7. Knuth, D.E. and Bendix, P.B. Simple word problems in universal algebras. In *Computational Problems in Universal Algebras*, J. Leech, Ed., Pergamon Press, Oxford, 1969, pp. 263–297.
8. Lankford, D.S. Canonical algebraic simplification in computational logic. Memo ATP-25, Automatic Theorem Proving Project, U. of Texas, Austin, Texas, May 1975.
9. Lipton, R.J. and Snyder, L. On the halting of tree replacement systems. Proc. Conf. on Theoret. Comptr. Sci., Waterloo, Ontario, Aug. 1977, pp. 43–46.
10. Manna, Z. and Ness, S. On the termination of Markov algorithms. Proc. Third Hawaii Int. Conf. on Syst. Sci., Honolulu, Hawaii, Jan. 1970, pp. 789–792.
11. Manna, Z. and Waldinger, R.J. Is SOMETIME sometimes better than ALWAYS? Intermittent assertions in proving program correctness. *Comm. ACM 21*, 2 (Feb. 1978), 159–172.
12. Plaisted, D. Well-founded orderings for proving the termination of rewrite rules. Memo R-78-932, Dept. of Comptr. Sci., U. of Illinois, Urbana, Ill., July 1978.
13. Plaisted, D. A recursively defined ordering for proving termination of term rewriting systems. Memo R-78-943, Dept. of Comptr. Sci., U. of Illinois, Urbana, Ill., Oct. 1978.

Operating Systems

R. Stockton Gaines
Editor

Secure Personal Computing in an Insecure Network

Dorothy E. Denning
Purdue University

A method for implementing secure personal computing in a network with one or more central facilities is proposed. 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. A user can safely store confidential files in the central facility or transmit confidential data to other users on the network.

Key Words and Phrases: personal computing, security, privacy, networks, public-key encryption

CR Categories: 2.12, 6.20

1. Introduction

Within the next ten years many of us will have personal computers linked to a central facility. The central facility (CF) will offer many attractive features: long-term storage, text editors, language processors, spe-

This work was supported in part by National Science Foundation Grant MCS77-04835.
Author's address: D.E. Denning, Computer Science Department, Purdue University, West Lafayette, IN 47907.

cial-purpose software, video games, access to large data banks, and electronic mail. The CF could also pose a serious threat: Any or all of the secrets we entrust to it could be stolen surreptitiously; personal communications over the network could be intercepted; files stored in the CF could be copied; booby-trapped software borrowed from the CF 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 three basic premises: Users must be responsible for their own security, the CF may be insecure, and users will share confidential information in limited ways.

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 property. Numerous options are available for safeguarding jewelry or important papers—e.g. an unlocked drawer, a locked cabinet, a steel vault, or a safe deposit box. One evaluates the risks, cost, and convenience of each option to select the most suitable alternative. For the proposed system, each user can select safeguards for computer files and communication, applying the same criteria for risks, cost, and convenience as he does for his other possessions. 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 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 proved secure, there would be no guarantee that its specifications were complete or that an unsuspected compromise could not occur. Nevertheless, 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 may be subject to accidental (or intentional) loss or destruction. Whereas a user can tell when his data has been lost or destroyed, he cannot tell when it has illegitimately been copied. Yet the customers of the CF must feel that their personal data cannot be copied 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 facilitates sharing and whose philosophy encourages it, there is in fact little interuser sharing [10]. Consequently, users can share copies of confidential files rather than originals without unduly loading the resources of the CF. Users can share originals of nonconfidential files, however.

In the proposed system, the responsibility for safeguarding personal data belongs primarily to 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 [13] and Needham and Schroeder [12] in that both of these approaches rely on the security and correctness of the network, principally its key management facilities.

The encryption scheme proposed here applies to any network with one or more central nodes that provide message processing, software, file storage, or database facilities. The basic inspiration came not from "personal computing," but from Tanenbaum's distributed interactive system [18]. Tanenbaum proposed that each user have his own dedicated LSI microcomputer connected to a central minicomputer that provides file storage and software. The central machine sends a copy of a program to the user's computer for execution. Although Tanenbaum did not discuss data security, his system's principal feature—user isolation—is a sound basis for security. The simple encryption mechanism proposed here could make his system secure.

Section 2 outlines the mechanism. Section 3 describes how the mechanism solves three important security problems: personal security, secure communications and sharing, and secure signatures. Section 4 outlines the requirements of the interface with the CF. Finally, Section 5 treats cost and convenience.

2. The Security Mechanism

The security mechanism consists of an encryption device and hardware keys. The encryption device implements public-key encryption, a concept introduced by Diffie and Hellman [1].

