COMPARATIVE STUDY OF REAL ESTATE LAW AND PROTECTION SYSTEMS

by

Bernard Louis Peuto

Memorandum No. ERL-M439

May 14, 1974

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Research supported by the National Science Foundation Grant GJ-

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Martin Graham, for guiding the research reported in this dissertation, and for giving me moral and financial support during my years in the Computer Science Department at Berkeley.

I would also like to thank the other members of my dissertation committee, Butler W. Lampson, and Professor Austin C. Hoggatt for reviewing and commenting upon the drafts of this dissertation.

When my morale was at its lowest point, Charlie Bass, Jim Gray, Jim Morris, and Dave Redell helped me, I thank these friends. To Paul McJones and many other friends that have read and commented upon my drafts, thank you.

Harold Shattuck and Kornel Spiro, my managers at Amdahl Corporation provided me with a payed leave of absence to finish this dissertation. I am very grateful to them.

Edith Purser typed my drafts and despite my deadlines and my poor spelling, always managed to be smiling and friendly.

Research supported by National Science Foundation Grant GJ - 31772.

COMPARATIVE STUDY OF REAL ESTATE LAW AND PROTECTION SYSTEMS

by

Bernard Louis Peuto

ABSTRACT

There is a great need to describe a set of philosophical concepts and principles applicable to protection. This dissertation attempts to do such a description by making a comparison between protection systems and Real Estate Law. The choice of Real Estate Law is justified by a similar goal, a similar structure, and a common basic entity: the right.

This work is divided in four parts which compare together protection concepts and legal concepts. The chapter on taxonomy of protection introduces the goals of protection and of the Real Estate Laws. It also defines the concepts used later: property, ownership, right, estate and principal, computation, directory, object, etc... The chapter on structures to support protection describes the parallel between objects and property, principals and owners, directories and estates. But because of differences in enforcement policies it also introduces the domain. The chapter on transfers describes three types of transfers: simple transfer, transfer with reversion and transfer with intermediary. Finally, the chapter

on pragmatic limits mainly describes the problem of the match between objects and reality.

The format used is to present the Real Estate Laws first and then to discuss the protection concepts in light of the legal viewpoint and current implementations.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

When we consider the many cases of protection, we can easily conclude that protection is a relative notion dependent on human judgement. This characterization of protection is commonly accepted. This should surprise us, after we notice the large number of works concentrating on practical solutions to narrow implementation problems and the quasi-absence of work dealing with the philosophy of protection.

This dissertation attempts to restore some balance by describing concepts and principles useful to protection. The work takes the form of a comparison between protection systems and California Real Estate Law. The recourse to another field to find useful protection principles is needed because of the neglect of this subject by most of the computer protection papers.

One has only to read the abundant literature which confuses security, privacy and protection in various trade journals to find that its most recent proponents may not be its best defenders. The protection problem has been deluged with simplistic solutions and inadequate definitions of its scope. We don't share the view of protection that "Ideally, it should be possible for the Democrats and the Republicans to store their confidential records on an ITT

owned computer located in the Watergate building and operated by a member of the SDS... [without worrying]", [Weinstock 73]. It is symptomatic of much computer system work that most protection problems have been solved without understanding protection or the problem.

But we are frustrated, for we too have made very little progress in providing a philosophy of protection and a description of what protection is about. Only now can we fully appreciate the shrewdness of solving a problem without knowing its essence; most of the hard issues can be avoided. We are left with the quote by R.M. Needham, "protection is not an after thought". We certainly endorse such a battlecry.

1.1 Selection of dissertation goals

We view protection in an operating system as consisting of three levels: 1) the philosophy, 2) the model, and 3) the implementation of the model. These levels are somewhat inter-dependent, but we can hope to achieve a significant degree of independence in their description.

By philosophy of protection we mean the set of motivating concepts and principles specific to our application. These concepts characterize it in very general terms; for example, there exists a problem of conflicting rights on scarce resources, of revocating privileges, of mutual suspicion, etc...

At the model level we try to isolate a number of principles and use them to express our philosophy. For example, we adopt the principle that all objects have at least one owner, that transactions are described in terms of transfer of rights, etc....

The implementation involves the specification of the primitives and objects used to implement the features and transactions of the model.

We feel it necessary to describe a philosophy of protection before attempting to suggest an acceptable model of protection, since we were encouraged by our finding that a large number of Real Estate Law principles and concepts apply to protection. But without any limit on the scope of the study, such a goal would probably be too ambitious, despite the help of the Real Estate Laws. The next paragraphs describe the pragmatic limits accepted in this dissertation.

1.2 Definitions of protection

It seems logical to start the description of a philosophy of protection with a definition of protection. This definition will help to mark the boundaries of the problem in order to avoid confusion and conflicts.

We have already given one author's version of an ideal protection goal. In fact, few authors attempt to provide definitions of protection. Lampson [Cosine 71, see also Lampson 71]

defines protection as "a general term for all the mechanisms which control the access of a process to other objects of the system". Although there are many diverse implementations of "processes", the notion is generally understood; but "objects" are neither well understood nor is their use well documented by systems which have been implemented using them. We feel uneasy about a description of protection which reduce the problem to a question of access control. Schroeder and Saltzer [Schroeder 72a] state that "the role of protection in a computer utility is to control user interaction -- guaranteeing total user separation when desired, allowing unrestricted user cooperation when desired, and providing as many intermediate degrees of control as will be useful". If we compare these two definitions, we must admit that although the second seems quite general and less restrictive than the first, it does not help us because in its generality, absolutely no information is given on how to implement it. What is user interaction? What is user separation? etc....

These two definitions are representative of the definitions given by others, often with only minor changes.

If we want to provide, at this stage of the research, a definition of protection that is general, independent of implementation decision and even independent of model concepts, we must conclude that the best source is an American language dictionary.

Our view of protection is contained in this modification of the Webster's dictionary definition of protection:

> "Protection is a mechanism aimed at shielding from that which would encroach, detrimentally affect, harm or destroy. It must not eradicate cooperation. The purpose of this mechanism is also to build up confidence and its dependability and inviolability cannot be questioned".

We like this definition because some important points are made; in particular, protection is tied to the notion of cooperation and to the relative notion of confidence. But we cannot be blind to its defects. Although the terms used have well recognized meaning in English, one would be hard pressed to explain with precision what an "encroachment" is in a computer operating system. Even if deliberate attempts were made to replace this language by some more acceptable operating system related language, this definition is still not very useful. We shall attempt to explain why.

Many authors have discussed the large variety of protection mechanisms, often unrelated, that have been used in operating systems [Lampson 71, Saltzer 73, Graham GS 72]. Some of them are historical, it so happened that different mechanisms were implemented at different times, but there is no inherent need to do so. Most of these mechanisms are an attempt to provide for the very large number of possible cases in which protection is required. The aim of a definition is to clarify a problem. Given the present

understanding of protection, it does not seem that a few vague sentences tied together as a "definition" will achieve this aim with any real degree of usefulness. Any attempt to provide such a definition of protection and work down to more precise levels is almost sure to miss the point.

The appropriate goal, at this stage, is to cast some light on the multiplicy of types of protection by classifying the common characteristics of the various protection viewpoints. The end result of this process will be a model of protection: a set of abstractions and a list of relations between them. But, in contrast, we are trying to provide constructs which capture the "macro-transactions" of the users, which should be reflected in the behavior of the protection mechanism. When system designers implement policies, these constructs will provide them with the necessary benchmarks or goals. Whereas, the previous researchers have, quite often, limited themselves to the description of a set of primitives, necessary to implement the abstractions and relations of a protection model, with no more than a casual interest for the problems of the user.

1.3 More realistic goals

In this paragraph and in many other places in the dissertation, we often mention operating systems independently of protection systems. We feel very strongly that in a good design, protection is inextricably interwoven with the operating system. But,

historically, this has not always been the case. The knowledge of protection has often been narrower than the knowledge of other important parts of operating system design, such as virtual memory, process implementation, etc... This distinction is useful when we want to highlight the influence of the protection requirements on operating system design.

We support the statement made in the Cosine report [Cosine 71] that a good design methodology is very important for the success of an operating system implementation. If we follow the report's classification, most models of protection have been devised and implemented using the level approach [Dijkstra 68], or the nucleus-extension approach [Hansen 70, Wulf 73]. This dissertation suggests the need for a top-down approach. No method alone is going to solve the design problem of a protection system, but our emphasis is a reply to what, we think, have been unconvincing results derived from the application of bottom-up methods (level approach and nucleus-extension approach).

Some confusion arises because these methods have been successfully applied to operating systems. The methods work with operating systems because we now have a substantial body of practical knowledge about them. We are able to distinguish at least six or seven primary aspects of operating systems [Cosine 71]. In most cases we can define some abstractions, and relate the users needs and problems to problems solvable with these abstractions. An implementer may then implement some form of the abstractions, and the user may model his

application in these terms because it is possible to describe most practical problems in terms of these abstractions.

Such is not the case with protection. We have little information about practical problems, let alone about how to describe them in terms of abstractions. To attempt to define a set of primitives, independent of protection policies, is bound to give little relief to the implementer. We maintain that most of the papers trying to describe these primitives have added little to Dennis and Van Horn's description of 8 years ago, [Dennis 66]. Improvement will come more from describing policy-dependent primitives than from adding yet another set of "meta-instructions".

A useful philosophy of protection should provide the user with practical guidance in implementing a protection system. We maintain that this goal will not be achieved by describing another set of meta-instructions. Rather, we need to provide for a classification of the numerous protection viewpoints, using a description of protected transactions, and their resulting abstractions. To succeed in this task we use the California Real Estate Laws.

A first design principle is to construct the description of a set of transactions. The transactions used are macro-operations representative of user needs. They could be defined as sequences of primitive actions tied together for some reason. The sale of property is a good example of such a transaction. The word

"representative" was chosen because no attempt is made to describe a set of canonical transactions to which all possible transactions could be reduced. The description of the transactions is used to introduce the various protection viewpoints. Realism is what is needed.

To allow for a comprehensive description of these transactions, one must create some abstractions. The choice of these abstractions is influenced in part by the set of objects these transactions are applied to, and in part by the factors that influence the execution of the transaction.

Another design principle is to give importance to the conse quences of an action. This affects the enforcement policies: the actions with the greatest consequences are checked most thoroughly. What is done after a violation is more drastic according to the consequences. The time has come to justify the choice of Real Estate Laws.

1.4 Usefulness of Real Estate Laws

The history of protection of computer data is very short. The long history of the protection of property rights provides enduring principles which can be applied to computer data. The notions of property and ownership pervade our society and our long cultural contact with legal concepts may explain why in an increasing number of protection papers, terms like "owner", "right", "legal" etc... are mentioned.

We must recognize that a significant part of the challenge of protection is the human interface. It is not sufficient to create a functional protection system. The system will succeed only if the mechanisms provided appeal to the user, as illustrated by Saltzer's description of many subtle issues encountered in Multics [Saltzer 73]. One cannot realistically sever the protection problem from its human dimension. What protection is or should be is largely a matter of opinion. Protection involves some regulation of human activity. Because of this fact, the law ultimately will come into play. We cannot choose an abstract solution to protection and work on it as if it solved the real problem because the business and legal requirements of most users will constantly remind us of the gap between our solution and reality.

This view made a study of existing legal principles a potentially interesting problem. But our research was inspired by a remark made by M.D. Schroeder: in order to appreciate the problem of mutually suspicious cooperating processes, one could think of two persons cooperating for a transaction like a buyer and a seller [Schroeder 72c]. As Martin Graham pointed out, this analogy of a buyer - seller relationship is especially attractive in the case of a real estate transaction. In California, for example, the transaction may involve an escrow and a title insurance company. They are both neutral, have rigidly defined powers for the transaction and are intended to provide protection for both the buyer

and the seller.

Most models of protection have been defined using protection principles selected because they meet specific requirements, such as ease of assertion, modularity, extendability, programming generality, etc. To use the concepts of Real Estate Laws is a variation on the same idea.

Three important characteristics of the law complete our list of arguments. First, Real Estate Laws have evolved because of the existence of land. The same piece of land can be used concurrently by many people for different purposes. It becomes the object of rights so extensive that individual enumeration of all rights is impossible, but instead they are divided in general classes of rights. It becomes apparent that the necessity 1) for ascertaining each right, 2) for preventing one from usurping another and 3) for the identification of the physical object of the rights, calls for a large body of rules. Real Estate Law is the set of these rules, based upon the principles of equity and justice and reduced to a specific language.

Second, the legal system is a construction of rules and procedures, all of them aimed at defining and limiting various aspects of human behavior. But to enhance its effectiveness, law has evolved through the ages and now comprises a limited number of concepts, aimed at regulating a multitude of man-created phenomena. We call the concepts the <u>legal framework</u>. And although it can happen, this framework is not often altered because of newly

introduced phenomena.

