and each of which, individually, is relatively straightforward to measure.

In the United States, there exists the *Trusted Computer Systems Evaluation Criteria* [TCSEC85], also known by the acronym TCSEC or, more colloquially, as the ''Orange Book'', that establishes a graded set of measures against which a computer system can be calibrated to establish levels of security confidence. We first explore the hierarchy of security guarantee as it is embedded in that document.

2.2. THE ORANGE BOOK

The Orange Book divides the fundamental computer security requirements into three groups: Security Policy, Accountability, and Assurance. The detailed criteria statements further subdivide these into nine requirements:

Security Policy:
- Discretionary Access Control
- Object Reuse
- Mandatory Access Control
- Labeling

Accountability:
- Identification and Authentication
- Audit

Assurance:
- Operational Assurance
- Life-Cycle Assurance
- Documentation

Neglecting the Assurance requirements for the time being, we will examine the detailed specifications contained in the Orange Book relating to the Security Policy and Accountability requirements.

The Orange Book contains specifications for six classes of trusted systems. We will deal only with the first five since the last two differ only in their Assurance requirements. These five classes (C1, C2, B1, B2, and B3) each contain different combinations of specifications for the six Security Policy and Accountability requirements, ordered such that each successive class contains all the security mechanisms of its predecessors.

Thus the classes form an ordered progression from least-trustworthy (C1) to most-trustworthy (B3), but it is interesting that, given six rather broad categories of requirements, only five levels of trustworthiness were defined (again, neglecting the Assurance requirements). As we will see, this is partly because some aspects of the different requirements are highly interdependent, and partly due to a desire to create a small number of trust classes. This desire may spring from the difficulty in making fine distinctions during the accreditation process, whereas we are interested in making the finest distinctions possible.

2.2.1. Discretionary Access Control

The Orange Book defines Discretionary Access Control (DAC) as:

> A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control).

Discretionary Access Control allows access to an object to be controlled by those subjects who already have access to it. This is in contrast to Mandatory Access Control, which controls access to objects based on global notions of sensitivity and clearance.

Also, the Orange Book requires only that Discretionary Access Control be enforced on ‘‘named objects’’, in contrast to the requirement that Mandatory Access Control apply to all objects, whether or not a subject can refer to them by name. We will make note of this again later.

Three levels of Discretionary Access Control are specified in the Orange Book. There are minimal requirements at class C1, additional requirements at C2, and further requirements at B3. It should be noted that all the requirements stated in the Orange Book are minimal, in the sense that conforming systems can enforce stronger requirements as long as the specified criteria are satisfied.

2.2.1.1. C1 Discretionary Access Control

At class C1, Discretionary Access Control need not apply to all named objects in the system. Thus, some objects may be subject to unlimited access, but the capability must exist for specific objects to be brought under Discretionary Access Control at the request of some user.

This access control, however, need only distinguish accesses by users in distinct named groups. Thus it might be impossible for a user to allow only himself access to an object that he created, unless a group exists of which he is the only member. This may well not be the case since it is not required that users may define their own groups.

Additionally, groups need not be defined as a list of user names; in fact, individual users do not have to be uniquely identifiable. It is acceptable for users to identify themselves to the Discretionary Access Control mechanism as members of some group.

Also, at this level there is no distinction made between the types of accesses that subjects can make. The ability to read an object implies the ability to write to it, as well as the ability to delete it. Therefore, there is no notion of individual ‘‘ownership’’ in C1 systems. Objects belong to groups of users, or to all users. Since no one user controls a specific object, any user with access to an object can change its access groups as well. Thus it is possible for the creator of an object to be denied access to it by another user.

Clearly, C1 systems are intended for use in environments where users can be expected to cooperate, and where the main purpose of the security mechanisms is to prevent accidental damage.

2.2.1.2. C2 Discretionary Access Control

At class C2, Discretionary Access Control must protect all named objects in the system. In the event that a user does not specifically request that access be limited in some way, a default access restriction must be provided.

Access must also be controllable to the granularity of individual users, so that specific users can be allowed access if desired. If group access controls are provided, these groups must be defined as lists of unique user identifiers.

Most importantly, special authorization is needed to be able to change an object's access permissions. A ''control'' permission, distinct from access permission, must be defined per object, so that users with access to an object cannot, in general, change that object's accessibility. Those possessing this control permission can therefore be considered ''owners'' or administrators of the object.

2.2.1.3. B3 Discretionary Access Control

At class B3, access must be specifiable by listing individual users and groups of named users for whom access is allowed, as well as those users and groups for whom access is explicitly disallowed.

