Secure Personal Computing in an Insecure Network

Dorothy E. Denning

Report Number:
78-270

2448

SECURE PERSONAL COMPUTING IN AN INSECURE NETWORK (1)

Dorothy E. Denning (2)
Purdue University

July 1978

CSD-TR-270

Abstract

A method for implementing secure personal computing in a network with one or more central facilities is described. The method employs a public-key encryption device and hardware keys. Each user is responsible for his own security and need not rely on the security of the central facility or the communication links.

(1) This work was supported in part by NSF Grant MCS77-04835.

(2) Computer Science Department, Purdue University, W. Lafayette, IN 47907.

Introduction

Within the next ten years many of us will have personal computers linked to a central facility.[1] The central facility (CF) will offer many attractive features: long term storage, text editors, language processors, special purpose software, video games, access to large data banks, and electronic mail for communication among users on the network. The CF could also pose a serious threat: any or all of the secrets we entrust with it could be stolen without our even being aware of the theft. Personal communication sent over the network could be intercepted; files stored in the CF could be copied; booby-trapped software borrowed from the CF and run on our personal computers could transmit confidential data back to its owner via the CF.

This paper describes a simple method for safeguarding personal data in the network. The method evolved from consideration of three basic premises: user responsibility, possible security flaws in the CF, and limited sharing of confidential information among users.

The first premise is that each user should be responsible for the security of his electronic possessions, just as he is for his other possessions. He should be able to protect his electronic possessions to the same degree and with the same precautions as he protects his other possessions. For example, several options are available for safeguarding

1. Although I shall assume there is a single central facility, the security mechanisms described here apply equally to networks with multiple central facilities.

jewelry or important papers: an unlocked drawer, a locked cabinet, a steel vault, a safe deposit box, etc. One evaluates the risks, cost, and inconvenience of each option to select the most suitable alternative. Likewise, I propose a system in which each user can select safeguards for computer files and communication with roughly similar risks, costs, and convenience. It is important that the user feel confident in understanding the limitations of the safeguards he selects.

The second premise is that a user should not have to rely on or trust the CF or the communication links of the network for the safety of his data. The proof of a complex CF should not be a prerequisite for security to the customers. Even if the CF could be proven secure, there would be no guarantee that its specifications were complete or that an unsuspected compromise could not occur. However, there are strong economic reasons for the designers of the CF to build a secure and reliable system. An unreliable or insecure CF will lose its customers: no user will entrust a CF with files or mail that are subject to accidental (or intential) loss or destruction. But whereas a user can recognize the loss or destruction of his data, he cannot recognize its theft. Nevertheless, the customers of the CF must feel that their personal data cannot be stolen even in the presence of hardware faults, software errors, or malicious attacks.

The third premise is that sharing of confidential information among users of the CF is limited. In MULTICS, for example, whose design is based on sharability and whose philosophy encourages sharing, there is in fact little inter-user sharing [Mon77]. Consequently, users can share copies of confidential files rather than originals without straining the resources of the CF. Users can share originals of nonconfidential files, however.

My proposal places the responsibility for safeguarding personal data on the owner. The security of data stored in the CF or transmitted through the CF does not depend on the security or correctness of the CF or the communication links. The principal mechanism is a public-key encryption device and hardware keys. The mechanism allows a user to protect personal data to the extent that he protects the hardware unit containing his secret key. The method differs from those described by Popek and Kline [PoK78] and Needham and Schroeder [NeS77] in that both of these approaches rely on the security and correctness of the network, principally its key management facilities.

The basic idea was inspired by Tannenbaum's paper [Tan77], which describes a distributed interactive system wherein each user has his own dedicated LSI microcomputer. These "personal computers" are each connected to a central minicomputer, which provides file storage and software. To run a program, a user submits a request to the central machine, which then sends a copy of the program to the user's computer for execution.

Tanenbaum's design has considerable merit. The central operating system is considerably less complex than is customary for large centralized time-sharing systems. The system supports a heterogeneous network of microcomputers, enabling it to take advantage of advances in microcomputer technology. Although Tanenbaum does not discuss data security, the principle feature of user isolation provides a good basis for implementing security. The encryption scheme proposed in this paper could complete the design. I am currently investigating the security properties of a system patterned after Tannenbaum's design.