Finally, despite the layman's expectation, property in Real Estate Laws does not refer to the thing itself that is owned, but in a strict legal sense, refers to "the rights or interests that a person has on a thing", often referred to as a "bundle of rights". This "bundle of rights" is the "exclusive right of a person to own, possess, use and dispose of a determinate thing, either real property or personal property, consistent with the law" [Bowman 70, p 23].

For the purpose of protection the Real Estate Laws offer the following advantage. They are a transactional model whose goal is to regulate the transfer of property and ownership. This is achieved by providing abstract descriptions of the resources and rules to manipulate them with special emphases on the consequences of actions. The knowledge of how to describe resources and their transfer with special attention to consequences is a first step toward assuring protection.

1.5 Limitations of study

There is enough generality in the concept and similarity in the situations to provide for a useful comparison between Real Estate Laws and protection. But, for example, the resource characteristics and the consequences of action are bound to be different. Thus, we are not seeking a perfect one-to-one correspondence, such similarity would be miraculous. If, somehow, this

even existed, it would, among other things, imply a thorough knowledge of the law; a preposterous idea considering our overall ignorance of legal matters. Nor do we have a vested interest in our legal model being appropriate. We are more interested in studying the similarities and differences than in proving that the Real Estate Laws are the only or even the best way to approach the problem of protection.

We do not feel that the legal origin of our suggestions should be minimized. The legal field is complex enough to allow different assumptions of what is useful, and we accept that our description reveals our explicit or implicit assumptions. But we must admit that the number of legal concepts eliminated from our description have been very large. First, we were obliged to make a distinction between the set of laws themselves, often dependent on specific human realities, and the legal framework, very general and independent of specific applications. We were also obliged to limit the number of points of view worthy of comparison because there is something wrong with describing the same thing over and over, if no implementation is envisioned to determine the limits of each point of view. Finally, it is sometimes difficult to accept the results of a comparison. Static and dynamic capabilities are mentioned by M.D. Schroeder [Schroeder 72b]. Capabilities useable for a finite number of times have also been mentioned. Our comparisons suggest the need for future capabilities (that is, capabilities valid only in the future).

The final limitation of our comparison deals with the importance of physical resources in the legal model and thus the possibility that the comparison will deal primarily with the aspects of protection tied to resource management. We do not think that this limitation is as acute as it has sometimes been with other work. Whereas, many models are based on a description of the physical reality of resources first and then on primitive operations to manipulate them, the legal model is based on a description of transactions first. These transactions introduce principles and types of interests that seem to apply as well to physical resource as to "information" as long as both can be described in terms of bundles of rights. This central point will be dealt with in the coming chapters.

1.6 Plan of dissertation

Our principle source of information for the legal concepts is a book by A.G. Bowman: "Real Estate Law in California", [Bowman 70]. To limit the problem of comparing the vast field of Real Estate Law with protection, we have selected four areas of study: taxonomy of protection, structures to support protection, transfers, and pragmatic limits. These four subjects correspond to four chapters of the dissertation. The last chapter is the conclusion. In each of these four chapters, the Real Estate Law

concepts selected for the comparison are described first, then the protection concepts and solutions are introduced and compared with the legal concepts.

Chapter 2, on taxonomy of protection, introduces the Real Estate Law and protection. It describes the basic concepts of the legal system: property, ownership, right, estate, instrument. This is followed by a description of the functioning principles of the Real Estate Laws. The second section of Chapter 2 first places protection in the context of accepted operating system principles before describing its two main aspects: protection of resources and protection of information. Then it describes the most common method of protection and implementation: privilege restriction and access checking.

Chapter 3 is on the structures to support protection. It first describes the elements of ownership: classes of property, methods of ownership, estates, rights of the owner, easements and liens. Then it compares objects and property, principals and owners, directories and estates. It introduces the domain as the enforcement structure of the directory.

Chapter 4 on transfers describes deeds and typical legal transactions: sale of property, lease, agencies and escrows. It classifies three types of transactions which are described in the protection section: simple transfer, transfer with reversion, and transfer with intermediary.

Chapter 5 on pragmatic limits introduces a few miscellaneous Real Estate Law concepts: property conflicts, land description, recording, homestead, and title insurance. Section two concentrates on the match between objects and reality in the current protection schemes.

The conclusion summarizes the concepts introduced for the classification of protection and presents once again the reasons why Real Estate Laws are a good model; they deal with failures and consequences of failures.

CHAPTER 2

TAXONOMY OF PROTECTION

2.1 Taxonomy of Real Estate Laws

2.1.1 Goals of Real Estate Laws

Some of the goals of the Real Estate Law system were introduced in 1.4 to justify its choice. We can summarize them in a form closer to the goals and components of protection described below. The Real Estate Law system attempts to regulate the conflicts on land, a scarce resource. It does so by regulating land transactions in order to minimize undesirable consequences and thus achieves some social order. The transactions which are regulated are quite representative because owners are induced to use them in order to benefit from the legal protection. Ownership and its abstract element, the right, are the major concepts used in describing these transactions. The important problem of identification of the objects of rights is solved by creating abstract descriptions of the physical objects that are regulated. It is then easier to refer to them and to describe the rules that apply to them.

For our purposes, it is quite accurate to state that: Real Estate Laws deal primarily with the creation, transmission, recording, exercise, etc... , of rights.

2.1.2 Functioning Principles of the Real Estate Law

The California Real Estate Law system is based on Old English Common Law. The Common Law system is based on legislation created when cases and conflicts arise. This tends to show more clearly the relationships between the law and the problems the law attempts to solve, a characteristic that worked in our favor.

The set of concepts that make up what we call the legal framework is fairly independent of any application and has been quite stable over the years. These concepts represent the abstractions used to describe the real objects or events the legal system wants to regulate.

The description given of the Real Estate Law system in later chapters involves quite often, three stages:

1) A description of the abstractions used in lieu of the real objects. For example, a piece of land is replaced by its description in metes and bounds and its title chain. For example, once rights or interests have been introduced, property is defined as a bundle of rights. An owner is defined as a fictional person since he may be one or more human beings or a corporation.

The main thrust of this stage is <u>not</u> the description of a small number of abstractions: it would take only a few pages. It is the case by case matching between the abstract entities and reality, as spelled out by the legal precedents. For example, is a house built on a piece of land, part of that land, and thus immovable real property,or is it a piece of personal property that could be

used to pay creditors?

2) Using these abstractions then the law proceeds by describing the elements necessary for a legal transaction. The owner has some freedom in deciding the terms of a contract. But often, to be valid, some a priori conditions must be met. The ultimate validity is determined only by a posteriori checks. In order to protect himself an owner is encouraged but not forced to follow standard procedures.

3) The last stage describes in detail what is done if a conflict arises. For example, an important item is missing, or the contract is ambiguous or has some illegal clauses. This last stage is so important that one can consider without exaggeration that all the abstractions, all the rules, have been created and organized to solve the problem of what to do when a conflict arises and how to solve it according to the importance of the possible consequences.

The abstractions of the real objects are introduced first to facilitate the description of the legal system. They reflect the factors that affect the various transactions, but they are not an abstract representation of the physical characteristics of an object. For example, the legal system distinguishes between the earth that constitutes a piece of land and the minerals in it, because earth is an immovable part of the real property and mineral ore is a movable piece of personal property. Mineral rights can be sold independently of property

rights and the minerals can be taken from the property without the property owner loosing his title to the rest of the property. The distinction, quite surprising for a layman, is not based on chemical composition, but on the different transactions that affect the physical object.

The Real Estate Laws are not isolated. They are part of the legal structure from the Constitution of the United States to the Court decisions. And all the elements of this structure provide sources for Real Estate Laws. In particular, general legal doctrines are used throughout, and specific rules of proceedings are used for trials and for the search of evidence.

2.1.3 Real Estate Law concepts

Property is the thing of which there may be ownership. Ownership is defined as "the right of one or more persons to possess and use property to the exclusion of others" [Bowman 70,p 22]. The legal system distinguishes between two classes of property: real or immovable and personal or movable. The numerous consequences of this distinction will be described later. It is enough now to give an example of real property: a piece of land.

A right is defined to be as in English: "something (as a power, a condition of existence, or a possession) to which one is entitled, by nature, by the principles of morality, by grant, by the laws of the land, or by purchase" [Webster 68, p 699]. Often the word interest is used to mean the same notion referred to as a right.

The reader should be cautioned that in the legal system having a right does not always imply an absolute or exclusive power.

An estate is the degree, quantity, nature and extent of the interest a person has in real property. At this point we should indicate that the law distinguishes between possessory interests and non-possessory interests. An example of non-possessory interest is an easement since it describes an interest in the land of another person.

Most of the legal transactions, to be valid, require the existence of an instrument, a paper signed and delivered expressing the transaction. A deed is an instrument used to transfer ownership of land.

We have selected six elements for the classification of Real Estate Laws.

1) Nature and class of property. The three principal classes of rights that characterize an owner are defined, as well as the two classes of property: real and personal.

2) Property as a thing made out of land and the consequences of the unique characteristics of land. This element deals with the match between reality and the abstraction.

3) Class of ownership describes the consequences of having multiple owners.

4) Nature and classes of rights. Specific rights like easements and liens are introduced, as well as the encumbrances on property.

5) The estate is the abstraction that ties together properties, owners and rights. We introduce it.

6) Acquisition and transfers of property can be described now since most of the abstractions used in contracts are defined.

This classification is made for the purpose of our protection studies. It is not a reflection of the spirit of the law; it is not a real legal classification as could be found in any textbook.

2.2 Taxonomy of protection

Although this section deals with the classification of protection concepts, we are going to discuss at length operating systems concepts. The principle reason for this approach was discussed in paragraph 1.3. Protection is an integral part of operating systems, and it would be impossible to describe one without the other. But in each of the following paragraphs, we will, in fact, separate the operating systems concepts from our personal selection of protection concepts because the former are more universally recognized and thus more stable.

2.2.1 Goals of protection

The need for protection is increased by the goals and the typical implementation of an operating system. Many goals have been described for operating systems, [Cosine 71]. For the purpose of this dissertation we insist on the following:

An operating system creates an environment for efficient program

execution. This goal is achieved mainly through the management of computer system resources. To a lesser extent, when information is not tied to a resource, this goal requires the management of information. Our favored inplementation method uses the definition (and implementation) of extended machines.

The resources are the physical components of the computer system shared by the users: i.e., the central memory, processors, channels, mass storage devices and other types of pheripherals, or their logical equivalent.

The problems associated with information deal more with how to control its dissemination, how to protect it in the same sense that one protects a trade secret in industry; once it is known the total damage is done.

Whatever goals one tends to favor,there are some common characteristics: 1) concurrency, 2) sharing of information, 3) long term storage, 4) non determinacy, 5) sharing of resources, and 6) modularity, [Cosine 71].

We should note that, in general, these characteristics result in a greater importance given to problems involving protection.

Our description of the goals of an operating system has already introduced the two aspects of protection most frequently mentioned: protection of resources and protection of information.

The computer system has a finite set of resources which the principals and their computations can use. A principal is the

fictional personification to which resource usage is charged. The computations can be viewed as having resource requirements, since these requirements must be served by resources shared by a large group of computations, conflicts arise. The first goal of protection is to provide a set of rules to regulate these conflicting requirements.

Quite often individual pieces of information have been artificially represented as elements (indistinguishable for protection purposes) of a larger object. These objects, the smallest protectable entities, have often been pieces of a physical or logical resource. Thus, quite often, protection has been reduced to the regulation of conflicting accesses to resources.

In the example of central memory, the management of this resource involves deciding how the computations are going to share it. Protection, for example, involves checking that no computation accesses part of central memory not alloted to it.

The protection of information involves other issues. Let us suppose that we have already implemented a resource protection system. Denying access to a piece of information is then relatively easy since we may deny access, for example, to the piece of resource where it is stored. But if the information is shared, we also want to be able to control its dissemination and its usage among those that can access it, notions that are not connected, at least directly, with resources control.

There is a recent trend toward avoiding the subject of protection of resources in favor of protection of information since quite satisfactory solutions have been found for the former, but very little is known of the latter [Jones 73, Lampson 73]. We support this trend which enlarges substantially the scope of protection.

At first it would seem that the notion of protection of information is quite close to the notion of privacy, which too deals with the control of the dissemination of information. But privacy involves social issues whereas protection is more a matter of technical issues. The principle difference between our point of view and previously expressed opinions is that we do not contend that a technical issue is an issue in which a deterministic and mechanistic solution exists. This is well illustrated by the confinement problem as described by Lampson; in some cases it seems to be an unsolvable problem [Lampson 73].

2.2.2 Protection concepts

In the remainder of this dissertation we will use the following set of well accepted operating system abstractions. The principal is the abstraction used for a user. A user is, in non-technical English, the person or group using the computer system's services and submitting programs. Following the ideas of Dennis and Van Horn, we define a principal to be the fictional person to which resource usage is charged [Dennis 66]. A principal

owns and is also charged for the resources used by his retained objects. The retained objects are needed because of the long-term storage facilities of most operating systems. Even when no program is in execution,a principal has data, program modules, and directories stored in the system. They are all retained objects.