2.1 Public-Key Encryption

Under public-key encryption, a plaintext message is enciphered using a *public key* P and deciphered using a *secret* (or *private*) key S. Let X^Y denote the enciphering or deciphering of message X with key Y.[1] For a given plaintext message M, the corresponding ciphertext C is related to M by the relations

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

To implement digital signatures, we shall further assume the enciphering algorithm is onto the same message space and that the keys P and S are commutative; therefore, for either plaintext or ciphertext message X,

[1] The notation used here follows that of Needham and Schroeder [12] rather than that of Diffie and Hellman [1], who used "E" and "D" to denote encryption and decryption transformations, respectively.

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

Simmons calls this asymmetric encryption because different keys are used at the end of the channel [17]. Specific algorithms are proposed in [1, 8, 9, 14] and surveyed in [3, 5, 7].

Public-key encryption has the important property that given a ciphertext C and public-key P, it is computationally infeasible to compute the corresponding plaintext message $M = C^S$ without knowing the secret key S. Consequently, public keys can be freely distributed without risking the security of enciphered data.

2.2 Hardware Keys

Each public-secret key pair is recorded on a pair of Read Only Memory (ROM) chips. The owner of the keys is told what sequence is engraved in the memory chip of the public "*P*-key" so that he can give it to his associates or list it with the CF. However, the sequence engraved in the memory chip of the secret "*S*-key" can remain secret, even to the key's owner. Flynn and Campasano discuss hardware-implemented keys in [2].

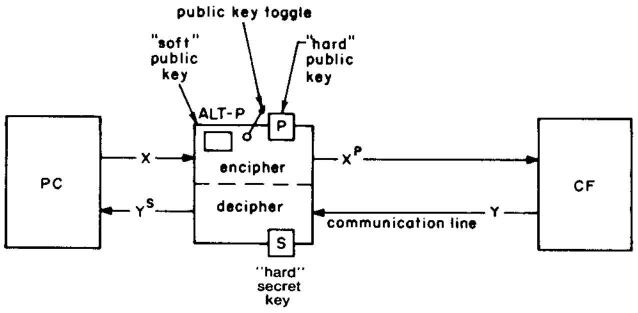
The key manufacturer may keep records of keys in order to handle lost or stolen keys, but these could be securely stored in a steel vault. The risk associated with this is no greater than that found in other areas; we do not worry about lock manufacturers breaking into our homes or stealing our cars. An alternative would be to give the user a mechanism for generating a random key pair on a writeable memory chip and sealing the chip from further alteration.

2.3 Encryption Device

The encryption device (Figure 1) has separate units for enciphering and deciphering. The *P* and *S* keys plug into separate sockets in the enciphering and deciphering units respectively. There is an additional facility for setting an alternate "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 and 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 personal computer (PC) and the CF. A message X originating from the user's PC passes through the enciphering unit and 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 passes through the deciphering unit and is deciphered with the *S*-key when it is received. The encryption device is an integral part of the PC-CF channel; all data transmitted between a PC and the CF passes through this device. This contrasts with the suggestion in [14] that the device not be wired in

Fig. 1. The encryption mechanism.



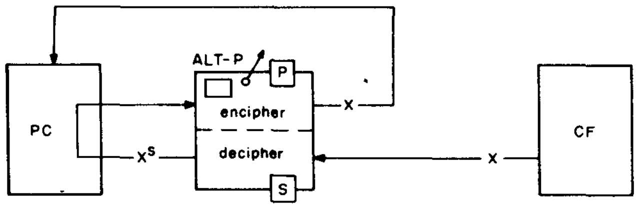
between the PC and the communication channel so that data can be successively enciphered with two keys in order to implement digital signatures. We will show (Section 3.3) that messages can be signed via an additional data path between the PC and the encryption device. However, no other communication lines between the PC and CF are permitted, thus assuring a user that his confidential data is properly enciphered and deciphered.

The purpose of the soft public key (*ALT-P*) is to enable a user to transmit messages to the CF and other users on the network. When the toggle is set to 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 assign the receiver's public key P' to *ALT-P* and manually set the toggle to use *ALT-P*. The assignment to *ALT-P* could be done under the control of a program running on the user's PC. Moreover, if *ALT-P* is assigned the user's public key P, the user need not move the toggle manually 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. There is no need for an "*ALT-S*" key since each user has a single secret key for deciphering information which has been enciphered with his public key.

