III. THE STATE OF THE ART

A. Implementations of Protection Mechanisms

Until quite recently, the protection of computer-stored information has been given relatively low priority by both the major computer manufacturers and a majority of their customers. Although research time-sharing systems using base and bound registers appeared as early as 1960 and Burroughs marketed a descriptor-based system in 1961, those early features were directed more toward preventing accidents than toward providing absolute interuser protection. Thus in the design of the IBM System/360, which appeared in 1964 [73], the only protection mechanisms were a privileged state and a protection key scheme that prevented writing in those blocks of memory allocated to other users. Although the 360 appears to be the first system in which hardware protection was also applied to the I/O channels, the early IBM software used these mechanisms only to the minimum extent necessary to allow accident free multiprogramming. Not until 1970 did "fetch protect" (the ability to prevent one user from reading primary memory allocated to another user) become a standard feature of the IBM architecture [74]. Recently, descriptor-based architectures, which can be a basis for the more sophisticated protection mechanisms described in Section II, have become common in commercially marketed systems and in most manufacturers' plans for forthcoming product lines. Examples of commercially available descriptor-based systems are the IBM System/370 models that support virtual memory, the Univac (formerly RCA) System 7, the Honeywell 6180, the Control Data Corporation Star-100, the Burroughs B5700/6700, the Hitachi 8800, the Digital Equipment Corporation PDP- 11/45, and the Plessey System 250. On the other hand, exploitation of such features for controlled sharing of information is still the exception rather than the rule. Users with a need for security find that they must improvise or use brute force techniques such as complete dedication of a system to a single task at a time [75]. The Department of Defense guide for safeguarding classified information stored in computers provides a good example of such brute force techniques [76].

In the decade between 1964 and 1974, several protection architectures were implemented as research and development projects, usually starting with a computer that provided only a privileged mode, adding minor hardware features and interpreting with software the desired protection architecture. Among these were M.I.T.'s CTSS which, in 1961, implemented user authentication with all-or-nothing sharing and, in 1965, added shared files with permission lists [12]. In 1967, the ADEPT system of the System Development Corporation implemented in software on an IBM System/360 a model of the U.S. military security system, complete with clearance levels, compartments, need-to-know, and centralized authority control [14]. At about the same time, the IBM Cambridge Scientific Center released an operating system named CP/67, later marketed under the

name VM/370, that used descriptor-based hardware to implement virtual System/360 computers using a single System/360 Model 67 [11]. In 1969, the University of California (at Berkeley) CAL system implemented a software-interpreted capability system on a Control Data 6400 computer [17]. Also in 1969, the Multics system, a joint project of M.I.T. and Honeywell, implemented in software and hardware a complete descriptor-based access control list system with hierarchical control of authorization on a Honeywell 645 computer system [26], [77]. Based on the plans for Multics, the Hitachi Central Research Laboratory implemented a simplified descriptor-based system with hardware-implemented ordered domains (rings of protection) on the HITAC 5020E computer in 1968 [78]. In 1970, the Berkeley Computer Corporation also implemented rings of protection in the BCC 500 computer [19]. In 1973, a hardware version of the idea of rings of protection together with automatic argument address validation was implemented for Multics in the Honeywell 6180 [63]. At about the same time, the Plessey Corporation announced a telephone switching computer system, the Plessey 250 [53], based on a capability architecture.

Current experimentation with new protection architectures is represented by the CAP system being built at Cambridge University [20] and the HYDRA system being built at Carnegie-Mellon University [21] . Recent research reports by Schroeder [70], Rotenberg [59], Spier et al. [79], and Redell [54] propose new architectures that appear practical to implement.

B. Current Research Directions

Experimentation with different protection architectures has been receiving less attention recently. Instead, the trend has been to concentrate in the following five areas: 1) certification of the correctness of protection system designs and implementations, 2) invulnerability to single faults, 3) constraints on use of information after release, 4) encipherment of information with secret keys, and 5) improved authentication mechanisms. These five areas are discussed in turn below.

A research problem attracting much attention today is how to certify the correctness of the design and implementation of hardware and software protection mechanisms. There are actually several sub-problems in this area.

a) One must have a precise model of the protection goals of a system against which to measure the design and implementation. When the goal is complete isolation of independent users, the model is straightforward and the mechanisms of the virtual machine are relatively easy to match with it. When controlled sharing of information is desired, however, the model is much less clear and the attempt to clarify it generates many

unsuspected questions of policy. Even attempts to model the well-documented military security system have led to surprisingly complex formulations and have exposed formidable implementation problems [14], [62] .

b) Given a precise model of the protection goals of a system and a working implementation of that system, the next challenge is to verify somehow that the presented implementation actually does what it claims. Since protection functions are usually a kind of negative specification, testing by sample cases provides almost no information. One proposed approach uses proofs of correctness to establish formally that a system is implemented correctly. Most work in this area consists of attempts to extend methods of proving assertions about programs to cover the constructs typically encountered in operating systems [52] .

c) Most current systems present the user with an intricate interface for specifying his protection needs. The result is that the user has trouble figuring out how to make the specification and verifying that he requested the right thing. User interfaces that more closely match the mental models people have of information protection are needed.

d) In most operating systems, an unreasonably large quantity of "system" software runs without protection constraints. The reasons are many: fancied higher efficiency, historical accident, misunderstood design, and inadequate hardware support. The usual result is that the essential mechanisms that implement protection are thoroughly tangled with a much larger body of mechanisms, making certification impossibly complex. In any case, a minimum set of protected supervisor functions--a protected kernel--has not yet been established for a full-scale modern operating system. Groups at M.I.T. [80] and at Mitre [81], [82] are working in this area.