The abstraction of a program is a set of procedures. A procedure, when in execution consists of 1) instructions, 2) an activation record made up of the information needed to execute a call to the procedure, and 3) a non-local environment [Cosine 71].

When a program is in execution,we introduce another abstraction to characterize it: a computation. The total environment of a computation is the data structures and procedures currently accessible.

Concurrent programming results in the introduction of processes. In order to achieve concurrent programming we implement sequential processes interacting on well defined events. A processor is a resource needed for the execution of a process but is not associated permanently with it. This is in contrast with a procedure where the notion of availability or unavailability of a processor does not exist.

A directory is used to associate a name called a path-name and the retained object this name describes. Directories are often used to solve the name management problems which result from the controlled sharing of access to data bases and procedures. This problem will be studied in Chapter 3.

The abstractions introduced in the coming paragraphs have been used to solve protection problems. They are followed by a description of abstractions derived from the legal system.

Domains are a protection entity which can be viewed as a simple extension of concepts developed above. The notion of total environment was defined as what a computation needed to access, and thus could access at an instant in time. But the accessing mechanisms provided by the hardware or the software of a computer system usually allow access to a larger number of procedures and data structures than are needed. A domain reduces, for protection purposes, what is accessed by a computation to its total environment. The word domain is used either in place of the notion of total environment, or, as was done above, to describe an enforcement structure used to reduce the accesses of a computation to its total environment.

Like Dennis and Van Horn [Dennis 66] we have associated a domain with a computation. Some suggestions have been made to associate a domain with a process [Lampson 69b]. These differences arise because two views exist of the relation between processes and computations. In the first case, because of parallelism, one can have many processes per computation, and the computation becomes one unit of protection. In the second case, because processes are expensive to implement, only one process per computation is allowed, and the notion of computation is replaced by the notion of a process.

When one implements a domain as an enforcement structure, one feels the need to describe it in terms of the set of "powers" available to the computation. The word "capability" has been used by Dennis and Van Horn to represent this notion [Dennis 66]. The list of capabilities of a computation, or its C-list, defines the domain associated with the computation. In this context we have introduced a capability only as a proof of a right of access. But capabilities have come to be a rather precise notion: an unforgeable descriptor of an access right naming an object for which the access is allowed as well as a prima facie proof of the right of access [Lampson 69b, Sturgis 74 , Fabry 68].

The last abstraction introduced now is that of an object. "Objects" have been defined as "the things in the system which have to be protected. Typical objects in existing systems are processes, domains, files, segments, and terminals. The question of what to designate as objects is a matter of convention, to be determined by the protection requirements of each system" [Cosine 71]. This is rather vague. A little more insight will be given with the description of an object oriented system in 2.2.3. But this is an area where much work needs to be done.

The problem arises of naming protected objects, we will study it in the coming chapters. We suggest the following abstractions be added:

Property is the thing of which there may be ownership.

An owner is the fictional personification which has the right to possess and use property to the exclusion of others. The owner is characterized by three classes of rights: the rights of disposition, exclusion and use.

Access to an object implies the possession of some of the rights of use. We must insist that "use" does not imply unlimited use.

The relation between property, owner, rights, accesses and objects, principals, capabilities, as well as domains will be described in the coming chapters.

2.2.3 Principles of protection implementation.

Most of the operating systems for third generation computers have been designed using modular implementation of operating system functions. Large amounts of data are accessible by all modules. This is similar to the first method described, but not recommended, by Parnas for decomposing programs into modules [Parnas 72]. Instead he recommends that modules be designed so that the amount of information shared and transmitted is minimized. The criterion used is that of "information hiding" which implies, in particular, that modules are also designed to isolate the information transmitted from its representation.

Practical implementations of this second design method are illustrated by the level of abstractions described by Dijkstra [Dijkstra 68], or the object-oriented system used in CAL-TSS

[Lampson 69a, Sturgis 74]. The computer system is viewed as a limited set of object types (say 10) on which a large number of operations are applied (say 100). The objects can only be manipulated by the given set of operations. The operations are the shield provided between the information and its representation. The objects provide a structure for the data which allows one to access its content with a minimal dependency on its representation.

The advantages of such a design methodology have often been described [Parnas 72]:

1) Changes in implementation decisions are facilitated.
2) The system appears simpler and easier to verify.
3) Isolation from the physical representation allows one to work at levels of abstraction closer and closer to the user needs, unencumbered by hardware idiosyncrasies.

At least one disadvantage is known: some apparently simple operating system actions trigger large numbers of basic operation calls, resulting in a large amount of overhead [Gray 72].

In the BCC 500 system there are seven types of objects [Lampson 69b, BCC 69]: files, pages of memory, processes, domains, interrupt calls, terminals, access keys. Operations on these objects involve creation, deletion, copy, etc. In the ECS system of CAL-TSS [Lampson 69a, Sturgis 74], there are also seven object types, only six are similar to objects of the BCC 500. There is no object in the ECS system to represent a terminal, this is because the ECS system is only one layer of

CAL-TSS and such an object is implemented at a higher level. The seventh object of CAL-TSS is an operation. This allows one to manipulate the operations used on objects with the same kind of safeguards and restrictions that are used to manipulate objects.

One basic concept of the current protection systems is well illustrated by Lampson's access matrix model [Lampson 71]. The protection context is made up of domains, objects and accesses as defined in 2.2.2. Enforcing protection is a matter of allowing (or disallowing) a requested access to an object by a domain, if it is known that the domain has (or has not) a right to this access. The protection context is described by a set of triples which can be represented as a matrix.

There is an even more basic principle which emerges from this example. We call it the checker-enforcer principle. There exists some notion of what the reality should be; there exists some notion of what the reality is. A checker detects any discrepancies between them and signals an enforcer which takes appropriate actions. Some form of this mechanism will certainly exist in a protection system, just as it exists in a large number of other human systems, including the law. When we talk about some notion of the reality, we imply that one is able to represent an important parameter in a form acceptable by a checker.

A simple example of this principle is the lock and key mechanism. The lock is what the reality should be, the key is what the reality is. Each is represented by a value so a checker

can be built which compares their values. The enforcer allows access to the memory block or causes a trap according to the result of the match detected by the checker.

In the case of the access matrix model, the checker informed of the identity of the domain of the object to which access is requested, and of the type of access; checks in the matrix what the reality should be, and signals the enforcer to allow the access or not.

This principle seems easier to use with an object oriented operating system which provides a set of objects and operations on them. This fact justifies their popularity and motivates their description in this dissertation.

There are two basic mechanisms used in implementing protection: privileged restriction and access checking. These two protection mechanisms are based on a permission rather than exclusion philosophy; the default is lack of access or privilege.

Privileged restriction deals with the mechanisms used to provide a computation with the minimum possible set of powers or privileges or actions needed to perform its task. You cannot misuse what you cannot use. Connected to this design principle are some of the justifications for associating a sphere of protection or a domain with a computation. The domain reflects all the privileges available to the computation. It should be implemented in such a way that the computation has no way to exercise any privilege other than those described in its domain. The principle of 'need to know' described by R.M. Graham [Graham RM 68] is also

connected with this. Very often the control over the "amount of privilege" given to a computation is done through the device of a hierarchy of decreasingly powerful computation. If one does not need all the privileges allocated to a computation to perform a subtask, one creates a subcomputation (for example, an inferior sphere of protection [Dennis 66]). The simplest example of this principle has been implemented in hardware as the user monitor mode.

Access checking is the mechanism used to check and enforce the variety of possible accesses between a set of domains and a set of objects. The access matrix model is an example of such a mechanism.

A requirement often mentioned with an implementation of protection deals with program generality. Protection should be available without special programming from the calling computation.

A protection system according to the points of view described above:

1) creates some environment to describe the resources available to the computation. This environment is used to implement some form of the privilege restriction policy.

2) Finds ways to identify a computation, and to authorize and establish access to other objects.

3) Insures that a computation stays in its proper environment throughout its life.

One requirement is to allow different types of access to different computations accessing the same object. As a matter

of fact we feel this computation-object relation is the best definition of an access in the sense used in the current implementations. Two representations of accesses have been used: capabilities and access lists. The duality between capabilities and access lists is described by Lampson [Lampson 71]. There is no need to elaborate these points here.

The capability scheme solves the requirements of privilege restriction and access control since having a capability is the proof of allowed access. Unfortunately, when transfer of capabilities is also allowed, this simplification causes revocation of capability problems.

When access lists are used, protection has often been enforced by appending the access list to the directory of retained objects. This allows the privilege restriction to be enforced at the time objects are called from the filing system. Access checking is then enforced by using some hardware mechanism applied to the operating system objects representing the retained objects at the time of execution. Memory protection is often used at this last stage. If, in particular, virtual memory is used, a simple mechanism exists to achieve restriction to the privilege of a computation. Objects that the computation cannot use are not in its address space and cannot be named.

The principles of protection implementation described above will be analysed in the coming chapters. Finally, we would like to introduce some principles from the law that seem to be useful.

Ownership is the most important of these principles. It implies that all resources of the system have at least one owner. The owner owns a resource because of a specific set of rights he has on it. The resource is not defined as a set of physical or logical characteristics, but as a bundle of rights. To avoid the potential confusion that would be created by the usage of the word resource, we talk of property. Property is a bundle of rights, it is anything that can be owned. In particular, it can be a resource or a piece of information.

In an object oriented operating system, protection enforcement is very dependent on the objects defined by the system. This dependency exists because, in part, protection is achieved by matching the objects real characteristics with what they should be. It also exists because control on the operations performed on the objects is usually achieved by checking for the existence of the requested operation in a table of privileges.

In contrast, the legal system is more interested in the consequences for the owners and society of the changes that affect property in a transaction than it is in the physical characteristics of property. Because of the influence of society, the legal system has some notion of cost. It tends to allow simpler and less costly checking even if this implies possible violations.

A typical transaction in the legal system is a reciprocal exchange of rights. Thus, the simple minded checker-enforcer method of protection usually does not apply.

CHAPTER 3

STRUCTURES TO SUPPORT PROTECTION

3.1 Structure of ownership *

In Chapter 2 we introduced some of the goals of the Real Estate Laws, and we defined property, ownership, rights, etc... In this chapter we extend these notions by describing the components of property, the types of ownership, the classes of rights,and the interaction between them with the notion of an estate.

In describing this structure we are interested in a specific point of view: how an owner, by creating a legal structure for his rights, protects himself. Protection results from such a structure because 1) the consequences of the choice of specific elements are known and 2) the intentions of the parties are spelled out unambiguously.

The reader must remember that an owner may not have in reality full choice, but this is immaterial for our comparison because we are more interested in the variety of structures than in some of the specific usage made of them by human beings. A simple example of such a limited choice arises when ownership is acquired by succession.

---

* The intent of this dissertation is not to write an original study on Real Estate Laws. Consequently the legal sections use large excerpts from Bowman's book [Bowman 70], often slightly abridged or modified for our purpose.

If an owner wants to protect himself in the legal system he must:

1) choose the type of ownership best fitting his need
2) be aware of the kind of encumbrance his property will suffer
3) draw up a legal contract for each major transformation of his estate
4) and finally draw up a will if his property is to be disposed of at his death.

The last two points envolve transfer and termination and will be studied later.

In the choice of ownership one either deals with the various types of owner, or the various types of estate. With the rights of ownership we also introduce easements and liens. We must first describe the two classes of property.

3.1.1 Classes of property

There exists two classes of property: real or immovable property, and personal or movable property. Real property is defined as land, what is affixed to land, incidental or appurtenant to land, or immovable by law. Personal property is what is not real property. Technically,real property is not the land itself but the rights or interests that are estate in fee or for life. All lesser estates are personal property. Under certain conditions real property and personal property may be changed into another class of property.

We can give a quick idea of what are the elements of real property by using the following example: real property includes the ground, also what is affixed to the land and incidental to its use. The ground can be divided into what is above ground and underground, each part can be divided further and we could draw the graph of Figure 1. The figure illustrates our point by naming typical members of each category. These members are differentiated because of the conditions of their manipulation: a house needs to be built, a tree may have been there before, oil is a migratory substance whereas minerals are unambiguously located, etc.....

Notice, to illustrate a point made earlier, that minerals while in the ground are real property, but taken out they become personal property. Most of the members of our figure can be the objects of specific rights and owned separately. The minerals could be owned separately from the land. Many of the complex situations that arise from these subtle differences are beyond the level of this description.

The distinction between real and personal property is also important because state laws govern real property but personal property is governed by laws of the owner's domicile. The transfer of real property can only be made by an instrument in writing. The taxation is different for both types of property. There are many other consequences of lesser importance.