2.4 Security of Keys

An important principle of this mechanism is that the CF keeps no records of secret keys. This is the primary reason this system uses public-key encryption rather than single-key encryption. Under single-key encryption, such as the Data Encryption Standard (DES) [11], the same secret key is used both for encryption and decryption. If keys are to be managed by the system, the CF must maintain and safeguard lists of secret keys. To transmit ciphertext to a user, the CF must know the user's secret key; in order that two users may communicate, the CF must provide a secret communication key. This violates the premise that the users should not have to rely on the security mechanisms of the CF. Under single-key encryption, users could exchange communications keys outside of the system. However, this may not be conven-

Fig. 2. Transmission of plaintext X from central facility to user's personal computer.



ient, and the CF still needs a user's secret key in order to encipher messages sent to him.

Users may reveal their public keys to the CF. Since all information arriving at a user's PC is automatically deciphered with the user's S-key, it must previously have been 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 plaintext programs requested by him before transmitting them to his PC. Moreover, the CF can provide directory service for public keys registered with the CF (see Section 4).

Although the user may optionally give a copy of his public key to the CF, it is not necessary for him to do so; he 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). This requires an extra data path through the enciphering unit; however, this data path is also required to implement digital signatures (see Section 3.3). Alternatively, it might seem desirable to provide a means whereby plaintext transmitted from the CF could bypass the encryption device. The problem with this is that the user may have no way of knowing if confidential data sent from the CF was properly enciphered before transmission.

3. Applications

3.1 Personal Security

To safeguard personal data, a 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 no copy of it in the CF; thus a perpetrator must steal or duplicate it in order to decipher the data.[2]

[2] 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. There is some advantage to a single device containing both the encryption algorithms and the memory chips implementing the keys.

With this mechanism a user can safely store (enciphered) confidential 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 program attempted to transmit the user's data back to the program's owner, the data would be automatically enciphered with the user's key, thereby rendering it useless.

A compiler, for example, could not steal proprietary software under development, nor could an income tax program steal confidential financial records. The mechanism can thus be used to implement confined (or memory-less) subsystems [6] although it may be possible to leak information on "covert channels" (e.g. by encoding it in the rate or quantity of transmitted ciphertext). The mechanism does not, 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. For a limited number of applications, it may be possible to use an encryption algorithm which allows the programs to operate directly on ciphertext [15].

The above approach can be used to implement personal security in single-key systems as well. A user would select a secret "personal" key which would be used to encipher secret documents. Unlike the secret keys used to communicate with the CF or other users on the network, this personal key would not be known outside the user's PC. However, this is slightly less attractive than the public-key implementation, as it requires the use of additional secret keys. Under public-key encryption, a user needs but a single secret key.

3.2 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 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 [1], A sends messages enciphered with P_B to B; similarly B sends messages enciphered with P_A to A (see Figure 3).

There is clearly no danger of an intruder intercepting and deciphering messages exchanged this way. To guard against an intruder recording and later replaying these messages, a sequence number or time stamp can be inserted into a message before it is enciphered.

The method also permits 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

Fig. 3. Secure Communication between Two Users A and B.

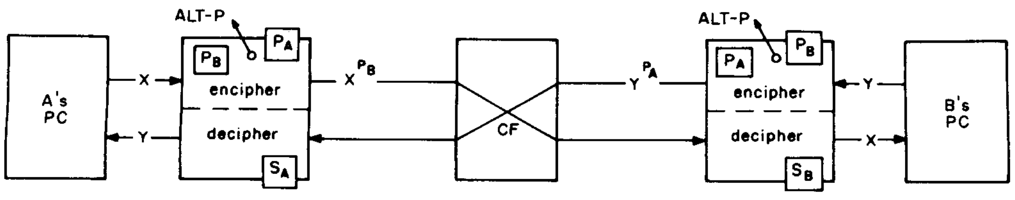


Fig. 4. User A Shares Confidential File F with User B.

