The Protection of Information in Computer Systems

How to prepare a sample talk

Alastair Beresford
8 October 2018

PICS: a classic computer security paper

- First major survey of local system security
- MIT coauthors working on Multics
- Comms of the ACM 1973; Proc. IEEE 1975
- 2,600+ citations

PICS introduces or formalizes many ideas found in computer security

- What did you spot?

PICS introduces or formalizes many ideas found in computer security

- Integrity, confidentiality, availability; security vs. privacy
- Password protection and hashing; one-time passwords
- Psychology, human factors, and economics of security
- Software vulnerabilities; protecting the TCB
- Insider threat; electromagnetic leakage; physical security
- …

- Defines many ideas from **1970s local system security**

PICS: Explains state-of-the-art, imposes structure

- Define key terms clearly for the first time
- Where there is ambiguity or disagreement, select a definition – often with lasting effect
- Describe principles of protection
- Describe implementations
- Speculate about future directions

Implicitly: helps us understand the debates of the time, and the origins of many current ideas

related to hazard from lasers and other light sources," *Amer. J. Ophthalmol.*, vol. 66, p. 15, 1968.
[57] A. Vassiliadis, H. C. Zweng, N. A. Peppers, R. R. Peabody, and R. C. Honey, "Thresholds of laser eye hazards," *Arch. Environ. Health*, vol. 20, p. 161, 1970.
[58] P. W. Lappin, "Ocular damage thresholds for the helium-neon laser," *Arch. Environ. Health*, vol. 20, p. 177, 1970.
[59] W. T. Ham *et al.*, "Retinal burn thresholds for the He-Ne laser in the rhesus monkey," *Arch. Ophthalmol.*, to be published.
[60] T. P. Davis and W. J. Mautner, "Helium-neon laser effects on the eye," U.S. Army Med. Res. Develop. Com., Washington, D.C., Annu. Rep. Contr. DADA 17-69-C-9013, 1969.
[61] J. J. Vos, "Digital computations of temperature in retinal burn problems," Inst. Perception, Soesterberg, The Netherlands, RVO-TNO, Rep. IZF 1965016, 1963.
[62] M. A. Mainster, T. J. White, J. H. Tips, and P. W. Wilson, "Retinal-temperature increases produced by intense light sources," *J. Opt. Soc. Amer.*, vol. 60, p. 264, 1970.
[63] A. M. Clarke, W. T. Ham, W. J. Geeraets, R. C. Williams, and H. A. Mueller, "Laser effects on the eye," *Arch. Environ. Health*, vol. 18, p. 424, 1969.
[64] R. H. Stern and R. F. Sognnaes, "Laser beam on dental hard tissues," *J. Dent. Res.*, vol. 43, p. 873, 1964.
[65] R. H. Stern, "Dentistry and the laser," in *Laser Applications in Medicine and Biology*, vol. II, Dr. M. L. Wolbarsht, Ed. New York: Plenum, 1974, pp. 361-388.
[66] T. E. Gordon, Jr., and D. L. Smith, "Laser welding of prostheses—an initial report," *J. Prosth. Dent.*, vol. 24, p. 472, 1970.

The Protection of Information in Computer Systems

JEROME H. SALTZER, SENIOR MEMBER, IEEE, AND MICHAEL D. SCHROEDER, MEMBER, IEEE

Invited Paper

Abstract—This tutorial paper explores the mechanics of protecting computer-stored information from unauthorized use or modification. It concentrates on those architectural structures—whether hardware or software—that are necessary to support information protection. The paper develops in three main sections. Section I describes desired functions, design principles, and examples of elementary protection and authentication mechanisms. Any reader familiar with computers should find the first section to be reasonably accessible. Section II requires some familiarity with descriptor-based computer architecture. It examines in depth the principles of modern protection architectures and the relation between capability systems and access control list systems, and ends with a brief analysis of protected subsystems and protected objects. The reader who is dismayed by either the prerequisites or the level of detail in the second section may wish to skip to Section III, which reviews the state of the art and current research projects and provides suggestions for further reading.

GLOSSARY

THE FOLLOWING glossary provides, for reference, brief definitions for several terms as used in this paper in the context of protecting information in computers.