Also, different object access modes (e.g. read, write, delete) must be provided. There are no required access modes, but finer distinctions than access/no access must be achievable.

2.2.1.4. Guarantee

We can organize the Discretionary Access Control requirements into three independent considerations:

- which objects need access controls (domain),
- how finely to distinguish among users (control granularity), and
- how finely to distinguish access rights (access granularity).

Each level of Discretionary Access Control can now be seen as defining a location along each of these three axes. The following table summarizes the Orange Book Discretionary Access Control requirements:

	C1	C2	B3
Domain	some objects	all named objects	all named objects
Control Granularity	groups	individual users	users, excluded users, groups, excluded groups
Access Granularity	all/none	control/access/ both/none	control/...others.../ none

Clearly, many finer distinctions among Discretionary Access Control policies could be drawn. We will consider some general evaluation schemes for Discretionary Access Control after investigating the European approach.

2.2.2. Object Reuse

The Orange Book requirements for Object Reuse are quite simple. For classes C2 and above:

> No information produced by a prior subject's actions is to be available to any subject that obtains access to a (storage) object that has been released back to the system.

This statement effectively requires the elimination of all covert storage channels involving reusable shared system resources. It seems to be necessary because Discretionary Access Control only applies to named objects. ''Anonymous'' objects created in the course of a program's execution, say, are not subject to Discretionary Access Control and thus need protection from reuse by other subjects.

Mandatory Access Control, if interpreted in a reasonably broad sense, does protect the information in unnamed system objects, rendering the Object Reuse requirement somewhat redundant. The Object Reuse requirement is quite a bit stronger, however, since Mandatory Access Control would permit reading of ''leftover'' information by dominating subjects, and writing by dominated subjects.

Thus the Discretionary Access Control and Object Reuse requirements appear to form a closely-related pair, as do the Mandatory Access Control and Labeling requirements. The Discretionary Access Control and Object Reuse requirements protect named and unnamed objects, respectively. In fact, Object Reuse can be viewed as a highly-restrictive application of Discretionary Access Control: information in unnamed system objects can only be accessed by the creating subject.

2.2.2.1. Guarantee

Although the Orange Book treats Object Reuse as a single atomic requirement, we can envision quantifying it by considering:

- which reusable objects require purging of information,
- which subjects have access to unpurged released objects, and
- the effectiveness of the information purging.

The Orange Book requires that all released objects be purged before reuse, but, depending on the specific implementation, it may be quite practical (or perhaps unavoidable) for a system to allow some resources to be reused intact. Thus it seems reasonable for guarantee to increase as a larger fraction of a system's reusable objects is purged upon release.

Guarantee for the second aspect of Object Reuse increases as fewer subjects are allowed access to released objects, or as the privilege required for that access increases. For example, a system with very low guarantee in this aspect might allow every subject access to reused objects. Slightly higher in guarantee might be a system allowing access by subjects dominating (in the Mandatory Access Control sense) the information in the objects, as mentioned previously. This approach would likely be feasible and efficient on machines with hardware support for enforcing Mandatory Access Control on regions of memory.

Higher still in guarantee would be a scheme allowing subjects access only to information in reusable objects that they themselves had released. An even stronger requirement would allow only the system security officer access to released objects, presumably for some type of auditing purpose. Finally, the highest guarantee would be provided by allowing no subject, including the subject that released it immediately before, to access information in reusable objects. This is the level of guarantee provided in the Orange Book.

The third aspect of Object Reuse guarantee can be measured by examining the ''left-over'' information content accessible to subjects which are not allowed access. Thus, a very low guarantee results if the information in released objects is untouched before reuse. Higher guarantee would be obtained if the information is encrypted or otherwise randomized prior to reuse. The highest guarantee is provided by clearing all information completely from the released object, by setting all bits in its representation to zero, for example. This, again, is the level of guarantee provided in the Orange Book.

2.2.3. Mandatory Access Control

Mandatory Access Control (MAC) is defined in the Orange Book as:

> A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity.

In contrast to Discretionary Access Control, this scheme protects objects based on their information content. Ordinary users cannot grant access to objects; some central authority certifies (in effect) all access requests.

The Orange Book specifies two levels of Mandatory Access Control. The basic Mandatory Access Control policy must be enforced at B1, and a strengthened requirement comes into effect at B2.