The following section outlines the mechanism. Subsequent sections describe how the mechanism solves three important security problems: personal secrecy, secure communications and sharing, and secure signatures. A final section outlines the requirements of the interface with the CF. In the interest of conveying the basic ideas, I have omitted many details.

The Security Mechanism

The security mechanism consists of an encryption device, which connects to a personal computer, and hardware keys. The encryption device implements a public-key encryption algorithm as proposed by Diffie and Hellman [DiH76] and further investigated by Rivest, Shamir, and Adleman [RSA78]. Under public-key encryption, a plaintext message is enciphered using a public key P and deciphered using a secret (or private) key S.[2] The encipher and decipher algorithms are inverse algorithms over the same message space and are publicly available.

Let X^Y denote the enciphering or deciphering of X with key Y. For a given plaintext message M, the corresponding ciphertext C is related to M by the relations:

$$C = M^P \quad \text{and} \quad P = C^S.$$

Furthermore, for either plaintext or ciphertext message X,

$$(X^P)^S = (X^S)^P = X.$$

There are two important security properties of public-key encryption. First, given a ciphertext C, it is computationally infeasible to compute the corresponding plaintext message $M = C^S$ without knowledge of the secret key S. Second, given a public key P, it is computationally infeasible to compute the corresponding secret key S. Thus public keys can be freely distributed without risking the security of enciphered data.

2. The notation used here follows that suggested in [NeS77] rather than that in [DiH76,RSA78], wherein "E" and "D" denote encryption and decryption transformations, respectively.

A pair of hardware keys (memory chips) implement a (public,secret) key pair. The owner of the keys is told what character sequence is "burned in" the memory chip of the public "P-key" so that he can tell it to his associates. However, the sequence "burned in" the memory chip of the secret "S-key" is not revealed to anyone, including the key's owner.[3]

The encryption device connects to a users personal computer (PC) and has separate sockets for the P and S keys. In addition, it has a facility for setting an alternative "soft" public key, ALT-P, and a toggle switch for selecting between the P and ALT-P keys. The encryption device and hardware keys could be built as a single unit. However, it is essential that the device containing the memory chip for the S-key be detachable fom the PC and be sufficiently small that the user can protect it as he would any other key.

Figure 1 shows how the device would be used to encipher and decipher data transmitted between a user's PC and the CF. A message X originating from the user's PC is enciphered with the P-key (or the ALT-P key) before it is transmitted to the CF. A message Y originating from the CF is deciphered with the S-key when it is received. All data transmitted between a user's PC and the CF must pass through the encryption device. No other communication lines between a PC and the CF are permitted, thus assuring a user that his confidential data is properly enciphered and deciphered.

The purpose of the toggle switch and soft public key is to enable a user to transmit messages to the CF and other users on the network. When