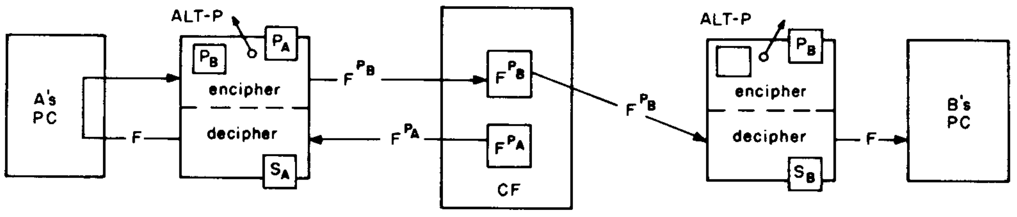
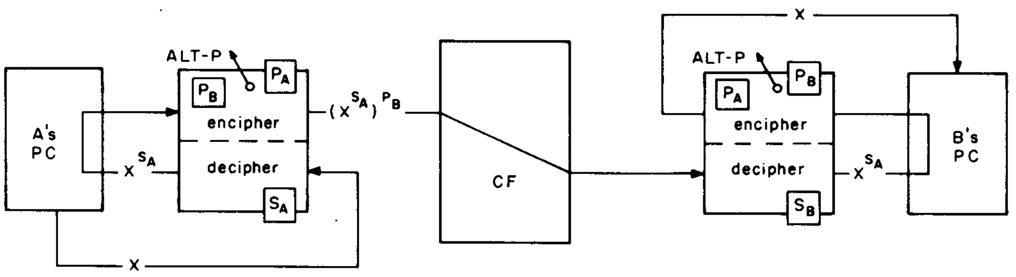


Fig. 5. Message X Securely Signed by A and Transmitted to B.



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 is that all confidential information traveling 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.

3.3 Secure Signatures

The proposed system can also be used to implement secure signatures as described in [1, 14]. To send a signed message X to B, A first operates on 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 operate on it with P_A to obtain the original message and know that it came from A, since only A could have used S_A (see Figure 5).

As outlined above, the method suffers from a problem pointed out by Saltzer [16]: B has no assurance that A did not lend or lose his secret key. Indeed, A could simply pretend to have lost his secret key! This problem does not arise with written signatures, since one cannot simply lend the ability to write his own signature. The problem of A's intentional loan or loss of his secret key can be solved by requiring that A sign (by hand!) a prior agreement making him responsible for all use of his key. Preventing signature misuse due to lost or stolen keys requires that the loss or theft be reported. On the other hand, digital signatures have two advantages: They cannot be forged (without acquiring the secret key), and it is possible when needed to lend one's key to an associate or secretary for signing documents and correspondence.

Another potential problem remains. To implement message-signing, an additional data path is required from the user's PC, through the deciphering unit, and back to the user's PC. This path could be a threat. Imagine, for example, a borrowed program processing confidential data X on a user's PC. Suppose that this program operates on X with the user's secret key S and then transmits the result X^S to the CF; this automatically causes encipherment with the user's public key P as the message enters the channel. Since $(X^S)^P = X$, the confidential data X is sent to the CF in plaintext! To prevent this "trojan horse" attack, a switch is needed whereby the user can control the use of the message-signing data path. This problem is even more difficult to solve in the implementation suggested by Rivest, Shamir, and Adleman [14], which allows the encryption unit to be invoked as a "hardware subroutine." Their scheme has the advantage that a message can easily be transformed under any sequence of public and/or private keys; it has the disadvantage that users cannot easily control the transformations performed by programs running on their PC's.

An additional data path through the enciphering unit is also required to validate a signature. However this does not appear to present a security threat, as performing additional public-key transformations (without corresponding secret-key ones) only tends to further scramble a message.

4. Interface with Central Facility

The CF is also equipped with a pair of keys and one or more encryption devices. In order that the CF may identify and decipher messages addressed to it, protocols are needed for communicating with the CF.

For example, messages addressed to the CF could be prefixed 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 the CF's public key P_{CF}. This would be followed by the file, enciphered under the user's public key P_A.

The CF may provide "directory assistance" for public keys of its customers. A user may reveal his *P*-key to the general public, or he may have an unlisted key that he personally gives to his associates.

There may be some risk associated with obtaining keys from the directory [12, 14]. If a user requests the key of an associate, the directory manager (or an imposter) could accidentally or intentionally supply an incorrect key; the user may unknowingly encipher confidential messages that are decipherable to a perpetrator rather than his associate! The problem of protecting against imposters can be solved by requiring a signature from the directory manager [12, 14]. However this does not protect against a faulty or untrustworthy directory manager. Both problems can be solved if users exchange signed "certificates" from the directory manager [4]. When a user registers his public key with the directory manager, he receives in return a signed certificate containing his public key. After verifying that the certificate came from the directory manager and contains his correct public key, he distributes the certificate directly to his associates. The receiver of a certificate can verify its authenticity before using the public key contained therein.

5. Practical Considerations

5.1 Cost

The ultimate utility of the proposal depends on the encryption speed and storage requirements of public-key encryption. The efficiency of the encryption algorithm is critical considering that all data sent to or from a user's PC must pass through the user's encryption device, and possibly the CF's as well. The encryption device cannot be bypassed even for nonconfidential data. For efficient transmissions, the encryption rate must be at least the network transmission rate. Hardware implementations of the (single-key) Data Encryption Standard (DES) satisfy this requirement. Although many researchers have been skeptical of the performance of public-key encryption, Rivest told the author that he estimates that there will soon be a two- to three-chip implementation of the prime factor method that runs at 5,000 baud or better. If this is so, public-key encryption schemes will soon be competitive with DES.