2.2.3.1. B1 Mandatory Access Control

At class B1, Mandatory Access Control must protect all storage objects under control of the Trusted Computing Base (TCB). Subjects and objects must be labeled using the well-known scheme of hierarchical levels and non-hierarchical categories, with at least two hierarchical levels available for labeling.

Labels serve as the basis for enforcing the Simple Security (‘‘no read up’’) Property and the * (‘‘no write down’’) Property, and for ensuring that all subjects created by a user have a clearance dominated by that of the user.

2.2.3.2. B2 Mandatory Access Control

Class B2 Mandatory Access Control differs from the B1 requirements in only one respect. The access controls must now extend to cover all resources directly or indirectly accessible by subjects. For systems on which the TCB controls all subjects and objects (i.e. there is no untrusted portion), this appears to imply only extending Mandatory Access Control to I/O devices. Information imported and exported by the system must already be labeled and protected in class B1.

2.2.3.3. Guarantee

We can organize the Mandatory Access Control requirements into three independent considerations:

- which objects need access controls (domain),
- how finely to distinguish levels (level granularity), and
- how finely to distinguish categories (category granularity).

Each level of Mandatory Access Control can now be seen as defining a location along each of these three axes. The following table summarizes the Orange Book Mandatory Access Control requirements:

	B1	B2
Domain	all storage objects	all resources
Level Granularity	> 1	> 1
Category Granularity	no requirement	no requirement

Again, many finer distinctions could be drawn.

2.2.4. Labeling

The Orange Book Labeling requirements are closely tied to Mandatory Access Control. These are mainly common-sense restrictions on the pairing of subjects and objects with their labels.

2.2.4.1. B1 Labeling

At class B1, all subjects and all storage objects must be labeled. This includes any data originating outside the system, which must be labeled by an authorized user.

Any information exported from the system must also be labeled. Each communication channel and I/O device is designated as single- or multi-level, and information passing over a multi-level channel must be unambiguously paired with its label. Any change in level of a channel or device must of course be controlled by the TCB.

2.2.4.2. B2 Labeling

At class B2, users must be immediately notified of any change in their clearance. They must also be able to query their clearance at any time.

Also, since B2 Mandatory Access Control extends to I/O devices, all devices must be assigned minimum and maximum security levels, reflecting the devices' ''physical environments.''

2.2.4.3. Guarantee

Very little of the Orange Book Labeling requirements specify more than the minimum functionality necessary for implementing Mandatory Access Control. However, there are aspects of Labeling itself that are sufficiently distinct from access control considerations that they may serve as useful metrics for Labeling guarantee. These relate to the ''reliability'' of labels:

- how well labels reflect the true level of their associated information, and
- how tightly labels are bound to their associated information.

The first consideration relates to a potential problem with any Mandatory Access Control implementation: who decides the level of information when it is created? It appears that a higher measure of guarantee should accrue when an authorized security officer makes this determination, rather than allowing the creating user to do so. In fact, one can imagine scenarios wherein groups of highly-trusted users determine the level of certain highly-classified pieces of information. This should imply a much higher measure of guarantee than if the creator(s) were alone given this responsibility.

In fact, this problem takes on a much greater importance when we begin to consider the composition of systems: how much should one system trust another system's labels? Assuming no errors occur in communication, this depends mainly on the quality of ''separation'' observed in the other system. This topic is discussed in some depth in chapter 4 of the final Trusted Network Architecture report [Thomson88].

The second consideration above is a measure of the ''strength'' of an implementation's labeling support. Mandatory Access Control increases in effectiveness as labels become progressively more intrinsic properties of the information stored in objects. For example, a high measure of guarantee should result when an object and its label are cryptographically bound together, so that they cannot be separated outside of the TCB. Likewise, hardware support such as restricting different levels of information to reside in distinct regions of memory, or using tag bits to label individual words, should lead to a high guarantee for this aspect. Alternatives to this type of low-level information labeling include formal verification of program source code and use of ''certified-trustworthy'' system programs.

2.2.5. Identification and Authentication

The Identification and Authentication requirements in the Orange Book exist to support the provision of discretionary and mandatory access controls. Thus the graduated requirements detailed below parallel the graduated Discretionary Access Control and Mandatory Access Control requirements.

2.2.5.1. C1 Identification and Authentication

At class C1, users must identify themselves to the TCB before accessing protected information. Some protected mechanism must be used for the authentication, and the authentication data must be protected from unauthorized access.

2.2.5.2. C2 Identification and Authentication

Since C2 Discretionary Access Control requires protection for all named objects, the additional C2 Identification and Authentication requirement is that all users must be uniquely identifiable.