3. By keeping the S-key secret, it is more difficult for someone to duplicate it.

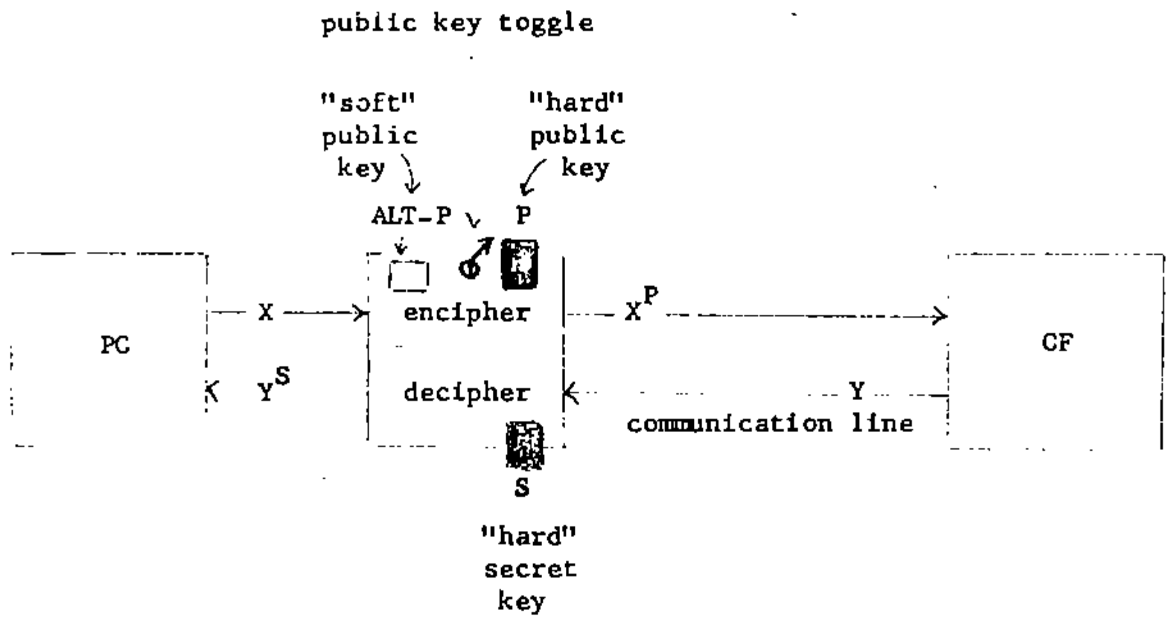

Figure 1. The Encryption Mechanism.

the toggle is set ot the P-key, information transmitted from the user's PC cannot be deciphered by anyone but the user. In order to transmit messages to the CF or another user, the sender must set his toggle to ALT-P and supply the receiver's public key. The receiver's key could be supplied, for example, by a command issued from the user's PC. Moreover, while the toggle is set to ALT-P, software running on the PC can set ALT-P to be the same as P, freeing the user of the need to reset the toggle to use his personal public key. However, this is less secure than enciphering with the hard P-key, as the user must rely on (possibly borrowed) software to supply the correct key.

An important property of this mechanism is that the CF does not keep a record of secret keys. The key manufacturer may keep records of keys in order to handle lost or stolen keys, but these could be securely stored in a steel vault. It is primarily for this reason this system uses public-key encryption rather than single-key encryption, such as the Data Encryption Standard (DES) [NBS77]. Under single-key encryption, the same secret key is used both for encryption and decryption, making it necessary for the CF to maintain and safeguard lists of secret keys. In order for the CF to transmit ciphertext to a user, it must know the user's secret key. In order for two users to communicate, the CF must generate and distribute a secret communication key. Thus, the security of the system depends on the security of the key management facilities, violating the premise that the users should not have to rely on the security mechanisms of the CF.

A user may reveal his public key to the CF, however. Since all information arriving at his personal computer is automatically deciphered with his S-key, it also must be enciphered with his P-key in order to appear in plaintext in his PC. If the user reveals his public key to the CF, the CF can, for example, encipher programs requested by the user before transmitting them to the user's PC.

However, it is not necessary for a user to give his public key to the CF, as the user can perform the encryption himself. This works as follows: the CF transmits plaintext message X, which is then deciphered upon arrival at the user's PC, giving X^S. The user then routes X^S back through his encryption device to get $(X^S)^P = X$ (see Figure 2). Therefore, a user need give his P-key to the CF only if he wishes the CF to transmit data in ciphertext and the data has not been previously enciphered (either by the user himself or some other user).

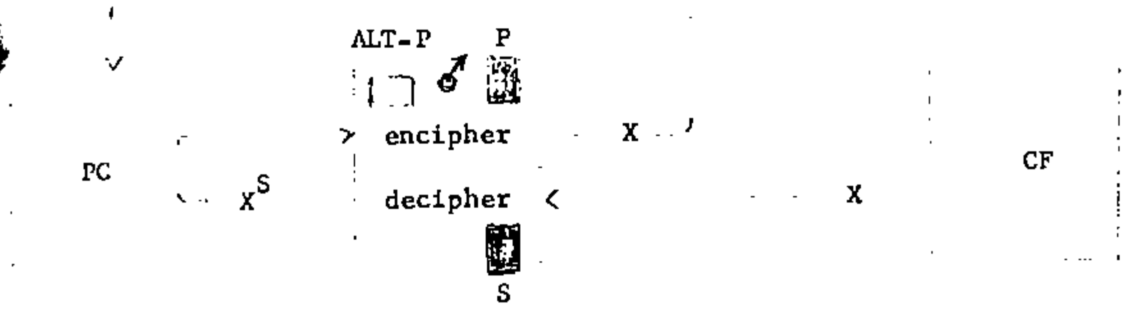

Figure 2. Transmission of Plaintext X from Central Facility to User's Personal Computer.