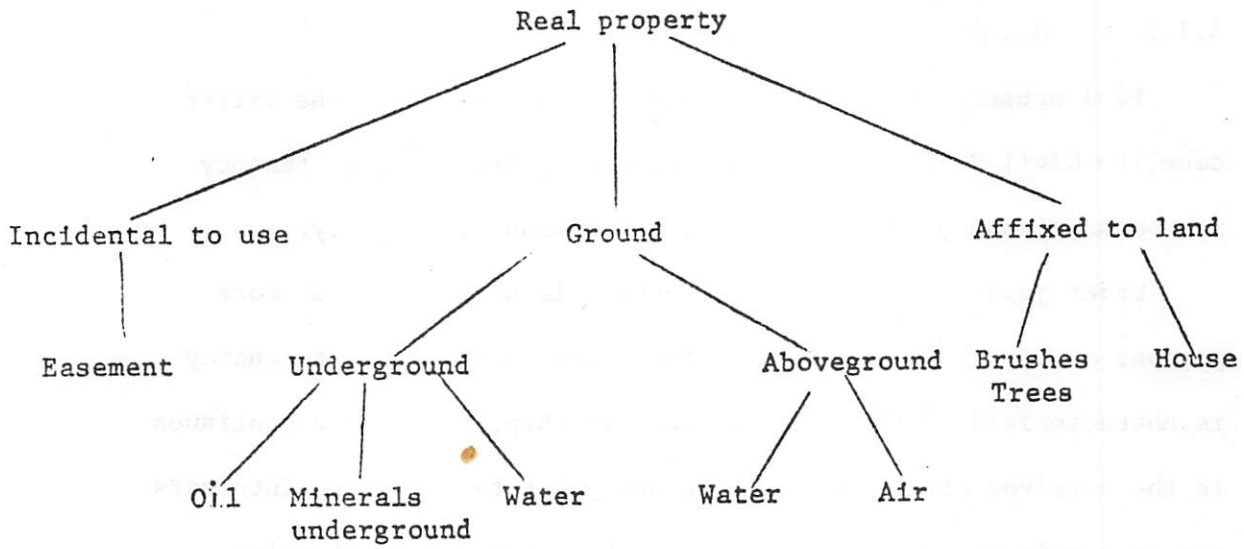
Real property
Incidental to use
Ground
Affixed to land
Easement
Underground
Aboveground
Brushes
Trees
House
Oil
Minerals
underground
Water
Water
Air

Figure 1.

3.1.2 Methods of ownership

Real property may be owned solely or jointly. In the latter case,the Civil Code distinguishes between joint tenancy, tenancy in common, tenancy in partnership, and community property.

Under joint tenancy a single estate is held by two or more persons as if they constituted a fictitious unity. Joint tenancy is characterized by the right of survivorship; the estate continues in the survivor after the death of one joint tenant whose interests are so terminated. Joint tenancy can be created only if unity of time, interest and possession exist. This is in order to maintain the requirement of the tenants being in a fictitious unity. Joint tenancy has several disadvantages dealing mainly with the fact that the entire estate is held by more than one person.

There is only one unity in the tenancy in common, that of possession. A cotenant owns undivided interests in quality and duration, coming from different conveyances at different times. There is no right of survivorship.

Community property is the property acquired by husband and wife, or either, during marriage, when not acquired as the separate property of either. Separate property consists mainly of property owned before marriage and that acquired afterwards by gifts, bequest, devise, or descent, and rents, issues and profits thereof. The main difficulty with this form of ownership is the determination of the type of property: separate or community.

Several presumptions are made by the law. Community property can be awarded in part in the case of divorce proceedings and can be made liable for debts of any spouse to varying degree.

A partnership is defined by the Corporation Code as "an association of two or more persons to carry on as co-owners a business for profit". Partners are liable for all firm debts, but the interests of a partner in the partnership are not subject to attachment. The interests of a partner's heirs are only in the profits and surpluses which are personal property.

Joint ventures are a somewhat simplified version of partnership.

3.1.3 Estates

An "estate" is the ownership interest that a person has in land. It can vary from absolute ownership or "fee simple absolute" to mere tolerated possession, called estate at sufferance. But not all interests create an estate: for example a mortgage does not create an estate.

The word "estate" is used to express the degree, quality, nature, duration, or extent of an interest in land. The main classification of estates is according to their duration of enjoyment: 1) estates of inheritance or perpetual estates, called estates in fee, 2) estates for life, 3) estates for years, 4) estates at will.

But other classifications are made: 1) according to quality, i.e., absolute or subject to contingencies, legal or equitable; 2) according to their time of enjoyment, i.e. immediate possession or possession at some time in the future; and 3) according to the number of owners. Finally, possessory as opposed to non-possessory means that the holder is presently in possession of the property. And freehold as opposed to nonfreehold means that the duration of the estate is unknown, that it is a bigger estate than an estate with a known limit.

3.1.4 Rights of owner

Three principal classes or rights characterize the complete authority of the owner of a piece of property: disposition, exclusion and use.

The right of disposition refers to the following points:

-- The owner may keep the land as long as he wishes (but subject to the government's power of domain).

-- He may sell or give away part or all of it.

-- The manner of disposition and terms of sale can be chosen freely.

-- He may devise it by will.

-- He may create a life estate or some lesser estate.

-- He may do nothing and upon his death the law of succession will be used.

The right of exclusion deals obviously with the right to exclude, but it is also used in the creation of smaller bundles of rights as when the owner

-- Gives to somebody the right to enter.

-- Lease or rent any portion of the property.

-- Grants an easement or a license.

Finally, the exclusive right of use is subject to the paramount right of the state to control the use of land. It is not a right for unlimited use and freedom of enjoyment.

3.1.5 Easements

An easement is a nonpossessory interest in the land of another person. The owner of an easement has a limited use of enjoyment of the land. Easements can be a benefit to the land or a burden. They are created by a conveyance, a written instrument transferring the title to an interest. There are two classes of easements: appurtenant and in gross.

An easement appurtenant is part of the land it is attached to and must be transferred with it. It is created when at least two tracts of land exist and one tract obtains the benefit and the other the burden of the easement. For example, the right to use a private road to reach one of the tracts.

An easement in gross does not benefit the land owned by the

easement holder. It is a personal right on somebody else's land. An example would be the right of an utility to install and maintain power lines on the land.

It is important to distinguish an easement from a license. A license is a personal, revocable, and nonassignable permission to enter upon the land of another person without possessing any interest.

3.1.6 Involuntary liens of property

A lien is defined as a charge imposed upon specific property by which it is made security for the performance of an act, usually the payment of money.

Liens on property may be created by voluntary act of the landowner, such as the execution of a mortgage, or may be created by operation of law, such as tax liens. When a lien is created on real property, the ownership of the property, remains in the landowner, subject to the right of the lienholder to force a sale of the property under a prescribed method to satisfy the performance of the obligation secured by the lien. Upon forced sale of the property, title then passes to the purchaser of the sale. A lien can be discharged at any time by satisfying the claim. Lien usually cease or become unenforcable after a lapse of time. Other things being equal, different liens on the same property have priority according to their time of creation, subject to the operation of the recording laws and subject to the effect of statutes according special priority.

3.2 Protection structures

This section attempts to show that the structures used by the legal system to define and protect an owner are similar to the structures used in protection. We conclude from this fact that the legal framework may be used to begin a classification of the multiple cases of protection.

The notions of property, owner, and estate, will be compared to the notions of object, principal, and directory. The comparison between the two fields is not as good when we discuss multiple owners and when we introduce the domain which is an enforcement structure for the estate.

3.2.1 Objects and property

The legal system attempts to regulate transactions between owners. Transactions deal with pieces of property. The characteristics of pieces of property are introduced because of the need to distinguish between various consequences of transactions.

For protection we adopt a similar point of view: data needs to be protected because of what happens to it. Special consideration must be given to the consequences of accidental harm.

An object, our equivalent of a piece of property, is defined as a bundle of rights. Each right in a bundle represents an important factor in a transaction because its handling implies a different operating system processing. One of the reasons

it makes sense to use the right as the basic entity for describing objects and transactions is that a right can be tied to the notion of capability. The idea of reducing an object to a bundle of rights is not too different from the point of view illustrated by object-oriented operating systems, except possibly in its systematic usage. Usually, in an object-oriented operating system, when access is allowed to an object, a certain number of operations on that object are implicitly allowed. We can view these operations as implying that specific rights exist on a certain object.

The legal system distinguishes between real and personal property but describes them with the same kind of rights. We made a distinction between two aspects of protection: protection of resources and protection of information. We now extend this distinction by creating two classes of objects: resource objects and information objects.

A resource object is the protection abstraction used to represent a piece of a physical resource of the system or its logical equivalent. For example, it can be a piece of main memory, or a segment of the virtual memory of a process.

Often in an operating system, pieces of information must be globally known by every process; they usually reside in a fixed location. This is the case, for example, of the pointer to the communication vector table in OS/MVT, or the volume identifier of a tape or a disk stored in the first block of the tape or the disk. These pieces of information have been attached to a resource and, as in the legal system are resources.

For the purpose of resource management, table entries are often used by the operating system to point to the resource objects of the system. If one owns an object, this entry "incidental" to its usage should be part of the object and also owned. We hesitate to suggest this interpretation, because often the integrity of such a table can be achieved by denying its access to the principals. This is inconsistent with the fact that they own entries in it.

Information objects are everything that are not resource objects. Rules must be defined for transforming one type of object into the other. We illustrate this point with an example of a removable disk pack, which uses some of its space to store the directory of the removable volume. Directories are an important part of the description of the rights of a principal [Dennis 66], they must be a resource object, always accessible. But the part of the directory on the disk pack appears and disappears, which is a contradiction. A possible solution is to admit that when the disk pack is mounted its volume directory, an information object, becomes a resource object.

3.2.2 Principal and owner

Our intention to tie the concepts of owner and principal together seems obvious. But to convince the reader, we need to prove that the three classes of rights which define an owner are used in protection.

Protection examples of the rights of disposition, exclusion and use are given first. They are followed by a description of the consequences of co-ownership.

The classification made on disposition, exclusion, and use is not the only one possible. Other classifications are possible, but it is the most important one, because it is based on the consequences of owning the right.

A computation having the right of disposition on some specific object cannot be treated the same as computation with only the right of use. Because of its consequences, the right of disposition cannot be created, transmitted, etc., in the same way as lesser rights. The consequences of this point of view must be very clear. In some recent papers [Jones 73, Needham 72], the tendancy has been to define protection systems consisting of protection policies and a set of mechanisms to enforce or implement these policies. The hope is that any protection policy would be implemented, using a finite set of mechanisms. It may be possible to do so, but our distaste for programming Turing machines is such that we have taken a position in marked contrast to this approach. Having a set of representative policies, we want to define the objects and structures that will allow easy implementation of these policies. Some generality will be lost, but we feel that the complete generality of Turing machines have too big a price: the difficulty to implement meaningful policies with them. Our classification by consequences is the principal way we introduce the policies because the consequences are

dependent on the policies.

The right of disposition refers to the following:

- the owner of an object may keep it as long as he wishes (subject to some operating system privileges);
- he may dispose of it, or some part of it, in the manner he chooses (dispose means loose all rights);
- he may wait for an automatic disposal by the system when the computation is destroyed or he may specify the terms of disposition at that time.

Our model of protection uses the principle that every object has at least one owner. In the case of a resource object to dispose of it implies that it will cease to exist as an object owned by a specific principal,but must now belong to another principal that either acquired it or was a co-owner. Using the standard terminology of protection,this disposition implies to give it away. To dispose of information objects which one owns solely means to delete it. If the information object was shared, at least one co-owner exists, and he may remain in ownership of some of the rights.

This example illustrates once again the need for the distinction between resource objects and information objects. Basically, what happens to them is different.

The first notion of keeping an object as long as the owner wishes, is widely implemented. Operating systems in one form or another allow long term storage of objects (files in most cases) and except in some specific cases like the owner not paying its computer storage bill,the owner is always in charge of disposition

of the object. In a system like Multics, because of its backup system, copies of objects will always exist even after an owner disposes of them. However, accessing them becomes practically impossible for the ex-owner without the help of the system administrator.

We must distinguish between the two classes of objects when we deal with the notion of an owner choosing the manner of disposition of all or part of his objects. The explanations given above for disposing of an information object make it obvious that systems, CAL-TSS,for example, have implemented this notion with the help of "grant" and "delete" operations. But we do not know of an implemented operating system where a principal can freely dispose of its resource objects. This situation exists because, in fact, the resource objects are leased from the operating system to which they return by reversion. For principal A to dispose of its resource objects in favor of principal B, implies that B remains in possession when A disappears, a solution usually not implemented, although there is nothing against it [Dennis 66].

The last notion deals with what happens in the case of a computation being deleted or aborted. The typical point of view is based on two premises:

1) The retained objects are not destroyed.

2) The computation has the responsability of transforming its owned objects so that they are disposed of according to its

wishes when deletion occurs. In case of an aborted computation some conditional system processing may be allowed to insure disposition of the objects.