2.2.5.3. B1 Identification and Authentication

Since Mandatory Access Control comes into effect at B1, the Identification and Authentication requirement at B1 is that the TCB maintain authentication and clearance information for each user.

2.2.5.4. B2 Identification and Authentication

At class B2, a trusted path is required for login. The establishment of this trusted path must be initiated by the user. This requirement seems in keeping with the extension of Mandatory Access Control at B2 to the protection of I/O devices.

2.2.5.5. B3 Identification and Authentication

Class B3 Identification and Authentication makes additional demands on the trusted path mechanism. A trusted path must be used whenever a ‘‘positive TCB-user connection’’ is required, not only at login. Additionally, this path must be ‘‘logically isolated and unmistakably distinguishable’’ from other paths.

2.2.5.6. Guarantee

The Orange Book Identification and Authentication requirements can be organized along the following three axes:

- what users need to be identified,
- what information must be retained per user, and
- the strength of authentication required.

Although C1 systems do not require every system user to be uniquely identified, this is clearly a necessity when a reasonable amount of trust is required. Also, the information maintained per user is mainly dependent on the nature of the access controls and authentication mechanisms in effect.

The last consideration is the central issue here. It is complicated by the wide variety of practical authentication schemes, and by the changing requirements on authentication schemes in different situations. Since this topic is addressed in some detail in the German criteria, we will reserve further comment until that has been discussed.

2.2.6. Audit

The requirements for Audit in the Orange Book are rather demanding at class C2, and grow in complexity at higher levels. Auditing essentially provides a trail of accesses to protected objects, hopefully useful in providing assurance of a system's secure status, and in tracing activities that may have led to security violations.

2.2.6.1. C2 Audit

At class C2, the audit trail consists of events of the following types:

- identification and authentication transactions,
- introduction of objects into a subject's address space,
- deletion of objects,
- actions of administrators, and
- other ''security relevant'' events.

Each event in the audit trail must contain:

- the event's type,
- the time at which it occurred,
- the user involved, and
- whether the event was successful (if applicable).

For identification and authentication events, the logical/geographical origin of the request must be included. For events concerning introduction and deletion of objects, the name of the object must also be included. Also, the facility must exist for particular users' actions to be audited selectively.

2.2.6.2. B1 Audit

Class B1 introduces additional requirements that are clear manifestations of the Mandatory Access Control requirements at this level.

First, an additional event type is required to audit any override of human-readable output labels. In addition, events concerning introduction and deletion of objects must also include the object's security level. Also, actions must be selectively auditable based on the security level of the object involved.

2.2.6.3. B2 Audit

At class B2, the only additional requirement is for the audit of events which are believed exploitable as components of a covert storage channel. This parallels the requirement that a ‘‘thorough search’’ for covert storage channels be conducted at B2, a task that forms part of the Assurance requirement of the Orange Book. This will be discussed later.

2.2.6.4. B3 Audit

The B3 audit requirements demand a sophisticated mechanism for analyzing events as they occur, in order to detect imminent violations of the security policy. Once some ‘‘thresholds’’ have been exceeded, the administrator must be notified, and, if necessary, some actions may have to be taken to actively prevent the violation from occurring.

2.2.6.5. Guarantee

We can organize the Orange Book Audit requirements into these four considerations:

- which events are audited,
- the completeness of the event report,
- the flexibility of audit control, and
- the extent to which the audit trail is automatically analyzed.

It seems clear that a system’s Audit guarantee should increase as more information is captured in the audit trail. However, measurement along this axis is complicated by the varying security-relevance of different types of auditable events. Some ordering among these events must be defined such that audit trails containing events of high relevance to security receive a higher guarantee rating than more voluminous audit trails containing events of low importance.

Likewise, the information recorded per event report must be measurable in a security-relevant manner for comparison of Audit guarantees to be meaningful.

Measuring flexibility of audit control is problematic at first glance, but certainly the control which a security officer wields over audit trail production is a very important measure of a system’s resistance to unauthorized use. We will see, in fact, that flexibility of control over a system’s security features is one of the major principles behind guarantee assessment.

Finally, the provision of tools for the automated analysis of audit trails vastly improves the utility of auditing, but again, comparing guarantees in this respect will be a difficult task.

2.2.7. Conclusions

We have produced, by a careful examination of the security functionality requirements of the Orange Book at each class, a set of principles which characterize a system's trustworthiness in a relatively straightforward manner. Each principle is largely independent of the others, and, we hope, spans the space of possible trusted systems which adhere to the underlying policies in the Orange Book.