Personal Secrecy

To implement personal secrecy, the user sets the toggle switch of his encryption device to his public P-key. Since all information transmitted from his PC is automatically enciphered using his P-key, it is computationally infeasible for anyone to decipher information outside of his PC without acquiring his secret S-key. But the S-key is engraved in a memory chip and there is nc copy of it in the CF; thus a perpetrator must steal or duplicate it in order to decipher the data.[4]

With this mechanism, a user can safely store (enciphered) secret documents in the CF. No perpetrator will be able to break into the CF and decipher the documents.

A user could safely run software supplied by the CF on his PC without fear of a "trojan horse" theft. If the software package attempted to transmit the user's data back to its owner, the data would be automatically enciphered with the user's key, rendering it useless to the owner of the package.

Consequently, a compiler, for example, could not steal proprietary software under development, or an income tax program could not steal confidential financial records. The mechanism can thus be used to implement confined (or memoryless) subsystems [Lam71].[5] The mechanism does not,

4. Alternatively, it may be possible for a perpetrator to rig an encryption device to record secret keys. If this posed a serious threat, it would be necessary for a user to safeguard the encryption device he used as well as the key. Thus there is some advantage to a single device containing both the encryption algorithms and the memory chips implementing the keys.

5. However, it may be possible to leak information ov "covert channels"; e.g., by encoding it in the rate or quantity of transmitted ciphertext.

however, safeguard data supplied (in plaintext) to programs run at the CF. To safeguard data in this case requires sophisticated protection mechanisms within the CF.

Secure Communication and Sharing

Secure communication is achieved with end-to-end encryption; that is, the sender enciphers the message before transmission and the receiver deciphers the message upon receipt. Suppose users A and B wish to be able to communicate securely through the CF. This is easily done if A and B exchange their public keys P_A and P_B respectively. As suggested by Diffie and Hellman [DiH76], A sends messages enciphered with P_B to B; similarly B sends messages enciphered with P_A to A (see Figure 3). Secure one- way communication is achieved if either A or B transmits data enciphered with the other's public key.

There is clearly no danger of an intruder intercepting and deciphering messages transmitted this way. To guard against the problem of replay, a sequence number or time stamp can be inserted into a message before it is enciphered.

The method can also be used to implement sharing of confidential files. Suppose user A has a confidential file F stored in the CF and enciphered under P_A. To share F with another user B, A requests a copy of F from the CF. Since F is automatically deciphered under S_A when it reaches A's PC, A has only to send it back to the CF enciphered under P_B in order that B, and only B, be able to decipher it (see Figure 4). (A must also instruct the CF to add this new version of F to B's file directory.) Should A update F and wish to share the updated version with B, the process would be repeated.

The important point in both cases is that all confidential information travelling through the network or stored in the CF is enciphered. At no time does the CF have access to plaintext or to the secret keys required to decipher the information.

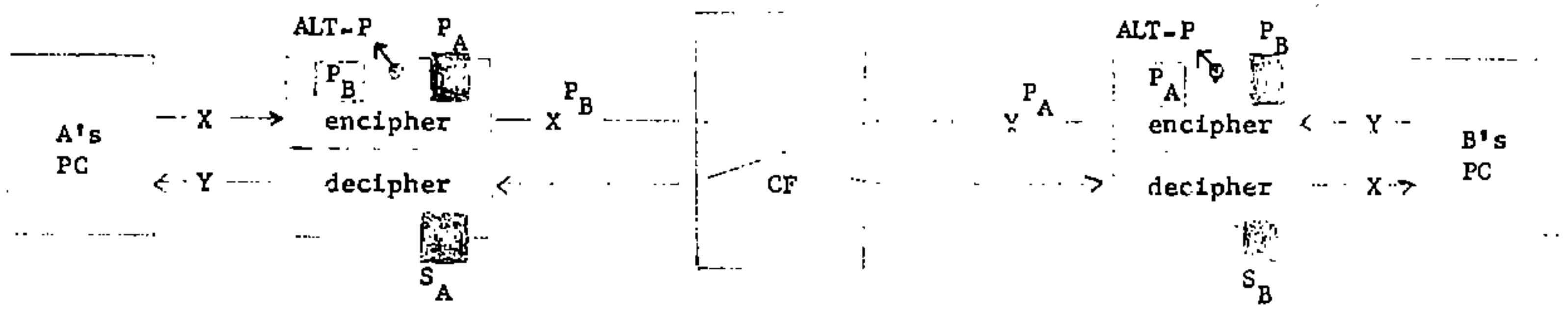

Figure 3. Secure Communication Between Two Users A and B.