The right of exclusion deals with the notion of forbidding access to the object but also involves the creation of a smaller bundle of rights given to a computation or a principal without loosing ultimate ownership. Examples of this are:

- the right for a computation to be called by another;
- leasing of any portion of an object;
- granting a read-only access to a file.

Absolute exclusion is one of the rights of an owner that is always implemented by the protection system. It may have a direct implementation through an access list attached to an object. Some information is furnished which identifies all the computations or principals that are not excluded, and all others are excluded. It may have an indirect implementation using capabilities. Exclusion exists because the computation does not have a capability for the object.

The right to be called refers, for example, to the right to use a protected entry point [Dennis 66]. If we can guarantee that all the calls to a procedure use a unique entry point, then we can check the parameters of the call and insure some minimum level of protection. Protected entry points are not a complete solution, and they must be used with other mechanisms which can be quite complicated [Schroeder 72b]. Another illustration that

having rights does not insure protection per se; one also needs to describe how one is allowed to use them.

Leasing will be studied later, in Chapter 4.

Any read only access to a shared piece of information can be viewed as an easement, as it only gives the right of usage. We will study the equivalents of easements and liens later in this section.

The last class of ownership rights is the right of use which is under the control of the operating system. For example, the owner of a file may use it as he wishes, reading and writing it, but he cannot increase the file size to any limit without being controlled by the operating system.

An object may be owned solely or jointly. Among the issues at stake are those of creation, survivorship, and transfer. Creation deals with the possible unity needed from the co-owners at the time of creation. Unity of time implies that all co-owners exist at the time of creation. For example, the computation driving an I/O device and the computation requesting that device both exist at the time an I/O buffer is created. They are co-owners of the buffer (maybe the operating system is a third owner) with utility of time at creation. Unity of interest implies that all owners have the same rights on the object. All I/O devices are co-owners of the I/O channels with unity of interest. But in the I/O buffer case, we could have decided to distinguish between reading and writing into a buffer and thus decide there

was no unity of interest; the driver's computation only reads when the user's computation writes and vice-versa.

The survivorship problem deals with what happens to the share of one co-owner when he is destroyed. The survivors are obviously the remaining co-owners. Nothing may happen; the remaining co-owners are given all the rights of the destroyed owner (this is called the right of survivorship). This may be the case with event-channels; the disappearnace of a co-owner (as well as its appearance) does not really change the fate of the survivors. Or something may happen; the co-owners may loose some part of the object that was associated with the destroyed owner.

The problem of transfer involves what happens to the object if one of the co-owners transfers his rights. He may or may not be allowed to do so. If the transfer is done when he disappears we are again in the case of survivorship.

Some form of ownership always exists in the current systems because at least some forms of the exclusion and use rights are implemented. For example, in Dennis and Van Horn a principal is the closest to the owner described above. It has the power to create, modify, and delete segments and give away accesses to them. In the computation's domain some form of ownership is created by appending the ownership indicator to a capability, making a distinction between the rights of an owner and a simple right of use.

In the Lampson matrix-model the control attribute, copy flag and protected access correspond to various combinations of exclusion and disposition.

3.2.3 Directory and estate

A directory matches the names a principal uses to describe its retained objects and the retained objects themselves [Dennis 66]. The directory holds two types of information; information describing how another principal can use the retained objects and information that represents the retained objects themselves. Knowing that all objects in our model are described by the bundle of rights that ties the owner or principal with them, we can give another definition of a directory; it is a structure that associates a principal with all his interests or rights.

The estate is the structure in law that fulfills a similar purpose. A principal owns a directory in the same way an owner owns an estate. Another similarity between an estate and our description of a directory exists when we associate access lists with the retained objects; the same way the law associates encumbrances with an estate.

We must remember that the legal system is based on a posteriori checking of rights owned by people. The contracts that are needed to create a valid property transaction are devices used to describe some of these rights but they do not describe all the rights and encumbrances that arise from the transaction. The court system is used to enforce the law. The courts are empowered to collect the facts and there is no need to gather beforehand an

exhaustive description of the rights of an owner. We cannot imply a large part of the rights manipulated by an operating system, because this would result in an unacceptable overhead for checking. Thus, the solution in this disseration as well as in many actual implementations, has been to exhaustively specify all the rights and encumbrances. The encumbrances represent the rights that others have on the retained objects. The name "access list" is chosen to represent this set of rights.

If we want to characterize an estate, we may specify the duration of the estate, the number of owners, or the time of enjoyment. We attempt now to characterize directories similarly. What is important is to recognize the need for such a structure and to provide for characteristics that are useful.

Duration can be divided into the following types:

1) Unlimited duration: this represents the largest bundle of rights. In particular, no limit is set on the transfer of rights precluding the disappearance of the owner or his death. A principal owning some retained objects would seem to have an unlimited directory. Only his disappearance from the system or a transfer initiated by him would result in the disappearance of the directory.

2) Life duration: the directory depends on the existence (life) of another principal, or the directory's owner. When the owner of such a directory disappears, the

directory reverts to another principal; this is the most frequent case. If a directory is dependent on a principal's life, then the principal does not have the powers of distribution by will. An example of this case is a principal disappearing and having his retained objects returned to the operating system. A directory may be dependent on the life of another principal as in the case of a principal owning objects with many other principals all working on the same project. The project manager creates a sub-directory for each principal, but these directories only exist as long as the project exists and disappears with it.

3) Limited duration: A known limit is set for the duration. This is often the case when a directory is created by a lease. Temporary files created for one day are a common operating system service. They result in cases of limited duration directories. This example illustrates a minor confusion that could arise because of our choice of the word directory. It describes the abstraction representing the set of rights and encumbrances of an owner. This abstraction may be implemented as a directory in the sense used in most filing systems. We may even talk of a directory that is a file.

4) Conditional duration: This directory lasts for an unknown period of time, as long as some specified condition holds. A temporary file that exists as long as a computation exists is a case using a conditional directory.

The last characteristic, the time of enjoyment deals with the fact that the owner of a right may not always be in possession of it. Possession may occur after a specific period of time or on the occurrence of an event. An example is the right of an operating system to abort a computation if a time limit is exceeded, or if some fatal error occurrs. A problem exists because we can also describe this event as being a condition of reversion in the lease given by an operating system to a computation. This is equivalent, and the reverted rights of the operating system are rights characterized by their future time of enjoyment.

3.2.4 Domain

We have already introduced one difference between Real Estate Law and protection: the problem of enforcement. In Real Estate Law the estate is not an enforcement structure. It is used in the description of transactions and thus represents part or all of the information needed to ascertain the rights of individuals. The process of enforcement in the legal system uses the court system,which is called upon to judge, only in cases of conflicts. It is an a posteriori system and, for it to work, the facts of the case must first be gathered and then judgement is given. Thus, the structure used to represent the interest of each party may not describe all the facts needed for a particular judgement, because some facts may come as consequences of specific situations. It would seem that in an operating system we need the equivalent of a court system in the sense of some entity able to weigh facts and render a judgement. This process must have some of the properties of the checker-enforcer described in 2.2.3. But most systems require an a priori checking of the facts, therefore, we need another structure to hold these facts.

Domains have been introduced in previous works to represent the set of powers available to a computation. Our model has described a principal owning some objects, represented for protection purposes by a bundle of rights. The structure that describes the relation between a principal and its objects,is the directory. Transactions on these objects may change the directory by removing or adding rights to the lists existing in the directory.

When a program is in execution we need to restrict it to the actions allowed by the set of rights of its principal. The domain is a structure used to achieve this goal. A domain does not own an object. It is a principal that owns an object and through this principal a computation may own the same object. The domain has been described as consisting of a list of capabilities. The capability is the enforcement representation of a right. We cannot consider a capability to be the exact equivalent of a right because the notion of a right describes something like a potential power subject to reevaluation at the time of usage. We know, for example, that the law makes a strong distinction between having a right and exercising it, because quite often the degree of use is a parameter of the transaction. Thus, the capability can be looked at as being the binding of the right to a physical representation valid for a specific computation.

In our model can we, during the existence of a computation, add new rights to the directory of a principal? The answer is yes. Does this mean that we have a domain? No, because a domain is related to a specific computation of a principal. The principal does not change if the directory grows or diminishes. The same thing is true of a domain. Notice in particular, that adding a right to a directory may not imply adding a related capability to the domain.

Notice, also, that most domains represent through capabilities only the rights owned by the computation, and none of the liabilities resulting from the rights of others on the objects owned by the computation. In line with our opinion that transactions involve exchange of rights between principals, we feel that improvements result from adding the liabilities to the list of facts represented

by the domain and used to enforce protection.

3.2.5 Other classes of rights

We do not have specific names to describe, in the protection context, the equivalents of an easement or a lien. Consequently, we now wish to characterize other classes of rights, using the examples of the easements and liens, that would be useful in a protection implementation.

We have described the rights of ownership. An easement gives its owner a limited use of enjoyment of a particular object. A lien gives its owner a conditional right of disposition on an object, for the purpose of securing some action.

Easements are created by contract or by implication. Two objects may be used in such a way that the owner of one of them has rights to the other object, because of the implications of its use. An example would be the rights of a computation to read information about this computation stored in tables which belong to the operating system. The constraint we have requiring the exhaustive listing of all the rights of a principal diminish the importance of implicitness in the creation of an easement.

Easements are also created by contract. An example is when a principal gives away to another principal a read-only right to a file. It is important to emphasize the fact that having a right often implies being allowed to perform some actions on an object, but usually the transactions performed by a principal imply

more than one right, because many elementary actions may result from them. In this context an easement implies a very limited use of the object.

In a similar way, a lien restricts its owner to the right of disposition of an object only if an agreed upon transaction is not performed by the owner of the object under the terms of the agreement. For example, using space on a disk constrains the owner of the files to pay some charge to the operating system. When the charge is due, the operating system may dispose of the space, if the charge is not payed.

CHAPTER 4

TRANSFERS

4.1 Property transfers

The civil code describes many methods of acquiring property: 1) by occupancy, which requires continued adverse possession for a minimum period of time; 2) by accession; 3) by transfer; 4) by will or succession. Other methods exist like acquisition: by condemnation through the use of state power of eminent domain; or by escheat, when property reverts to the state in the case of death of the owner intestate and without heirs.

This variety of acquisition methods do not really interest us, because they are dependent of human factors. For example, acquisition implies the automatic ownership of property used by a person for five years, without objection from the true owner. This seems in contradiction with the function of an operating system which attempts to manage resources in order to avoid waste.

From the many forms of transactions we have selected three. In the operating system section of this chapter we will call them: 1) the simple transfer, 2) the transfer with reversion, and 3) the transfer with intermediary. Illustration of the components of this slightly artificial classification, as far as the law is concerned, is given in the following fashion: the description of deeds and of the sale of property are used to illustrate the simple transfer; the landlord-

tenant relationship is used to illustrate the transfer with reversion; the real estate broker and the excrow are used to illustrate the transfer by means of an intermediary.

The study of trust deeds and mortgages, not undertaken here, could prove quite fruitful. They are instruments employed to create a lien on real property as security for the payment of money or the performance of some other obligation. In a contract of sale,legal title is retained by the seller until contract terms are met for the purchase price. In the case of a security transaction involving either a mortgage or a deed of trust, the legal title is entrusted to the debtor (buyer), subject to the effect of the trust deed or mortgage in favor of the creditor (seller).

Probate proceedings also are not described. When an owner is deceased, the court in charge of probate proceedings plays a neutral role, and for the purpose of our comparisons we found the description of the escrow to be sufficient.

4.1.1 Deeds

A deed is a written instrument, executed and delivered, by which the title to real property is transferred from one person to another. Grant deeds and quitclaim deeds are the main types of deeds used in California. In a grant deed the grantor warrants (doctrine of implied warranty) that he has not previously conveyed or

encumbered the property in question and conveys any after-acquired title, unless otherwise expressed. Whereas, a quitclaim deed transfers only the interests at the time of conveyance.

A valid deed must have:

1) a competent grantor,
2) a grantee capable of holding title,
3) a sufficient description of the property,
4) operative words of conveyance,
5) due execution by the grantor, which means that it must be signed,
6) delivery, and
7) acceptance.

Each requisite can be expanded. For example, the grantor must be designated properly as must be the grantee,etc. We will not study each point but we insist on two important points: 1) the delivery and 2) the acknowledgement of the deed. The latter of which is not a condition for a valid deed but helps in case of conflicts with a third party.

The acknowledgement made by an officer, designated by statute, states that the instrument was executed as stated. Its main purpose is to record the instrument. A failure to acknowledge does not necessarily prevent the admissibility of the deed as evidence, but such an admission could then require court action.

Delivery must be made to insure the validity of the deed. Manual delivery is not always legal delivery, and intention, certain presumptive facts, and time of delivery all influence the binding legal delivery.

The legal status of a grantee, (e.g., a convict), can also affect the validity of a transfer.