We will return to discuss the ‘‘independence’’ and ‘‘completeness’’ of this set of principles, after considering the German and European approaches. At that time we will form some general principles of security assessment which should make possible the accurate determination of a system's guarantee.

2.3. THE GERMAN SECURITY CRITERIA

In 1989 the German Information Security Agency published "IT Security Criteria: Criteria for the Evaluation of Trustworthiness of Information Technology Systems"[ITSEC89]. This document contrasts sharply with the American TCSEC of 1985, and not only due to its green colour. The Green Book is less formal, less redundant, and less rigidly prescriptive than the Orange Book. It offers a more comprehensive treatment of computer security issues, but would likely be less useful than the Orange Book as a guide for evaluating contemporary trusted systems. However, for our purposes, the Green Book is an abundant source of information and ideas. We will study it in some detail.

It should be noted that the Green Book discussed here is labeled a "First Version", and was meant to be rapidly updated, as well as accompanied by more detailed manuals on threat assessment and trusted system evaluation. Moreover, it has likely been rendered obsolete by Germany's participation in the "Harmonized Criteria". However, the latter document has appeared only in a preliminary draft form, and follows the general form of the German effort, though it happens to eliminate much of the material that we will be examining below. Therefore, we feel that the Green Book is currently of great value, mainly for its enumeration and organization of contemporary security issues.

Fundamentally, the Green Book differs from its American predecessor by not advocating a particular security policy, but attempting to address the security concerns of a wide variety of commercial and governmental users.

A very broad spectrum of security policies can be addressed by dealing with the following eight "Security Functions":

- Identification and Authentication,
- Administration of Rights,
- Verification of Rights,
- Audit,

- Object Reuse,
- Error Recovery,
- Continuity of Service, and
- Data Communication Security.

This is an interesting classification scheme that contrasts with the Orange Book's main groupings:

- Discretionary Access Control,
- Object Reuse,
- Mandatory Access Control,
- Labeling,
- Identification and Authentication, and
- Audit.

The Green Book's "Rights" categories subsume the MAC and DAC groupings, and the "Error Recovery" and "Continuity of Service" functions address areas that the Orange Book considers assurance requirements. The Orange Book's "Labeling" considerations are dealt with by the Green Book's "Identification and Authentication" function. We will consider these contrasting perspectives when we deal with the Security Functions in turn below.

A particular security policy is defined by stating specific requirements for each security function. Conformance of an implementation to the policy is then evaluated by examining the mechanisms (e.g. algorithms and architectural features) employed for each function. Mechanisms are scrutinized to determine whether:

1. they are sufficient to completely enforce the security policy,
2. given correct implementation, any fundamental weaknesses exist that allow policy violation, and
3. the mechanisms have been correctly implemented.

These three considerations address aspects of both guarantee and assurance. Item 1 is determined by the quality of the mapping from security policy to system specification, and item 3 must be evaluated after implementation. Item 2, on the other hand, is a question of "design", and is therefore a different type of consideration. Although we have yet to rigorously define the terms *guarantee* and *assurance,* it seems clear that guarantee is fundamentally a measure of design quality, while assurance is a measure of the qualities of the design and implementation processes. Thus, we will concern ourselves here (as does the Green Book) with the evaluation of mechanisms to determine their effectiveness in the absence of implementation flaws (that is, with item 2).

The fundamental weaknesses of item 2 above are of two kinds:

- inherent weaknesses of the mechanism which cannot be practically eliminated by organizational means (that is, by means external to the computer system itself), and
- weaknesses which can be effectively reduced or eliminated by organizational means.

The rating of a mechanism is complicated by weaknesses of the latter kind, since it then involves evaluation of a much larger field of activity. It is essential, therefore, that it be ascertained for each mechanism employed:

- which aspects of the security policy does the mechanism attempt to address,
- what fundamental weaknesses does the mechanism possess (and their possible effects on other related mechanisms), and
- what organizational measures must be employed to ensure the mechanism's effectiveness.

Given these last two items, a measure for the mechanism can be ascertained. A six-level quantization of mechanism effectiveness is recommended:

Ineffective
: The mechanism has no (positive) effect on policy enforcement.

Weak
: The mechanism is suitable for preventing accidental policy violations, but offers little resistance to deliberate attempts at violation.

Moderate
: The mechanism provides some protection against deliberate violation, but can be defeated with moderate effort by those familiar with the system.