Access	The ability to make use of information stored in a computer system. Used frequently as a verb, to the horror of grammarians.
Access control list	A list of principals that are authorized to have access to some object.
Authenticate	To verify the identity of a person (or other agent external to the protection system) making a request.
Authorize	To grant a principal access to certain information.
Capability	In a computer system, an unforgeable ticket, which when presented can be taken as incontestable proof that the presenter is authorized to have access to the object named in the ticket.
Certify	To check the accuracy, correctness, and completeness of a security or protection mechanism.
Complete isolation	A protection system that separates principals into compartments between which no flow of information or control is possible.
Confinement	Allowing a borrowed program to have access to data, while ensuring that the program cannot release the information.
Descriptor	A protected value which is (or leads to) the physical address of some protected object.
Discretionary	(In contrast with *nondiscretionary*.) Controls on access to an object that may be changed by the creator of the object.
Domain	The set of objects that currently may be directly accessed by a principal.
Encipherment	The (usually) reversible scrambling of data according to a secret transformation key, so as to make it safe for transmission or storage in a physically unprotected environment.
Grant	To authorize (*q.v.*).
Hierarchical control	Referring to ability to change authorization, a scheme in which the record of

Manuscript received October 11, 1974; revised April 17, 1975.
The authors are with Project MAC and the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

- Is PICS an "original research contribution"?
  - Enumerates, organises, and explains the work of other researchers
  - But **structure** imposed on ideas is very exciting
  - PICS is often cited for the wrong reason – e.g., **Principle of Least Privilege**

- Useful to investigate citations to/from PICS

Structure of the paper

I. Glossary (1 page)
II. Basic Principles of Information Protection (11 pages)
III. Descriptor-Based Protection Systems (14 pages)
IV. References (2 pages)

- You cannot explain it all in 15-20 minutes!
- Instead select suitable subsets to focus on
- What are high-level motivations, principles?

	each authorization is controlled by another authorization, resulting in a hierarchical tree of authorizations.
List-oriented	Used to describe a protection system in which each protected object has a list of authorized principals.
Password	A secret character string used to authenticate the claimed identity of an individual.
Permission	A particular form of allowed access, e.g., permission to READ as contrasted with permission to WRITE.
Prescript	A rule that must be followed before access to an object is permitted, thereby introducing an opportunity for human judgment about the need for access, so that abuse of the access is discouraged.
Principal	The entity in a computer system to which authorizations are granted; thus the unit of accountability in a computer system.
Privacy	The ability of an individual (or organization) to decide whether, when, and to whom personal (or organizational) information is released.
Propagation	When a principal, having been authorized access to some object, in turn authorizes access to another principal.
Protected object	A data structure whose existence is known, but whose internal organization is not accessible, except by invoking the protected subsystem (*q.v.*) that manages it.
Protected subsystem	A collection of procedures and data objects that is encapsulated in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the protected subsystem and the procedures may be called only at designated domain entry points.
Protection	1) Security (*q.v.*). 2) Used more narrowly to denote mechanisms and techniques that control the access of executing programs to stored information.
Protection group	A principal that may be used by several different individuals.
Revoke	To take away previously authorized access from some principal.
Security	With respect to information processing systems, used to denote mechanisms and techniques that control who may use or modify the computer or the information stored in it.
Self control	Referring to ability to change authorization, a scheme in which each authorization contains within it the specification of which principals may change it.
Ticket-oriented	Used to describe a protection system in which each principal maintains a list of unforgeable bit patterns, called tickets, one for each object the principal is authorized to have access.
User	Used imprecisely to refer to the individual who is accountable for some identifiable set of activities in a computer system.

I. Basic Principles of Information Protection

A. Considerations Surrounding the Study of Protection

1) General Observations: As computers become better understood and more economical, every day brings new applications. Many of these new applications involve both storing information and simultaneous use by several individuals. The key concern in this paper is multiple use. For those applications in which all users should not have identical authority, some scheme is needed to ensure that the computer system implements the desired authority structure.