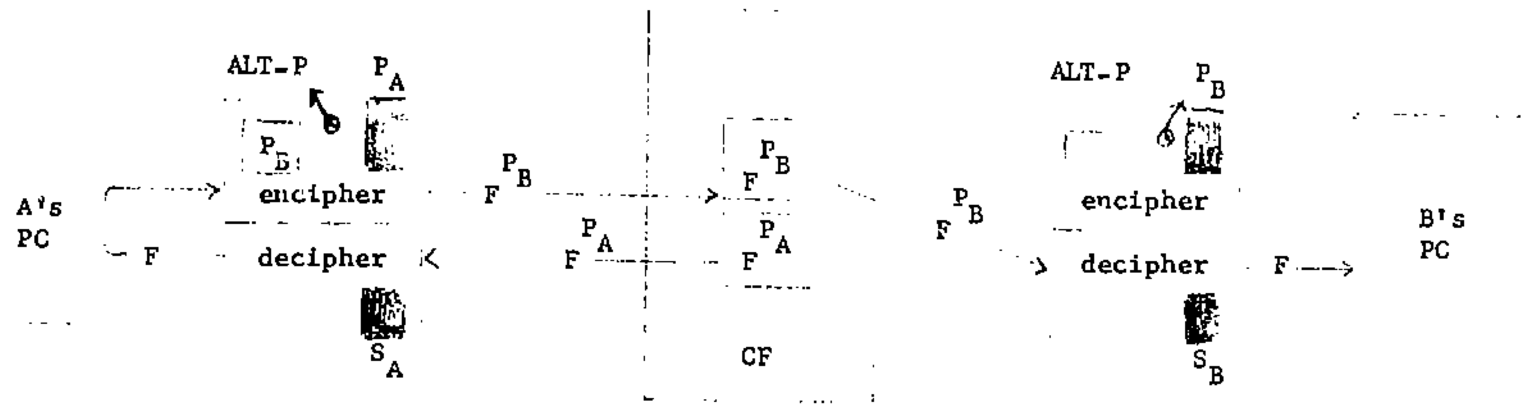

Figure 4. User A Shares Confidential File F with User B.

Secure Signatures

The method can be used to implement secure signatures as described by Diffie and Hellman [DiH76]: to send a signed message X to B, A first deciphers X with his secret key S_A before transmitting it, enciphered under B's public key P_B. When B receives the message, it is automatically deciphered under S_B, so that B has only to encipher it under P_A to obtain the original message and know that it came from A (since only A could have enciphered it under S_A) (see Figure 5).

As outlined above, the method suffers from the problem pointed out by Saltzer [Sal78]: B has no assurance that A did not loan, give away, or lose his private key. This problem could be solved with more sophisticated hardware. When a user purchased a (P,S) key pair, his voice-print (or some other identifying characteristic) could be recorded in the S-key. Activation of the S-key would than somehow require the user to supply a matching print. With this device, a signature could also be used to authenticate a user to the CF.