4.1.2 Sale of property

A land contract is a contract where the vendor agrees to convey the land to a buyer upon payment of the purchase price, or performance of some other act. Since it is a contract it must have capable parties, mutual consent, a lawful object, and sufficient consideration. In addition, specific requisites are:

1) a written agreement,
2) names and signatures of both parties,
3) sufficient description of the land,
4) a designated purchase price,
5) the time and manner of payment,
6) the number of years required to complete payment in accordance with the terms of the contract, and
7) the basis upon which the tax estimate is made.

The common types of contracts are: listings agreements, sale deposit receipts, escrow instructions, purchase and sale agreements, and installement contracts. The listing agreement is a contract between a real estate broker and a prospective buyer or seller of land. The sale deposit receipt, when executed by both seller and buyer, is a contract for the sale and purchase of land. Escrow instructions when executed constitute a valid contract of sale. Purchase or sale agreements may supplement escrow instructions for items with which the escrow is usually not concerned.

An option is a contract where a lack of mutuality in the obligation is created, since it is defined as a contract by which the owner of property invests another person with the right to purchase such property at a stipulated sum within a specified period of time, but without imposing any obligation to purchase.

A contract for the purchase and sale of real property passes to the purchaser the equitable ownership, leaving the legal title with the vendor for securing the payments and the performance of the other conditions of the contract by the buyer. The vendor may convey the land to a third party. Such a conveyance passes all the vendor's rights to the land, including the legal title and the right to receive the unpaid purchase price.

The purchaser is not entitled to possession of the land unless he is given the right of possession under the contract, or unless the vendor places him in possession. The question of risk of loss, and who should bear the loss of the property if the property is materially damaged, is usually dependent on the right of possession. When the purchaser has fulfilled his obligations under the contract, he is entitled to a conveyance in a form sufficient to pass the title.

4.1.3 The landlord-tenant relationship

The relationship of landlord and tenant arises when there is a hiring of real property. A hiring is defined as a contract by which one person gives to another the temporary possession and

and use of property, other than money, for reward, with an agreement that it shall be returned at a future time. A lease is the designation given to the contract by which the possession and profits of the land are exchanged for rent.

An occupant of real property can be a tenant or a licensee. The fundamental distinction is that a tenant has a legal interest in the possession or right to possession of property, whereas a licensee merely has permission to do certain acts on the property in the possession of another. Acts, which without the license, would constitute trespassing. A license is a right personal to the licensee and is not assignable. Whereas,the interests of a tenant are. In general a license is revocable at any time. The distinction is also important in the duty owed to the person in possession.

The relationship of landlord and tenant presupposes a contract, express or implied, from which the intention to create the relationship must appear. The law distinguishes between tenancy at sufferance, tenancy at will, periodic tenancy, and tenancy for a fixed term.

Tenancy at sufferance arises by implication; it implies an expiration of the right to remain in possession, by somebody who went into possession lawfully. The tenant at sufferance lacks much of the protection afforded other classes of tenants.

Tenancy at will is created by agreement of the parties, but has no fixed term, and is terminable at the will of either party.

Periodic tenancy is created by the parties to continue for successive periods of some length, unless terminated earlier by

notice.

In the tenancy for fixed term, a tenant has the rights to exclusive possession for a fixed period. A lease for a term exceeding one year must be in writing, must contain the names of the parties, must include a sufficient description of the property leased, must contain an agreement for the rental to be paid and the time and the manner of such payment; and must state the term of the lease. There must also be mutual assent of the parties, and a consideration (usually the undertaking to pay rent).

In the normal lease the lessee has generally only the right to the use of the property. In an oil or gas lease he has, in addition, the right to take something from the property, viz, oil and gas.

4.1.4 Real Estate Brokers

A real estate broker is an agent. An agent is one who represents another, called a principal, in dealing with third persons. Such a relationship is called an agency. An agent is distinguished from a servant or an employee, the latter being defined as persons who are employed to render services to the employer and who, in such service, remain entirely under the control and direction of the employer or master. An agent is also distinguished from an independent contractor, who in rendering services for another, exercises an independent employment or occupation and is responsible to the one hiring

him only as to the results of his work. A person hiring an independent contractor has no right of control as to the mode and method of doing the work.

The distinctions between an agent and an employee and an independent contractor are of primary concern in connection with the following: 1) where liability is sought to be imposed on the principal for the wrongful act of the person hired, in which case it must be shown that the wrongdoer was an agent or employee; and 2) where claims for injuries to the person hired are made.

The term broker is less inclusive than the term agent, since it applies to an agent who, for a commission or brokerage fee, acts as a negotiator between his principal and third persons in connection with the acquisition of contractual rights, or the sale or purchase of property, real or personal, where the custody of the property is not intrusted to him for the purpose of discharging his agency. Commonly a broker is regarded as a middleman whose duty is to bring a buyer and a seller together. A broker must have a license from a regulatory agency to lawfully engage in his business.

A written contract of employment is essential to the creation of the broker-client relationship. Most contracts are evidenced by a printed form of listing agreement. Commonly used listings differ in whom is entitled to the commission if the property is sold by 1) the agent, 2) another agent, or 3) the owner himself. The also differ in the termination date.

It is a general practice to use a deposit receipt when accepting "earnest money" to bind an offer for the purchase of property. When duly executed it constitutes a contract for the sale of real property. The broker as an agent must exhibit good faith; must disclose the best offer; has a confidential relationship with his client; must use care and skill; etc.

4.1.5 Escrows

An escrow is a deed or other written obligation, delivered to a third person, to be delivered by him to the grantee only upon the performance or fulfillment of some condition. The deposit of the escrow places it beyond the control of the grantor, but no title passes until the fulfillment of the condition.

The Civil Code provides that "a grant may be deposited by the grantor with a third person to be delivered on performance of a condition, and, on delivery of the depositary, it will take effect. While in possession of the third person, and subject to condition, it is called an escrow". An escrow must be a valid written contract and must contain a condition. The deed deposited in escrow must be a sufficient and valid deed, and the escrow holder must be a stranger to the transaction.

The escrow holder or escrow agent must be licensed by the commissioner of corporations and cannot be an individual, but a

corporation organized for the conduction of an escrow business. All escrow funds must be deposited in a special fund. The escrow holder is an agent for both parties, but when the conditions of the escrow have been performed, the dual agency becomes an agency for each of the parties, respecting those things placed in escrow which each is entitled to money for the seller, title for the buyer.

Full compliance with the conditions of the escrow is a must before the escrow holder may deliver funds or documents. He must also take into account the time limit fixed by the instructions for performance. After the expiration of the time limit the escrow holder generally has no authority to permit one of the parties to perform unless stated in the instructions. Deposited documents can only be returned if the other party agrees. The escrow holder must not resolve conflicts arising from conflicting demands, it must be the task of the courts. Instructions are confidential. The courts are best qualified in case of forfeiture to decide what to do. The parties can mutually agree to cancel the escrow. If one party wants to withdraw without the consent of the other party, this is normally controlled by provisions in the instructions. The party not in default when the specified time limit has expired without completion may withdraw without the consent of the other party. This is normally controlled by provisions in the instructions. The party not in default when the specified time limit has expired without completion may withdraw, subject to any specific provision, but must act promptly. If neither party has performed either one may withdraw.

4.2 Operating system transfers

The principal differences between the legal system and protection deal with:

1) The exhaustive representation of all the components of a transaction in the case of protection.

2) The enforcement procedures.

In the case of the transfer of property, because of the importance of the transaction, the legal system requires a written contract. The elements that must be present to make a contract valid plus some elements which, without being required, improve its protection constitute the exhaustive list needed to describe a transfer and insure its enforcement in case of conflict.

We require that this exhaustive list of elements also be present in our protected operating system transaction. It is needed because little room exists for human judgement in the mechanistic processing used in protection. Enforcement will be achieved by matching a priori the elements of the list and a template of required elements.

It would seem somewhat surprising that only three types of transfer: simple transfer, transfer with reversion, transfer with intermediary, need to be used to describe operating system transactions. We have to remember that what is transferred is a bundle of rights. Depending on these rights a wide vaiety of transactions can be simulated.

Resource-objects, as well as information objects, are described by bundles of rights, consequently, these transactions apply, since they only transfer rights. We make a distinction between classes of rights of ownership (disposition, exclusion, etc.), or other rights (easement, liens), because when transferred they have different consequences and thus result in a different type of transaction. Once again, we insist on the fact that the characteristics of an object, the bundle of rights describing it, exist because of what can happen to the object and not directly because of its physical characteristics.

When describing the different types of transfer in this section, we are confronted with a problem of terminology. Protection does not have precise terms to describe the ideas presented here, whereas the legal system is quite precise. In some cases we will be forced to use the legal terms.

4.2.1 Simple transfer

To be valid, a simple transfer requires:

1) Identification of the principals involved.
2) Identification of the object.
3) Conditions and validity of the specific type of transfer.
4) Acknowledgement.
5) Delivery and acceptance.

Each of these requirements is now explained.

1) Identification of the principals.

It would be more correct to talk about the identification of the computations involved, because they are the entities which effectively request a transfer. But a computation is always associated with a principal, which in turn needs to be identified, if we want to identify the computation.

Identification involves identifying the grantor and the grantee, but also checking their competence. The grantor is the principal in current ownership of the object to be transferred. To identify him we need a unique name, which is unforgeable and can be transmitted freely as a means of identification. The need for such a name has been recognized in many studies, [Lampson 71, Hansen 70, Sturgis 74]. The unique name is certainly a general notion. It is used to identify objects as well as principals. In the specific case of identifying a principal for the purpose of a transfer, the directory capability owned by a computation [Dennis 66], and the access key: a member of a domain C-list [Lampson 69b], fulfill the same goal.

This unique name meets the requirements of identification and execution of a legal contract. Execution means that the grantor has signed the contract. The need for a "seal" has been described by J.H. Morris [Morris 73].

We now need to check that the grantor has competence to perform the transaction. This is done by checking in the principal's directory for the presence of rights on the objects which allow the

specific operation of the transfer.

In the case of the grantee, we need to identify him with a unique name, and we need to verify its ability to own the object. For example, in the case of a principal receiving full ownership he needs to have enough space to store the object.

2) Identification of the object.

To identify an object, we use a unique name to refer to the valid and complete description of its characteristics. The bundle of rights represent the protection characteristics of the object, but the object may have other characteristics, used, for example, in resource management, that must be described.

3) Conditions and validity of transfer.

This phase of the verification of a transfer, overlaps somewhat with the checks made on the competence of the grantor and grantee, because it involves checking for the rights to perform the necessary operations. The consequences of a transfer depend on the types of right transferred and they are checked during this phase.

4) Acknowledgement.

Acknowledgement looses some of its importance because we advocate in our protection system the systematic recording of all transactions. In this phase of the transaction, the data base of protection information is updated with an entry representing this transfer.

5) Delivery and acceptance.

Delivery is when the evidence of ownership is delivered to the recipient. It implies updating the directory of the recipient

and his domain if needs be.

Acceptance is the phase when the checker-enforcer principal, that was performing all the checking described above, looses his rights on the parties involved in the transfer.

This list of requirements needed to achieve a protected transfer raises the question of how do we check and enforce it. As was suggested in the paragraph above,we need to have a hardware or software module that will perform functions similar to a checker-enforcer.

This module, invoked automatically when a transfer is initiated,must be given the unique names of the grantor and the grantee. It checks their competence by looking at their directories, taking into account the type of rights involved in the transfer. Then, if the transfer does not violate protection requirements, it allows the operating system tables and the directories to reflect the results of the transfer.

This processing is complex enough to justify a principal being in charge of it. This principal is given the minimum rights necessary to perform these operations. In particular, they exist only after a call, only for the specific parties involved in the transfer, and only for a limited duration.

Other requirements of a protected transfer involve the notion of time. In particular, because of concurrency in operating systems and some of the problems that may result from it, like deadlocks, time may be an important factor in the acknowledgement, delivery and acceptance phases of a transfer.

4.2.2 Transfer with reversion

This is a very common case of transfer when a user and the operating system are involved. Often in such a case, resource objects are involved. For example, when blocks of central memory are given to a computation we can consider that the computation received the rights to use this part of central memory; even to manage it between itself and other processes. But the computation will not be able to dispose of this part of central memory, because ultimately it will revert to the operating system. When files are stored on a non-removable storage medium a similar point of view could be expressed.

The important factors specific to a transfer with reversion deal with the smaller bundle of rights that the owner of the object accepts to transfer and the time and conditions of the reversion of this bundle of rights. In particular, the time of reversion may be known in advance, may depend on a condition, or may occur only at specific instances when the total duration is unknown. The first case could be used to describe a filing system where files are created with an expiration date. The second case would be the temporary files used by a computation for as long as it exists and automatically purged from the system when the computation is destroyed. The last case involves for example, the notion that instant reversion could be harmful, because retained objects are not in a consistent state. Therefore, reversion can only occur at some agreed upon time, and the lease lasts for multiples of some periods of time.