Strong
: The mechanism provides good protection against deliberate violation, and can be defeated only with great effort or extensive insider support. Any organizational measures necessary to support this level of effectiveness must be simple and reliable.

Very Strong
: The mechanism provides very good protection against deliberate violation, and can be defeated only with great effort and very extensive insider support. Any organizational measures necessary to support this level of effectiveness must be relatively simple and largely unsusceptible to fault or error. Monitoring and control of these measures must be integral to the mechanism and rated as a part of it.

Virtually Unbreakable
: The mechanism prevents all violations of the relevant aspects of the security policy, and is practically impossible to overcome given the current state of the art. All organizational measures employed must be completely protected from error by internal monitoring which forms an integral portion of the mechanism.

Again, issues of both guarantee and assurance are in evidence here, but it appears that the Green Book is, in the sections that we are about to examine, concerned largely with what we would term guarantee. Another section of the Green Book presents a graduated assurance measurement scheme that partly incorporates the rating of mechanisms, however, so there is some ambiguity as to how to interpret some of the discussion. In the following, we will ignore further guarantee/assurance distinctions, and present the issues just as the Green Book does.

For each of the eight security functions, the Green Book lists a series of topics relevant to the function which must be considered when forming a security policy. As well, a list of topics is given which may form a basis for evaluating relevant mechanisms. Below we discuss the security functions in turn.

2.3.1. Identification and Authentication

The identification and authentication requirement is discussed first since it is a prerequisite for the three functions following, concerning access control and audit.

Important issues in identification and authentication are:

- which subjects and objects require identification only,
- which subjects and objects require authentication as well,
- what circumstances necessitate identification and authentication, and
- actions to be taken on authentication failure.

The Green Book does not make, for many purposes, a clear distinction between subjects and objects. The term *object* is defined in a Glossary as an entity that "plays the passive role in the administration of rights", but the term *subject* is not defined. Let us define, for the time being, *subject* as "an active entity which seeks access to objects and other subjects." Subjects can be users or processes, there being no requirement that a process must act on a specific user's behalf.

It is a consequence of the Green Book's "Rights" discussion (see below) that subjects may seek access to other subjects. Clearly, in this case, the subjects being accessed must be processes. One could then attempt to split processes into subject and object components, and then make the statement "only subjects access only objects", but the Green Book does not address this detail.

It is interesting that the Green Book treats, according to the four issues listed above, identification and authentication as discrete, atomic operations, rather than viewing authentication as an integral, but variable-strength, component of identification. The strength of authentication is taken care of by the mechanism evaluation step discussed below. It appears that authentication is viewed as a binary design choice, with strength an assurance issue. However, we will discuss the mechanism rating here, since we feel that system designers want at least some flexibility in choosing authentication strength, and make other design choices based on it.

Also, the idea of authenticating objects, rather than merely identifying them, addresses the Labeling issue, and may be extended to include correctness and completeness checks on objects, addressing (at least a portion of) the Integrity issue. While authentication of a subject usually involves some exchange of information with the subject (possible due to the active nature of subjects), authentication of an object must be performed by inspecting the object's data. These sorts of considerations help to blur any distinctions one would like to make between identification and authentication.

Mechanisms implementing this function must be evaluated based on:

- the uniqueness offered by the identification scheme, and
- the possibility of authentication deception.

A guideline for rating an authentication mechanism based on uniqueness of the identification is provided. This is interesting because definite numerical values are given, which supports our hope that a quantifiable measure of guarantee is achievable.

In the following table, "P" represents a range of probabilities of incorrect identification, and "Rating" represents the best rating attainable over that range:

P	Rating
10^{-2} to 1	Weak
10^{-4} to 10^{-2}	Moderate
10^{-6} to 10^{-4}	Strong
10^{-8} to 10^{-6}	Very Strong

Thus, a "very strong" authentication mechanism must incorrectly identify fewer than one entity per million. For example, a user-authentication system which can distinguish among 1000 fingerprints is at best a "moderate" strength mechanism.

When evaluating the possibility of authentication deception, additional factors must be considered. These differ, depending on the authentication mechanism in use, which can take three forms. Authentication by ownership involves the presentation of some object (e.g. a hardware key) as proof of identity. Authentication by knowledge requires that some "fact" (e.g. a password), or "ability" (e.g. the computation of a secret algorithm) be presented as proof of identity. Authentication by characteristic features involves inspection of the subject or object for some unique, distinguishing characteristic (e.g. a fingerprint). These additional evaluation factors are listed below.