For example, in an airline seat reservation system, a reservation agent might have authority to make reservations and to cancel reservations for people whose names he can supply. A flight boarding agent might have the additional authority to print out the list of all passengers who hold reservations on the flights for which he is responsible. The airline might wish to withhold from the reservation agent the authority to print out a list of reservations, so as to be sure that a request for a passenger list from a law enforcement agency is reviewed by the correct level of management.

The airline example is one of protection of corporate information for corporate self-protection (or public interest, depending on one's view). A different kind of example is an on-line warehouse inventory management system that generates reports about the current status of the inventory. These reports not only represent corporate information that must be protected from release outside the company, but also may indicate the quality of the job being done by the warehouse manager. In order to preserve his personal privacy, it may be appropriate to restrict the access to such reports, even within the company, to those who have a legitimate reason to be judging the quality of the warehouse manager's work.

Many other examples of systems requiring protection of information are encountered every day: credit bureau data banks; law enforcement information systems; time-sharing service bureaus; on-line medical information systems; and government social service data processing systems. These examples span a wide range of needs for organizational and personal privacy. All have in common controlled sharing of information among multiple users. All, therefore, require some plan to ensure that the computer system helps implement the correct authority structure. Of course, in some applications no special provisions in the computer system are necessary. It may be, for instance, that an externally administered code of ethics or a lack of knowledge about computers adequately protects the stored information. Although there are situations in which the computer need provide no aids to ensure protection of information, often it is appropriate to have the computer enforce a desired authority structure.

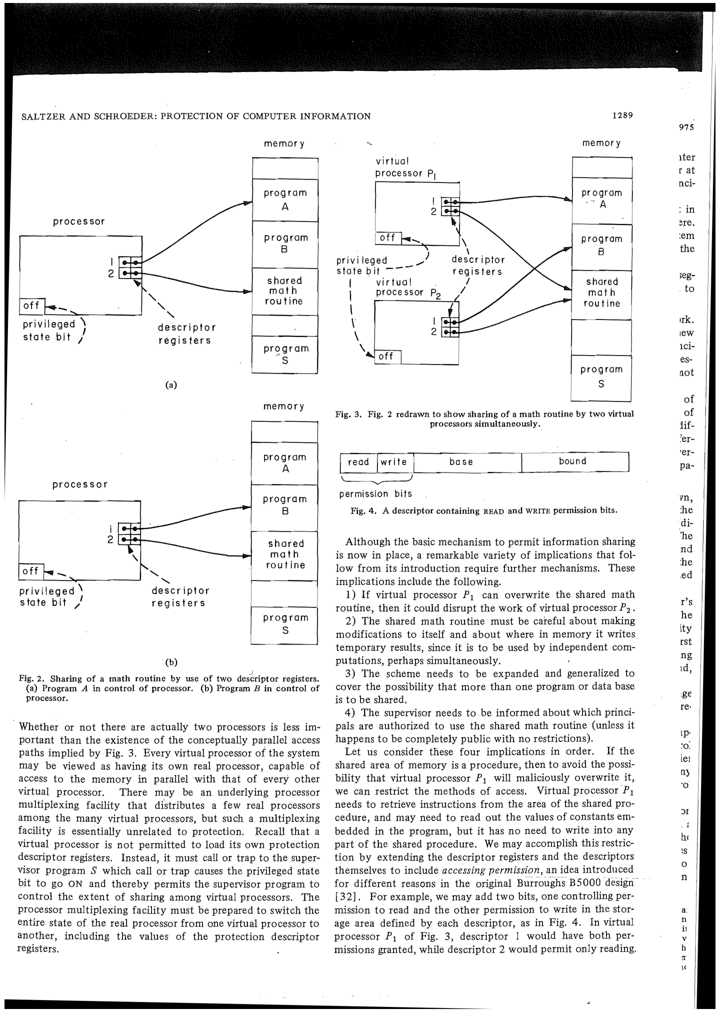
The words "privacy," "security," and "protection" are frequently used in connection with information-storing systems. Not all authors use these terms in the same way. This paper uses definitions commonly encountered in computer science literature.