Because of his rights of reversion, the owner of an object to be leased is given, in the transfer, rights to enforce the contract that will exist between him and the recipient. Before the execution of the transfer with reversion, he did not own these rights.

A transfer with reversion has most of the requirements of a simple transfer since they both involve a contractual agreement. The parties must be identified and competent; the object transferred must be described fully; the specific rights transferred must also be described. In the phase where the condition and validity of the transfer is checked, we must perform the simultaneous exchange of rights which is specific to this transaction.

We can summarize this by saying that transfer with reversion or lease describes an owner giving to another principal the temporary possession and use of an object to be returned in the future. The principal in temporary possession has more duties than if he were a full owner.

Transfer with reversion seems to be a frequent occurrence when we deal with pieces of resources owned by the operating system and shared by the user to achieve some management effectiveness. It is harder, or may be impossible, to find meaningful examples of transfer with reversion in the case of information resources. Once access has been given to the information, it is almost impossible to constrain the user not to copy this information and thus get most of the advantages of leasing the object.

This type of transfer does not solve this problem. Some aspects of constraining certain types of usage of information have been described under the same name of "confinement problem" [Lampson 73].

4.2.3 Transfer with intermediary

The idea of the intermediary involves three requirements: 1) existence of a specific contract that creates the relationship between the principal and the intermediary (this contract describes, in particular, the task to be performed); 2) description of the conditions and events that preside over the success or failure of the task to be performed ; and 3) description of the responsabilities of the parties in case of a protection violation.

A possible example of an agent would be a utility program. The description given of the parameters needed for the call, and of the action performed by the program are a form of contract. Although in the case of a call to a utility program nothing can be changed, and most of the rights needed are implied. The utility program is usually supposed to work so that no check is made on delivery and acceptance. Returned codes are used for the success or failure of the task. Finally, the utility is an agent, because a protection violation is the fault of the caller.

An escrow has the unique characteristic of being hired by the two parties of a transfer. In the transaction he plays the role of a neutral third party with specific duties to each party, checking

that each party performs as stated in the escrow contract. He takes no action in case of conflict not specifically mentioned in the contract.

An escrow implies the existence of a computation in charge of checking a transaction which does not have universal powers on all aspects of the two principals involved in the transaction. It achieves a protected transaction by having limited powers on the two dangerous phases of a transaction: the conditional delivery and the conditional acceptance. They are conditional on each party observing its own part of the contract.

We feel that many transactions in a computer can be reduced to a simple transfer or a transfer with reversion. We feel that the escrow system presents an interesting solution to the problem of insuring a higher level of confidence for each principal. The similarities with the mutually suspicious subsystem problem, and Schroeder's solution using dynamic capabilities [Schroeder 72b], are striking to us.

CHAPTER 5

PRAGMATIC LIMITS

5.1 Pragmatic limits in the law

This chapter exists mainly because of its second section where we briefly discuss some pragmatic limitations in protection. But we could not resist describing some Real Estate Law concepts which deal with somewhat related issues. In this section no real attempt is made to compare them side by side with protection concepts.

We first describe the set of events that take place for a trial arising from a property dispute. Then we describe how land is described, what is the purpose of recording, and a specific case of recording the "homestead".

Title insurance is a rather complex part of Californian Real Estate Law. The lengthy description we give of it, although quite simplified, is justified by the unique aspect it displays in trying to minimize overall cost to society and to individuals through the process of abrievated searchs and of insurance protection.

5.1.1 Resolution of property conflicts

In the case of a dispute on property or other rights the following set of events could take place. Notice first that a conflict needs to be detected by the parties, if the conflict is not resolved by the parties, then they hire specialized people, gather facts and the matter is taken to court. The court hears all

the pertinent facts, which means it requests some of them, and judges their admissibility. Then it applies law, which implies that a law exists, and decides in favor of one party. The court could also admit its inability to decide for whatever reason, including a lack of appropriate regulation. The losing party may appeal to a higher court. The "options" of these courts are a very important source of Real Estate Law since they discuss the legal principles involved and state the reasons for their decision.

5.1.2 Land description

Real property can only be transferred by an instrument in writing. For it to be valid it must contain, among other things, a sufficient description of the property. The exact limits of the property are of paramount importance because of the conflicting claims that could possibly arise between two adjacent real properties.

The widespread availability of non ambiguous descriptions of real property is a recent occurrence and many problems have arisen from the inconsistencies and vaguenesses of the old descriptions, usually because the methods described below were badly used even when used.

A typical case can be made with the description of land by metes and bounds. Today metes and bounds descriptions are done with reference to officially registered starting points stating the courses and distances of boundary lines from them with standard

measuring devices. But it used to be that natural boundaries were used, monuments or natural references such as a dead tree, or a stone pile. Obviously, the accuracy of such description faded with time.

California was surveyed using the government Survey System. It uses a set of officially defined landmarks designated by their longitude and latitude and a grid system. Each section of the grid system can be uniquely identified, and property inside the section is defined by standard rules.

We should mention also that maps or plats have been generally recorded at the office of county records, and it is a recognized practice to reference a piece of land by its lot number on the map along with the recorder's office identification of the map. This leads us to the second part of our description: recording.

5.1.3 Recording

An "instrument" is "a paper signed and delivered by one person to another, transferring the title to or creating a lien on property, or giving a right to a debt or duty" [Bowman 70, p.200]. A deed is an instrument transferring property.

Recording deeds and other instruments has two main purposes: to perserve the evidence of these instruments and to impart constructive notice to subsequent purchasers and encumbrancers. The original record or a certified copy of the record has the same force and effect as the original instrument. Even though he may

not have inspected the public records, anyone is deemed to have notice of the properly recorded conveyance from the time of the recording of this conveyance in the office of county recorder in the county where the property is situated.

A non recorded conveyance is valid between the parties and anyone who has actual notice, but is void against a subsequent purchaser in good faith. The time of the recording is important because usually the first conveyance recorded will prevail. But recording does not validate a void deed. Constructive notice results from the proper recording of certain specific instruments. In particular instruments or judgements affecting the title to or possession of real property may be recorded. Some instruments are ineffective unless recorded. It is possible to record notice of a pending action to protect against a move that would render it ineffectual.

5.1.4 Homestead

A homestead is a special estate created when two conditions are met: the residential property is actually occupied by the claimant and a declaration of homestead is properly executed, acknowledged and recorded.

The homestead law protects a home, within prescribed limits, from forced sale to satisfy certain debts, or in the case of a marital homestead restricts against conveyance by one spouse without the consent of the other, and gives the surviving spouse special

rights in the case of death of the other spouse. One can only have one valid homestead at a time.

5.1.5 Title insurance

A title insurance policy insures the ownership of an estate or interest in land, or the priority and validity of an encumbrance on land. It is a contract to idemnify against loss through defects in the title, or against liens or encumbrances that may affect the title at the time the policy is issued.

The danger, as time goes on, that important documents will be lost or destroyed, and the voluminous accumulation of documents resulting from a need to keep the originals over a long period of time, led to the establishment of the county recorder's office. As time went on, more and more reliance came to be placed upon the recorded title. The increase in the number of documents affecting a particular parcel, and their distribution among various public offices, led to the creation of searchers of title known as "abstracters".

The work of an abstracter related only to the compilation of the "chain of title". It did not involve the construction, interpretation, or legal significance of the various items comprising such chain, which was the task of the lawyer. To pool resources and save on already found chain of titles, abstracters formed abstract companies. But the abstract opinion system of establishing title failed in many instances, being too slow or too costly, or plagued by the fact that the liability of the abstracter and the attorney was limited.

The next important development was the insurance of a "certificate of title". Instead of preparing a formal abstract of title, the company would only reach an opinion as to the current condition of the title and issue a "certificate of title". The next development was to guarantee the title.

A title insurance policy represents the final result of three successive processes: an examination or investigation of the title, a determination of the amount of insurance required, and the protection of the insured against possible title losses. The elements of risk or chance in title insurance arise from the three principal sources:

1) errors in searching the record,

2) errors in interpreting the legal effects of instruments found in the chain of title,

3) facts external to the record.

The third point is the main characteristic that distinguishes a guaranteed title from an insured title. The main hazards involved relate to the identity and the capacity of the parties.

Some matters are usually excluded from coverage by the standard policies: taxes and assessments not shown as liens, easements and encumbrances not disclosed off the record, instruments not in chain of title, rights of persons in possession, other matters disclosed by inspection, physical characteristics of the land, errors on recorded maps, discrepancies in patents, mining claims, reservations or exceptions in patents, water rights, governmental acts and regulations relating to use, defects known to be insured.

5.2 Pragmatic limits in operating systems

What we call pragmatic limits involves two distinct subjects. One deals with technological problems which under different cost/technology tradeoffs should disappear. The other involves intrinsic problems, which arise because of specific assumptions or processing methods.

Examples of technological problems are the loss of granularity resulting from protecting whole segments instead of fields, and the loss of control from using only two or three types of accesses (read, write, execute).

Examples of the intrinsic problems resulting from implementation choices are the various advantages or disadvantages of capabilities versus access lists or versus message systems.

5.2.1 Some syllogisms about pragmatic limits

We are dealing with real systems, using computers made of components with a finite life, and software module with bugs in them. Thus, no flawless protection system could exist under these conditions. Even if we had computers without failures and software without bugs, most people would admit that a perfect protection system is impossible.

Pressed to say why this is true, possible arguments will be given. The overhead would be unacceptable, or the protection system interfaces with human beings: perfection is impossible. Why is it that some of the statements above seem true, but cannot

really be proved or disproved? The first clue is that we all have the conviction that a protection system mainly uses the checker-enforcer mechanism described in 2.2.3. It is based on the idea that a representation of what reality should be is compared to actual reality. Depending on the comparisons an enforcer allows or forbids some action. Obviously, this kind of checking will insure some protection. A dangerous temptation is to accept this mechanism as the definition of a protection system, implicitly or explicitly, or as the goal for a perfect protection system.

Let us describe some of the implications of this idea by introducing first what the "reality" should be and what it is. The problem lies in defining or encoding this reality. For physical resources it is somewhat easy. We represent them by some set of parameters and compare values, this also works for logical resources. There are many problems in which at the time of the checking,we don't know what the reality should be. There are many problems where, although we could find the values of the parameters, it would be costly. There are many problems where the time to do the checking is difficult to determine and yet checking at every possible time is too costly. Consequently, we do not implement this method in all the apparently possible instances, and maintain that improving protection is only a matter of adding more of these tests. This reinforces the impression of unacceptable overhead associated with full protection, since more protection implies more tests.

Most people feel that the enforcer is less objectionable.

Most of the implementation we have read about describes what we like to call a traffic light system. If the light is green the action is possible. It it is red no action is allowed. Unfortunately, the conjunction of some actions allowed independently on a go-no-go basis may be objectionable. A common example arises with capabilities. A computation can access an object if a capability for it is given to the computations and since that computation may trigger some other computations, one would limit drastically their power if the right to transfer capabilities was not allowed. But when a principal wants to revoke a capability he is confronted with the problem of many copies stored in unknown places unless... solutions can be found to this problem — most of them drastic. This example shows that, when a principal allows some action, we may require the transfer to the principal of new rights to deal with the consequences of this action; an exchange impossible to do with a traffic light enforcer.

5.2.2 Correct notions about pragmatic limits

The model of protection described above tried to encode reality into parameters that can be compared, limiting its scope to problems where such an encoding is possible. The mismatches between these parameters and reality are equivalent to unchecked protection differences, making more acute the inexactitude of the abstractions.

Our model of protection selects parameters (rights),not on their accuracy to represent reality, but on the consequence of their manipulation. A set of representative transfers is also described.

They manipulate the parameters by following specific rules.

Pragmatism appears when we isolate the classes of rights used to characterize our objects, and when we define the rules to be followed by a transfer. We feel that a mechanism based on the consequences of transactions is easier to use because protection failures are more directly related to the pragmatic limitations accepted. In contrast, in a mechanism based on the encoding of physical or logical characteristics of the objects, we find that complex combinations of the parameters are needed to achieve a control of the consequences of an action. The control of the consequences of an action is what a user really wants!

By design, the consequences of the usage (or the non usage) of the classes of rights described in previous chapters have been introduced with them. Enforcement of the protected relations has also been described. Thus, the rest of this section will concentrate on the pragmatic limits commonly found in current protection systems. The central issue is the match between objects and the protected reality.

5.2.3 Match between objects and the protected reality

In the most common protection schemes the objects used are essential to the achievement of protection. Their characteristics are compared to some expected value to allow or disallow actions which are used to achieve protection. The problem of the match between a) the objects, b) the characteristics used to represent

protection and c) the reality has often been centered around the following questions:

1) choice of the physical or logical level at which to apply the control;
2) granularity of what is accessed;
3) identification of accessing principal;
4) sophistication of the kind of restriction allowed to the user.

These problems must be viewed in light of the desire to tailor protection according to who is requesting access, the content of the data, the context of the request, etc.