1. Authentication by Ownership

   - the possibility of theft of the object used for authentication
   - the effort required for forging the object

This mechanism is prone to theft or forgery, and the only protection against these threats is an additional authentication action. Also, the channel used for communication with the authentication object must be evaluated, and if the object contains intelligence (e.g. a chip card), it too must be rated as a component of the overall system.

2. Authentication by Knowledge

   - the effort required for obtaining the knowledge

Knowledge can be transferred intentionally or accidentally outside of the system, and it is possible to defeat this type of mechanism by trial and error. The number of guesses possible per unit time, and the necessary special knowledge must be determined to assess the trial-and-error threat. Again, an additional authentication action is the only protection, and the communication channels must be rated as well.

3. Authentication by Characteristic Features

   - the effort required for forging the characteristics

The features must be sufficiently unambiguous to differentiate among all possible identities, and firmly linked to the subject or object to be identified. Forgery is the main threat, and, if the characteristic features are not of an IT (information technology) nature, an external assessment of their suitability is required.

Clearly this discussion is oriented toward authentication of users, but many of these concepts can be applied to authentication of objects and processes as well. Authentication by ownership is best exemplified by the association of objects and processes with “capabilities” — large random bit-strings, essentially unforgeable, yet often prone to copying. If unauthorized passing of capabilities cannot be prevented, then authentication must be ensured by other means. The point is that authentication of users by ownership, or objects and processes by capabilities, is rarely foolproof unless very tight control can be maintained on the system’s operating environment.

Authentication by knowledge is analogous to process authentication in the sense that password exchanges with users are analogous to challenge-response protocols in distributed systems. In fact, the greater computational power of processes relative to human users implies that authentication by “ability” may be better suited to processes. There is no natural analog for objects.

Authentication by characteristic features is essentially equivalent to authentication by ownership for objects and processes.

It is interesting that evaluation of this most fundamental of security requirements seems to require some form of composition of secure components. This is because authentication of subjects requires some sort of communication with peripherals for authenticating users or foreign processes. All elements on this communication path must be highly-trusted before the effectiveness of the authentication mechanism itself can be verified.

2.3.2. Administration of Rights

The Administration of Rights security function addresses the central issue of computer security: access control. The discussion in the Green Book is of a highly speculative nature, making it applicable to a very wide variety of access control requirements.

Subjects can possess rights with regard to other subjects and objects. The following may be relevant components of a system security policy:

- which subjects and objects (or classes of these) are subject to the administration of rights,
- which kinds of rights can exist between subjects and objects,
- who is entitled to grant or alter rights,
- what rules have to be enforced for the granting or alteration of rights,
- what conditions have to be fulfilled prior to the granting or alteration of rights,
- what roles have to be defined by the administration of rights,
- what rights are associated with special roles, and
- what roles are mutually incompatible.

The first component above corresponds to the Orange Book notion of trusted and untrusted portions of a single system. The TCB need not administer all subjects and objects, though one might expect that a highly-trusted system would have no untrusted portion, as required by Orange Book levels higher than C1.

The second component illustrates that the Administration of Rights security function encompasses the central concerns of computer security: access rights between subjects and objects. These "kinds of rights" include, for example:

- "reading" (non-modifying access) of objects by subjects,
- "writing" of objects by subjects,
- "appending" onto objects by subjects,
- "creation" of objects by subjects,
- "spawning" of new subjects by subjects, and
- "messaging" between subjects.

Clearly these rights have confidentiality and integrity concerns, but they also address a large portion of the availability problem, since part of guaranteeing availability is limiting access which can affect proper system functionality.

The third component, "who administers rights", is a crucial aspect of a security policy. In fact one of the prime distinctions between Discretionary and Mandatory Access Control mechanisms is the centralization of control enforced by MAC. In addition changes in the control structure produce a wide variety of DAC policies. Research has also lately considered the formulation of multi-person control policies, especially for the implementation of multiple-role mechanisms similar to those employed in certain commercial and military decision-making systems.

Rules for granting or altering rights encompass the access restrictions which are central to traditional security policies. Examples of access rules are the Simple Security and Star Properties of the MAC mechanism, the access-control list often used to implement DAC, and the validation requirements in the Clark-Wilson integrity mechanism.

The following component, on first glance, seems identical in intent. However, it is likely that the "conditions" in question are requirements on the operating environment at the time of rights administration. For example, changing the allowable rights on a particular set of objects may require that no subjects currently possess access rights to any of the objects. In some cases, altering the rights structure in certain ways may require that the system be brought into a "single-user" mode.