The term "privacy" denotes a socially defined ability of an individual (or organization) to determine whether, when, and

PICS Glossary

- Terms cleanly defined for the first time
- Terms we recognise:
  - *Access control list*
  - *Authenticate*
- Terms we might not:
  - *Descriptor*
  - *List-oriented*
- Do all the terms mean the same thing today?

PICS I. Basic Principles of Information Protection



SALTZER AND SCHROEDER: PROTECTION OF COMPUTER INFORMATION 1289

(a)

(b)

Fig. 2. Sharing of a math routine by use of two descriptor registers. (a) Program *A* in control of processor. (b) Program *B* in control of processor.

Whether or not there are actually two processors is less important than the existence of the conceptually parallel access paths implied by Fig. 3. Every virtual processor of the system may be viewed as having its own real processor, capable of access to the memory in parallel with that of every other virtual processor. There may be an underlying processor multiplexing facility that distributes a few real processors among the many virtual processors, but such a multiplexing facility is essentially unrelated to protection. Recall that a virtual processor is not permitted to load its own protection descriptor registers. Instead, it must call or trap to the supervisor program *S* which call or trap causes the privileged state bit to go ON and thereby permits the supervisor program to control the extent of sharing among virtual processors. The processor multiplexing facility must be prepared to switch the entire state of the real processor from one virtual processor to another, including the values of the protection descriptor registers.

Fig. 3. Fig. 2 redrawn to show sharing of a math routine by two virtual processors simultaneously.

Fig. 4. A descriptor containing READ and WRITE permission bits.

Although the basic mechanism to permit information sharing is now in place, a remarkable variety of implications that follow from its introduction require further mechanisms. These implications include the following.

1) If virtual processor P_1 can overwrite the shared math routine, then it could disrupt the work of virtual processor P_2.

2) The shared math routine must be careful about making modifications to itself and about where in memory it writes temporary results, since it is to be used by independent computations, perhaps simultaneously.

3) The scheme needs to be expanded and generalized to cover the possibility that more than one program or data base is to be shared.

4) The supervisor needs to be informed about which principals are authorized to use the shared math routine (unless it happens to be completely public with no restrictions).

Let us consider these four implications in order. If the shared area of memory is a procedure, then to avoid the possibility that virtual processor P_1 will maliciously overwrite it, we can restrict the methods of access. Virtual processor P_1 needs to retrieve instructions from the area of the shared procedure, and may need to read out the values of constants embedded in the program, but it has no need to write into any part of the shared procedure. We may accomplish this restriction by extending the descriptor registers and the descriptors themselves to include *accessing permission*, an idea introduced for different reasons in the original Burroughs B5000 design [32]. For example, we may add two bits, one controlling permission to read and the other permission to write in the storage area defined by each descriptor, as in Fig. 4. In virtual processor P_1 of Fig. 3, descriptor 1 would have both permissions granted, while descriptor 2 would permit only reading.

- A smorgasbord of amazing ideas!
- Considerations
  - Privacy vs. security vs. protection
  - Confidentiality, integrity, availability
- Levels of protection
  - Unprotected, controlled sharing, ...
- Design principles
  - Who can remember these?
- Technical underpinnings
  - E.g., implementing isolation, supervisor mode, passwords

The eight design principles

- Can you remember what these were?

The eight design principles

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design
- Separation of privilege
- Least privilege
- Least common mechanism
- Psychological acceptability

PICS II. Descriptor-Based Protection Systems

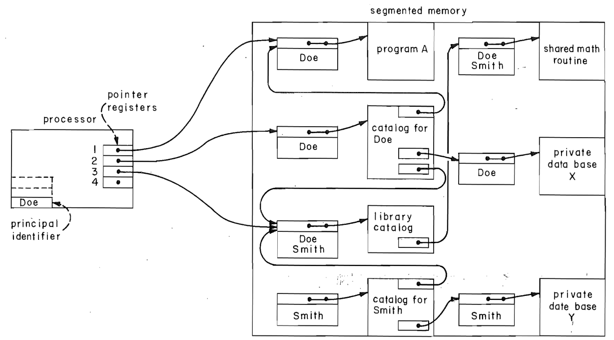


Fig. 10. A protection system using access controllers containing access control lists. In this system, every segment has a single corresponding access controller with its own unique identifier for addressing purposes; pointer registers always contain the unique identifiers of access controllers. Program *A* is in control of the processor, and it has already acquired a pointer to the library catalog. Since the access control list in the access controller for the library catalog contains Doe's name, the processor can use the catalog to find the pointer for the shared math routine. Since his name also appears in the access control list of the math routine, the processor will then be able to use the shared math routine.

of any desired grouping for protection purposes. Thus, in Fig. 10, a library catalog has been introduced.