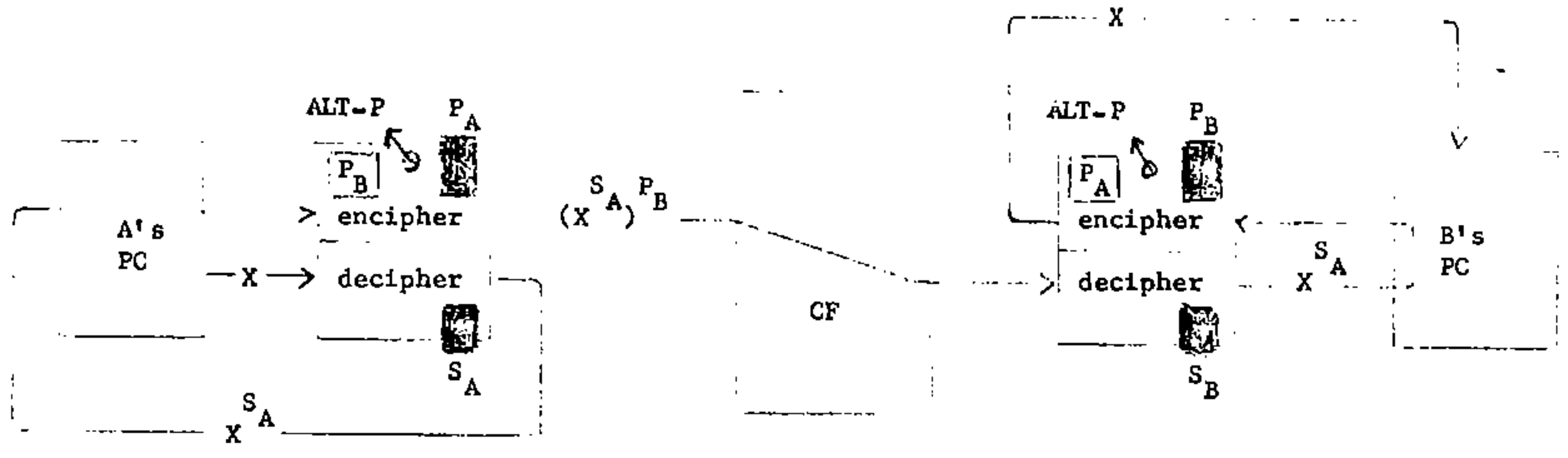

Figure 5. Message X Securely Signed by A and Transmitted to B.

Interface with Central Facility

The CF is also equiped with a pair of keys and one or more encryption devices. All incoming messages may be automatically deciphered, although this is not required for security. However, the CF must identify and decipher messages addressed to it. This could be done by prefixing messages to the CF with a fixed-format header identifying the CF, the sender, and the time of transmission (to guard against replay). Upon receipt of a message, the CF would attempt to decipher the beginning of the message. If the message begins with a recognizable header, the CF would continue deciphering the message; otherwise, the CF would simply route the message, in ciphertext, as directed by a previous command. For example, a user A wishing to store a confidential file at the CF would first send a request, properly headed and enciphered under D_{CF}. This would be followed by the file, enciphered under D_A.

The CF may provide "directory assistance" for public keys of its customers. If a user wished to make his key generally available, he could list it with the CF. Alternatively, a user can have an unlisted key and personally give it to his associates.

There is some risk associated with obtaining keys from the directory. If the CF sends an incorrect key (either accidently or intentionally), the recipient of the key may unknowingly encipher confidential messages that are decipherable to a perpetrator rather than his associate!

Summary

I have outlined a scheme for implementing secure personal computing in a large network with one or more central facilities. The scheme is based on the use of a public-key encryption device and hardware keys in the form of memory chips. Each user is responsible for protecting his own data and need not rely on the security of the network. All confidential data is enciphered before it is transmitted to the central facility or another user on the network. The central facility is not responsible for enciphering or deciphering data and, therefore, is not given access to either plaintext or the secret keys needed to decipher the ciphertext.

Acknowledgements

I am grateful to Peter Denning for contributing his usual assortment of insights, suggestions, and encouragement.

References

DiH76 Diffie, W. and Hellman, M., "New Directions in Cryptography," IEEE Trans. on Info. Theory, IT-22, 6, (Nov. 1976), 644-654.

Lam71 Lampson, B. W., "A Note on the Confinement Problem," Comm. ACM, 16, 10, (Oct. 1973), 613-615.

NeS77 Needham, R. and Schroeder, M., "Security and Authentication in Large Networks of Computers," Xerox Corp., Sept., 1977.

Mon77 Montgomery, W. A., "Measurements of Sharing in MULTICS," Proc. 6th Symp. on Operating Systems Principles, Nov. 1977, 85-90.

NBS77 National Bureau of Standards, Data Encryption Standard, FIPS PUB 46, Jan. 1977.

PoK78 Popek, G. J. and Kline, C. S., "Design Issues for Secure Computer Networks," Computer Science Dept., UCLA, 1978.

RSA78 Rivest, R. L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM, 21, 2, (Feb. 1973), 120-126.

Sal78 Saltzer, J., "On Digital Signatures," Operating Systems Review, 12, 2, (April 1978), 12-14.

Tan77 Tanenbaum, A., "A Distributed Interactive Computing System," IR-20, Vrije Universiteit, June 1977.