The final three components concern the function of "roles" in the administration of rights. The Green Book defines a role as "a grouping of rights allocated to a subject", which implies that, since rights exist between subjects and other subjects and objects, every subject has a "role" by definition. In fact this component is concerned with the possible requirement that certain "special" roles may need to be defined, for example, the roles of system administrator and system security officer. A policy may define these roles by specifying the rights the subjects in question have with respect to certain pre-defined subjects and objects, and to certain classes thereof. If a large number of roles is defined, some care must be taken that they are mutually compatible, in the sense that rights and responsibilities are partitioned in a consistent and reasonable manner.

Mechanisms implementing this function must be evaluated based on:

- the completeness of the administration of rights,
- the unambiguity and consistency of the structure of rights,
- the clarity of the structure of rights,
- the protection against covert changing of rights,
- the protection against development of unalterable rights relations,
- the administration of associated rights when deleting or renaming subjects or objects, and
- the protection against restrictions in the capability of exercising rights.

The "completeness" criterion is clearly the dual of the first policy component, concerning which subjects and objects are subject to rights administration. The intent appears to be that evaluation in this respect consists of verifying that the portion of the system which is designed to be trusted, is in fact administered by the implementation. Whether or not all accesses in this portion are properly restricted is a matter for the Verification of Rights function. Also, it must be demonstrated that all forms of rights can be granted as required by the security policy.

Unambiguity, consistency, and clarity are metrics of the practicality of a particular rights structure. Mandatory Access Controls based on levels of sensitivity are generally simple in structure, as long as the number of "need-to-know" categories is reasonably small. Discretionary Access Controls, on the other hand, can quickly lead to rights relations of great complexity and possibly unintentional structure, especially given a large number of users. If the structure of rights is such that the consequences of altering some right are not immediately obvious, the effectiveness of the mechanism is substantially weakened. In cases where complex rights structures must necessarily evolve (as would be the case in most real-world environments), some kind of automated support for the display and interpretation of rights relationships would be necessary.

The rights structure must itself be immune to covert alteration, of course, though, if the rights structure is viewed as an object, or set of objects, it could conceivably be protected by the normal mechanisms. If this were the case, there would remain the necessity for a higher-level protection scheme for certain rights data, since, in the limit, no protection mechanism can protect itself. However, it should be the case that this data is sufficiently simple and static in structure that very different, highly trusted, mechanisms would be available to safeguard it.

Unalterable rights relations may develop in exceptional circumstances, such as the loss of hardware keys, the simultaneous departure of critical security personnel, or machine malfunction. It must be possible in all cases to recover system functionality while maintaining secure operations. This will usually involve removing the system from normal use temporarily.

The deletion and renaming of subjects and objects has implications that must be dealt with, in similar fashion to the Orange Book “object reuse” requirements. Object Reuse is in fact a separate Green Book security function, to be discussed below.

Finally, the criterion dealing with restrictions in the exercising of rights is primarily an availability concern. It must be determined whether denying access (for example, due to the existence of multiple writing processes, or to avoid a deadlock situation), contravenes the security policy. The policy must be formulated with these possibilities in mind.

2.3.3. Verification of Rights

The Verification of Rights security function is concerned with the actual enforcement of access restrictions and other rights matters.

> The purpose of this security function is to permit subjects to exercise only the rights they have been granted. A security policy can specify:
>
> - which actions require a verification to be performed,
> - what should be done if an action is attempted without possessing the appropriate right, and
> - what exceptions shall exist to rights verification, and under what conditions.

On first glance, it would appear that all attempts at exercising rights should require a verification action. However, the inference here is that a security policy may specify which attempts need verifying. Two interpretations of this seem feasible.

First, attempts at exercising rights may be classified according to the level at which attempts are made. This is related to the particular set of objects and subjects which are considered subject to the administration of rights. For example, if “file” objects are protected, verification could be performed on file opens only, or on individual reads and writes. If smaller-grained objects, such as database records, were individually protected, a different set of considerations apply. If protection is enforced on memory segments, or protected variables, verification may conceivably be performed on nearly every machine instruction.

Second, rights verification may be implemented in two (or more) phases, as in the creation of an unforgeable ticket on a rights request, and the inspection of this ticket on subsequent attempts to exercise this right. Different forms (and different strengths) of verification are thus performed at each phase.

This discussion is a good example of the type of information presented in the Green Book. There is little in-depth consideration of any security issue,