It is also apparent that implementation, especially direct hardware implementation, of the access control list system could be quite an undertaking. We will later consider some strategies to simplify implementation with minimum compromise of functions, but first it will be helpful to introduce one more functional property–protection groups.

2) Protection Groups: Cases often arise where it would be inconvenient to list by name every individual who is to have access to a particular segment, either because the list would be awkwardly long or because the list would change frequently. To handle this situation, most access control list systems implement factoring into *protection groups*, which are principals that may be used by more than one user. If the name of a protection group appears in an access control list, all users who are members of that protection group are to be permitted access to that segment.

Methods of implementation of protection groups vary widely. A simple way to add them to the model of Figs. 9 and 10 is to extend the "principal holding" register of the processor so that it can hold two (or more) principal identifiers at once, one for a personal principal identifier and one for each protection group of which the user is a member. Fig. 10 shows this extension in dashed lines. In addition, we upgrade the access control list checker so that it searches for a match between any of the principal identifiers and any entries of the access control list.[35] Finally, who is allowed to use those principals that represent protection group identifiers must also be controlled systematically.

We might imagine that for each protection group there is a protection group list, that is, a list of the personal principal identifiers of all users authorized to use the protection group's principal identifier. (This list is an example of an access control list that is protecting an object—a principal identifier—other than a segment.) When a user logs in, he can specify the set of principal identifiers he proposes to use. His right to use his personal principal identifier is authenticated, for example, by a password. His right to use the remaining principal identifiers can then be authenticated by looking up the now-authenticated personal identifier on each named protection group list. If everything checks, a virtual processor can safely be created and started with the specified list of principal identifiers.[36]

3) Implementation Considerations: The model of a complete protection system as developed in Fig. 10 is one of many possible architectures, most of which have essentially identical functional properties; our choices among alternatives have been guided more by pedagogical considerations than by practical implementation issues. There are at least three key areas in which a direct implementation of Fig. 10 might encounter practical problems.

1) As proposed, every reference to an object in memory requires several steps: reference to a pointer register; indirect

[35] If there is more than one match, and the multiple access control list entries specify different access permissions, some resolution strategy is needed. For example, the INCLUSIVE-OR of the individually specified access permissions might be granted.

[36] In some systems (notably CAL TSS [17]), principal identifiers are treated as a special case of a capability, known as an access key, that can be copied about, stored anywhere, and passed on to friends. Although this approach appears to produce the same effect as protection groups, accountability for the use of a principal identifier no longer resides in an individual, since any holder of a key can make further copies for his friends.

- Make it practical with worked examples
- Starts with "descriptor and virtual memory systems" and "tagged capabilities"
- Builds up to access control – e.g., segments (files) in a persistent storage system

PICS III. The State of the Art

1304 PROCEEDINGS OF THE IEEE, SEPTEMBER 1975

We now have a controlled domain entry facility. A user wishing to provide a protected subsystem can do so by setting the access control lists of all objects that are to be internal parts of the system to contain one of his own principal identifiers. He also adds to the access control list of the initial procedure of his subsystem ENTER permission for any other principals who are allowed to use his protected subsystem.

In a capability system, a similar addition produces protected subsystems. The permission field of a capability is extended to include ENTER permission, and when a capability is used as the target of a GO TO or a CALL instruction, control is passed to the procedure in the segment pointed to by the capability. Simultaneous with passing control to the procedure, the processor switches on the READ permission bit of the capability, thereby making available to the virtual processor a new domain—all those objects that can be reached starting from capabilities found in the procedure.