Most modern operating systems are vulnerable in their reaction to hardware failures. Failures that cause the system to misbehave are usually easy to detect and, with experience, candidates for automatic recovery. Far more serious are failures that result in an undetected disabling of the protection mechanisms. Since routine use of the system may not include attempts to access things that should not be accessible, failures in access-checking circuitry may go unnoticed indefinitely. There is a challenging and probably solvable research problem involved in guaranteeing that protection mechanisms are invulnerable in the face of all single hardware failures. Molho [83] explored this topic in the IBM System 360/Model 50 computer and made several suggestions for its improvement. Fabry [84] has described an experimental "complete isolation" system in which all operating system decisions that could affect protection are duplicated by independent hardware and software.

Another area of research concerns constraining the use to which information may be put after its release to an executing program. In Section 1, we described such constraints as a fifth level of desired function. For example,

one might wish to "tag" a file with a notation that any program reading that file is to be restricted forever after from printing output on remote terminals located outside the headquarters building.

For this restriction to be complete, it should propagate with all results created by the program and into other files it writes. Information use restrictions such as these are common in legal agreements (as in the agreement between a taxpayer and a tax return preparing service) and the problem is to identify corresponding mechanisms for computer systems that could help enforce (or detect violations of) such agreements. Rotenberg explored this topic in depth [59] and proposed a "privacy restriction processor" to aid enforcement.

A potentially powerful technique for protecting information is to encipher it using a key known only to authorized accessors of the information. (Thus encipherment is basically a ticket-oriented system.) One research problem is how to communicate the keys to authorized users. If this communication is done inside the computer system, schemes for protecting the keys must be devised. Strategies for securing multinode computer communication networks using encipherment are a topic of current research; Branstad has summarized the state of the art [40] . Another research problem is development of encipherment techniques (sometimes called privacy transformations) for random access to data. Most well-understood enciphering techniques operate sequentially on long bit streams (as found in point-to-point communications, for example). Techniques for enciphering and deciphering small, randomly selected groups of bits such as a single word or byte of a file have been proposed, but finding simple and fast techniques that also require much effort to cryptanalyze (that is, with high work factors) is still a subject for research. A block enciphering system based on a scheme suggested by Feistel was developed at the IBM T. J. Watson Research Laboratory by Smith, Notz, and Osseck [38]. One special difficulty in this area is that research in encipherment encounters the practice of military classification. Since World War II, only three papers with significant contributions have appeared in the open literature [27], [39], [85]; other papers have only updated, reexamined, or rearranged concepts published many years earlier.

Finally, spurred by the need for better credit and check cashing authentication, considerable research and development effort is going into better authentication mechanisms. Many of these strategies are based on attempts to measure some combination of personal attributes, such as the dynamics of a handwritten signature or the rhythm of keyboard typing. Others are directed toward developing machine-readable identification cards that are hard to duplicate.

Work in progress is not well represented by published literature. The reader interested in further information on some of the current research projects mentioned may find useful the proceedings of two panel sessions at the 1974 National Computer Conference [86], [87], a recent workshop [88], and a survey paper [89].

C. Concluding Remarks

In reviewing the extent to which protection mechanisms are systematically understood (which is not a large extent) and the current state of the art, one cannot help but draw a parallel between current protection inventions and the first mass produced computers of the 1950's. At that time, by virtue of experience and strongly developed intuition, designers had confidence that the architectures being designed were complete enough to be useful. And it turned out that they were. Even so, it was quickly established that matching a problem statement to the architecture--programming--was a major effort whose magnitude was quite sensitive to the exact architecture. In a parallel way, matching a set of protection goals to a particular protection architecture by setting the bits and locations of access control lists or capabilities or by devising protected subsystems is a matter of programming the architecture. Following the parallel, it is not surprising that users of the current first crop of protection mechanisms have found them relatively clumsy to program and not especially well matched to the users' image of the problem to be solved, even though the mechanisms may be sufficient. As in the case of all programming systems, it will be necessary for protection systems to be used and analyzed and for their users to propose different, better views of the necessary and sufficient semantics to support information protection.

ACKNOWLEDGMENT

R. Needham, A. Jones, J. Dennis, J. P. Anderson, B. Lindsay, L. Rotenberg, B. Lampson, D. Redell, and M. Wilkes carefully reviewed drafts of the manuscript and offered technical suggestions. In addition, the preparation of this paper was aided by discussions with many people including H. Forsdick, P. Janson, A. Huber, V. Voydock, D. Reed, and R. Fabry. L. Schroeder ruthlessly edited out surplus jargon and prose inelegance.

SUGGESTIONS FOR FURTHER READING

The following short bibliography has been selected from the reference list to direct the reader to the most useful, up-todate, and significant materials currently available. Many of these readings have been collected and reprinted by L. J. Hoffman in [90]. The five bibliographies and collections (item 8 below) provide access to a vast collection of related literature.

1. Privacy and the impact of computers [1]-[3], [91], [92].
2. Case studies of protection systems [14], [17], [20], [26], [63], [83], [84].

3. Protected objects and protected subsystems [30], [45], [54], [59], [70]-[72].
4. Protection with encipherment [38]-[40], [93], [94].
5. Military security and nondiscretionary controls [82], [95], [96].
6. Comprehensive discussions of all aspects of computer security [6] - [8].
7. Surveys of work in progress [86]-[89] .
8. Bibliographies and collections on protection and privacy [90], [97]-[100].