The most commonly used objects are often parts of the physical resources managed by the operating system, or the logical equivalent of these resources. Memory protection is a simple example used as an illustration. One can protect blocks of physical memory,decide that the whole block will be accessed uniformly, associate a unique key with all the users able to access this block, and distinguish only between the existence or absence of reading and writing privilege. One has made pragmatic decisions on the four problems mentioned above. Other characteristics can be used, such as the size of the block, but very rapidly the possibilities are exhausted because one always deals with characteristics usable in a traffic light checking system.

The limited potential of the lock and key method described above has induced people to choose more subtle solutions which are still within the limited conceptual frame described

above. A sizable improvement is provided by the realization that one would want different accesses for different users of the same data. Thus, the most often mentioned improvement to this method is to associate access checking with a logical resource. In the case of a memory block, a segment is an obvious choice since virtual memory is often implemented. The block of physical memory could be accessed only by the principal holding a specific key identifying him and his access privilege.

In the scheme using a segment of virtual memory one may separate the issue of identification from that of access by having in the memory space of the computation only the segment it can access. This follows the principle of least privilege. We need a module to decide if a segment can be known to a principal. This module needs data to represent the access information in terms of the identity of the principal. Usually a directory is used for this purpose. Something is lost in the process as described above: the checking in the memory block case was done for all accesses, the checking in the segment case is only done when the segment is made known. Revocation of the access authorization can still be performed after the computation knows the segment, but this involves complicated back-pointer schemes as illustrated by Multics [Bensoussan 72, Organick 72]. A possible point of view in examining these trade-offs is to consider, as we did in our model,that a describing structure exists: the directory, and an enforcement structure exists: the domain. The enforcement structure provides as late a binding time as possible between the described access and its

data representation at execution time.

Once we have achieved the restriction of access to the principals described in the directory, we take advantage of our logical resource to attach to it a code describing one of the possible mode of accesses. Unfortunately, because of the nature of our protection method, the variety is limited. Read, write, execute are often mentioned. One possible reason for such a fact is technological since by having only a few cases one saves on the representation, but it is in contradiction to the fact that these access types are appended to blocks or segments, and the number of blocks or the number of active segments is not extremely large. A better reason would be to notice that the size of a segment or block is usually too large to allow meaningful types of accesses to be more sophisticated than read, write, execute. Another possible reason, which expresses our belief, suggests that with the traffic light checking method we cannot implement more sophisticated varieties of accesses. The type of accesses mentioned above all deal with gross misuse of the data: to modify read only data, to write on code, or to use code as data. What we are interested in now is control over how the data is used, since the most pressing problem of avoiding destruction is apparently solved. This control can only be achieved if dynamic information is provided to a module in charge of the access about the content of data, the environment, and the identity of the principal. This method of operation is not what we mean by a traffic light checking system. We might point out that it has often been suggested that some control over the use of information

can be achieved by using a procedure which must be accessed by everyone using the information. Seals, formularies, filters, etc. are among the many representatives of this idea [Morris 73, Fabry 68].

The granularity of what is accessed is often contrasted in the current schemes with the variety of accesses available. This results from the belief that a complex control of a data base can be achieved by controlling access to the elementary field of the data with primitive operations like read, write, delete, etc. This conviction would induce the protection designer to reduce the size of a protected object to the field of a record. This raises obvious technological difficulties because of the number of such fields. But, at least in the case of a large number of records, one can group all the fields requiring the same protected access into a segment, and by a data organization trick achieve protection. Instead of having a trade-off of fineness of protection versus size of protected object, we have now a trade-off of data organization versus fineness of protection.

If we want to protect access to a procedure instead of data, we find that the protected entry point solution works fairly well. One can pack more than one procedure per segment, if one uses an index to designate the real entry point, which is stored in a table at a known location in the segment.

CHAPTER 6

CONCLUSION

The legal system has provided us examples of representative transactions in a field preoccupied with regulating conflicting requirements on scarce resources. Most of the results of this dissertation depend on the facts that, not only does protection in operating systems encounter problems similar to those in the legal system, but also that the control of shared information, which does not seem to involve scarce resources, can also use the abstractions invented by the law.

Two aspects of the legal system may explain this situation. The law regulates transactions. The control of physical resources is only dependent on their manipulation by the transaction. Thus, the abstractions used for the objects of the legal system are quite independent of their physical realization. The law applies as well to wagons as to cars. The second aspect of the law which may explain this situation deals with the importance of the "right". Property is defined in terms of rights, ownership is defined in terms of rights, an estate is a specific group of rights, and transactions manipulate rights. As long as information can be described in terms of rights, for its protection requirements, we feel personally quite convinced that the structures described above work well to protect information.

This is not to say that resources and information protection will result in the same mechanisms at the implementation level. Some physical characteristics will result in different consequences in the transaction and will require specific rules.

The notion of right used in the law is very well matched to the ideas used in most protection systems to justify access to data. You may transform a piece of data if you have the right to do so.

A notion that is central to our view of protection is the notion of ownership. All the objects in the computer system must have at least one owner. Ownership, as in the law, suggests the existence of rights on objects. Thus, this point of view only represents a formalization of a more or less accepted philosophical concept.

The model of protection presented in this dissertation can be described briefly as follows: we introduce first the notion of a right. A principal is a fictional personification having a set of rights on a set of objects. An object is not defined as a physical piece of resource but as a bundle of rights. Full ownership of an object implies the existence of rights classified in three groups: rights of disposition, exclusion or use.

The difference between the enforcement in the legal system and in operating systems results in the introduction of the directory as the exhaustive list of all the interests and encumbrances of a principal. Rights are not limited to the three classes described above, we introduce easements and liens as example of other types of rights.

To allow dynamic enforcement of privileges for a computation, we must also introduce the domain: an enforcement structure binding the rights of the directory to capability-like entities.

Finally, we introduce three types of transfer: a simple transfer, a transfer with reversion, and a transfer with intermediary.

The goal of this dissertation was to put some order in to the large number of protection cases by suggesting some unifying principles. We selected a top-down approach to the problem, because we felt that previous research had basically exhausted the capacity of bottom-up approaches and that some questions were still unanswered.

Using the legal system for comparison, we supress most of the pitfalls of a top-down approach. The elements of classification described above would have been enough, but it happens that some transactions used in the legal system are largely usable in protection. Thus, providing a beginning of application of these elements of classification, further justifying their choice.

A reader might object to our claim that the set of representative transactions constitute a good proof of the applicability of our principles, since it is easy to invent transactions that do not seem to be reducible to the three types of transfer described. For example, "I give you a million dollars", does not seem to fit into this framework. First, the reality of this transaction is quite questionable although movies have been done about it. Second, we

feel that this does not really destroy our choice of concepts, since some rights are still involved, an owner can still be found, identification of the parties is required, identification of the object is obvious, delivery and acceptance too! Finally, we believe that the game of throwing counter-examples in the case of protection is uninteresting... what if somebody would unplug the computer!

We stated at the beginning of this dissertation, that protection is a matter of opinion, we want, in relation with the objection described above, to come back to this problem.

The analogy of the three levels of protection: the philosophy, the model and the implementation can be expressed again in the following terms. There is a high level human judgment involved in specifying and deciding protection. The same way there is a high level human judgement in interpreting and applying the law. There is a mechanistic application of rules to specific cases in protection and in the legal system. The specific cases being controlled do not involve human beings in the case of computers, whereas they do in the case of the legal system.

Thus it would seem that the level, at which the most important analogies can be made, is the level of the mechanistic application of rules, which we should note, could be encoded in both cases in a computer.

This description is perfectly acceptable. But we oppose the point of view about protection which reduce it to strictly mechanistic rules. Recognizing that the user's wishes may not be fulfilled but

still strongly opposed to the idea that a protection mechanism, worthy of the name, may not work in all cases.

We feel that requiring a protection mechanism to always work, puts an unacceptable burden on the user and is quite blind to the realities of its usage by human beings. Knowing that a perfect protection mechanism is a dream, a user would like to have mechanistic rules that work in most cases and the ability to intervene when the system fails.

Failures should be acceptable to a user if the product of their frequency and the cost associated with an occurrence represented a small burden, thus the idea developed in this dissertation that one should tailor the protection mechanisms to the consequences of a failure. The substance of this statement implies that high level human judgements are a part of protection.

The legal system is preoccupied with how to deal with failures. Thus, the analogies of protection and Real Estate Laws should provide a base for more work in this area.

BIBLIOGRAPHY

BCC 69 — BCC 500 System.
Various Internal Manuals, (1969).

BENSOUSSAN 72 — Bensoussan, A., Clingen, C.T. and Daley, R.C.
The Multics Virtual Memory: Concepts and Design
CACM. 15,5 (May 1972), 308-318.

BOWMAN 70 — Bowman, A.G.
Real Estate Law in California.
Prentice Hall, 3rd Edition (1970).

COSINE 71 — Cosine Committee
An Undergraduate Course on Operating System Principles.
Commission on Education of National Academy of Engineering (June 1971).

DENNIS 66 — Dennis, J.B. and Van Horn, E.C.
Programming Semantics for Multiprogrammed Computation.
CACM 9, 3 (March 1966), 143-155.

DIJKSTRA 68 — Dijkstra, E.W.
The Structure of the T.H.E. Multiprogramming System.
CACM 11, 5 (May 1968), 341-346.

FABRY 68 — Fabry, R.S.
Preliminary Description of a Supervisor for a Machine Oriented Around Capabilities.
Quarterly Progress Report 18, Institute of Computer Research, University of Chicago, (1968), 1-97.

GRAHAM GS 72 — Graham, G.S. and Denning, P.J.
Protection - Principles and Practice.
Proc. AFIPS 1972 SJCC, Vol. 40, AFIPS Press, Montvale, N.J., 417-429.

GRAHAM RM 68 — Graham, R.M.
Protection in an Information Processing Utility
CACM 11, 5 (May 1968), 365-369.

GRAY 72 — Gray, J., et al.
The Control Structure of an Operating System
Unpublished (1972).

HANSEN 70 Hansen, P.B.
The Nucleus of a Multiprogramming System
CACM 13, 4 (April 1970), 238-250.

JONES 73 Jones, A.K.
Protection in Programmed Systems
Ph.D. Thesis, Carnegie-Mellon University (June 1973).

LAMPSON 69a Lampson, B.W.
An Overview of CAL-TSS.
Computer Center, University of California, Berkeley (September 1969).

LAMPSON 69b Lampson, B.W.
Dynamic Protection Structures
Proc. AFIPS 1969 FJCC, Vol. 35, AFIPS Press, Montvale, N.J., 27-38.

LAMPSON 71 Lampson, B.W.
Protection
Proc. Fifth Annual Princeton Conference on Information Sciences and Systems, Department of Electrical Engineering, Princeton University, Princeton, N.J., (March 1971), 437-443.
[Chapter 6 of Cosine 71 is essentially identical to this paper.]

LAMPSON 73 Lampson, B.W.
A Note on the Confinement Problem
CACM 16, 10 (October 1973), 613-615.

MORRIS 73 Morris, J.H. Jr.
Protection in Programming Languages
CACM 16, 1 (January 1973), 15-21.

NEEDHAM 72 Needham, R.M.
Protection Systems and Protection Implementations
Proc. AFIPS 1972 FJCC, Vol. 41, AFIPS Press, Montvale, N.J., 571-578.

ORGANICK 72 Organick, E.I.
The Multics System: An Examination of its Structure.
MIT Press, 1972.

PARNAS 72 Parnas, D.L.
On the Criteria to be used in Decomposing Systems into Modules
CACM 15, 12 (December 1972), 1053-1058.

SALTZER 73
Saltzer, J.H.
Protection and the Control of Information Sharing in Multics.
Proc. Fourth Symposium on Operating System Principles (October 1973).

SCHROEDER 72a
Schroeder, M D. and Saltzer, J.h.
A Hardware Architecture for Implementing Protection Rings
CACM 15, 3 (March 1972), 157-170.

SCHROEDER 72b
Schroeder, M.D.
Cooperation of Mutually Suspicious Subsystems in a Computer Utility
Ph.D. Thesis, MAC Tr-104, MIT, (1972).

SCHROEDER 72c
Schroeder, M.D.
Seminar of the Electrical Engineering and Computer Science Department
University of California, Berkeley, (January 1972).

STURGIS 74
Sturgis, M.E.
A Postmortem for a Time Sharing System
Ph.D. Thesis, University of California, Berkeley
Document CSL 74-1, Xerox Palo Alto Research Center (January 1972).

WEBSTER 68
Webster's New Dictionary of Synonyms.
G. & C. Merriam Company, (1968).

WEINSTOCK 73
Weinstock, C.B.
A Survey of Protection Systems.
Computer Science Department, Carnegie-Mellon University, (1973).

WULF 73
Wulf, W.A., et al.
HYDRA: The Kernel of a Multiprocessor Operating System
Carnegie-Mellon University, Computer Science Department Report, (June 1973).