Two mechanisms introduced earlier can now be seen to be special cases of the general domain entry. In the initial discussion of the capability system, we noted that the authentication system starts a new user by allowing a virtual processor to enter that user's domain at a controlled starting point. We could use the domain entry mechanism to accomplish this result as follows. A system program is "listening" to all currently unused terminals or system ports. When a user walks up to a terminal and attempts to use it, the system program creates a new virtual processor and has that processor ENTER the domain named by the prospective user. The entry point would be to a program, perhaps supplied by the user himself, which authenticates his identity before doing any other computation. Because a protected subsystem has been used, the program that monitors the unused terminals does not have access to the data in the protected subsystem (in contrast with the system of Fig. 7), a situation in better accord with the principle of least privilege. Instead, it has an enter capability for every domain that is intended to be entered from a terminal, but that capability leads only to a program that demands authentication.

We have sketched only the bare essentials of the mechanism required to provide domain switching. The full mechanics of a practical system that implements protected objects and subsystems are beyond the scope of this tutorial, but it is useful to sketch quickly the considerations those mechanisms must handle.

1) The principle of "separation of privilege" is basic to the idea that the internal structure of some data objects is accessible to virtual processor *A*, but only when the virtual processor is executing in program *B*. If, for example, the protection system requires possession of two capabilities before it allows access to the internal contents of some objects, then the program responsible for maintenance of the objects can hold one of the capabilities while the user of the program can hold the other. Morris [72] has described an elegant semantics for separation of privilege in which the first capability is known as a *seal*. In terms of the earlier discussion of types, the type field of a protected object contains a seal that is unique to the protected subsystem; access to the internal structure of an object can be achieved only by presenting the original seal capability as well as the capability for the object itself. This idea apparently was suggested by H. Sturgis. The HYDRA and CAL systems illustrate two different implementations of this principle.

2) The switching of protection domains by a virtual processor should be carefully coordinated with the mechanisms that provide for dynamic activation records and static (own) variable storage, since both the activation records and the static storage of one protection domain must be distinct from that of another. (Using a multiple virtual processor implementation provides a neat automatic solution to these problems.)

3) The passing of arguments between domains must be carefully controlled to ensure that the called domain will be able to access its arguments without violating its own protection intentions. Calls by value represent no special problem, but other forms of argument reference that require access to the original argument are harder. One argument that must be especially controlled is the one that indicates how to return to the calling domain. Schroeder [70] explored argument passing in depth from the access control list point of view, while Jones [71] explored the same topic in the capability framework.

The reader interested in learning about the mechanics of protected objects and subsystems in detail is referred to the literature mentioned above and in the Suggestions for Further Reading. This area is in a state of rapid development, and several ideas have been tried out experimentally, but there is not yet much agreement on which mechanisms are fundamental. For this reason, the subject is best explored by case study.

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

- Brief section
  - On-going research and industrial projects
  - Bemoans the lack of publication of many exciting ideas by industry
- Future research directions
  - E.g., in certification, verification, human factors, TCB minimization
  - Information flow control, relationship to crypto

What doesn’t the paper talk about?

What doesn’t the paper talk about?

- “Out of scope” – but mentioned
  - Attacker models, EM leakage
  - Cryptography, cryptographic protocols
- Things since the 1970s
  - Ubiquitous computer networking
  - anonymous users, wireless, crypto advances, …
  - Network vulnerabilities
  - Current focus on “vulnerability mitigation”
  - Progress on formal verification
  - Programming-language security
  - Mobile and cyber-physical systems

If we were to write the same survey today, what would we focus on?

Possible talk structure

1.	Historical context: who, what, why?	1 minute
2.	Key definitions – and resolving ambiguities	3
	– E.g., protection vs. security vs. privacy	
3.	Ideas that foreshadow later things; e.g.,	3
	– Tamper/EM-related attack models	
	– Biometrics and authentication	
	– Economics and psychology	
4.	Exploration of "levels" of system designs	4
	– Unprotected systems	
	…	
	– User-programmed sharing	
5.	ACLs vs. capabilities in descriptor systems	2
6.	Papers cited – who/what/where?	1
7.	Work that cites PICs – who/what/where?	1
8.	What was missed / ideas invalidated?	2
		-----
		17 minutes