Low-cost hardware keys and encryption devices are also vital. Whereas "burning" a key into a ROM chip is attractive, it may not be economical. LSI is cheap, in part because identical components are mass produced; the cost of producing nonidentical components is much greater. It may be preferable to give users a means of generating their own keys, which they could record in an LSI/memory chip or possibly even on a magnetic stripe card.

5.2 Convenience

An economical encryption scheme must not only be efficient, it must be easy to use. The proposed scheme requires only that the user plug his encryption keys into the encryption device. He can, if he wishes, leave selection of the public key for enciphering outgoing data up to (borrowed or purchased) software running on his PC (via the *ALT-P* feature). However this does admit important security problems. For example, programs borrowed from the CF may compromise a user's security by enciphering confidential data transmitted to the CF (e.g. for storage) with the wrong public key. The presence of an additional data path through the deciphering unit (to implement digital signatures) gives the potential to leak confidential data in plaintext.

These problems are not unique to this proposal; they arise in any single-key or public-key system if security functions are trusted to software. However in this proposal the user has the option of not relying on the CF or borrowed software to safeguard his personal data.

It is also desirable to permit the option of connecting to the CF without an encryption device (or even software implemented encryption algorithms). This presents no problem in the proposed scheme as long as the CF can recognize messages addressed to it in either plaintext or ciphertext. A one-bit flag at the beginning of a message could indicate whether or not a message has been enciphered.

6. Conclusions

A method for implementing secure personal computing in a large network has been outlined. The method is based on the use of a public-key encryption device and hardware keys. All confidential data is enciphered as it is transmitted to the central facility or another node on the network. Because the central facility is not responsible for enciphering or deciphering a user's confidential data, it is not given access either to confidential plaintext or to the secret keys needed to decipher it. A user can safely store confidential files in the central facility or transmit confidential data to other users on the network.

Our objective has been to outline a promising approach to secure personal computing. There are a number of questions to be addressed. How should the encryption device and hardware keys be built? How should they interface with a user's personal computer? How should the mechanism be integrated into the network? To what extent can borrowed software be used safely? Is it possible to provide both confinement and digital signatures, or are these objectives conflicting?

Acknowledgments. I am grateful to J.P. Anderson, P. Denning, R. Stockton Gaines, M. Hellman, L. Hoffman, D. Parker, R. Rivest, A. Tanenbaum, and the referees for their comments and suggestions on an earlier version of this paper.

Received September 1978; revised March 1979

References
1. Diffie, W., and Hellman, M.E. New directions in cryptography. *IEEE Trans. Inform. Theory IT-22*, 6 (Nov. 1976), 644–654.
2. Flynn, R., and Campasano, A.S. Data dependent keys for a selective encryption terminal. Proc. AFIPS 1978 NCC, Vol. 47, AFIPS Press, Montvale, N.J., pp. 1127–1129.
3. Hellman, M.E. Security in communication networks. Proc. AFIPS 1978 NCC, Vol. 47, AFIPS Press, Montvale, N.J., pp. 1131–1134.
4. Konfelder, L.M. A method for certification. Tech. Rep., Lab. for Comptr. Sci., M.I.T., Cambridge, Mass., May 1978.
5. Konheim, A.G. Cryptographic methods for data protection. Res. Rep. RC 7026 (#30100), IBM Thomas J. Watson Res. Ctr., Yorktown Heights, N.Y., March 1978.
6. Lampson, B.W. A note on the confinement problem. *Comm. ACM 16*, 10 (Oct. 1973), 613–615.
7. Lempel, A. Cryptography in transition. *Comptg. Surveys* (to appear).
8. Merkle, R.C. Secure communication over an insecure channel. *Comm. ACM 21*, 4 (April 1978), 294–299.
9. Merkle, R.C., and Hellman, M.E. Hiding information and signatures in trap door knapsacks. *IEEE Trans. Inform. Theory IT-24*, 5 (Sept. 1978), 525–530.
10. Montgomery, W.A. Measurements of sharing in MULTICS. Proc. 6th Symp. on Operating Syst. Principles, Spec. issue, Operating Syst. Rev. (ACM) 11, 5, Nov. 1977, pp. 85–90.
11. National Bureau of Standards. *Data Encryption Standard.* FIPS PUB 46, Washington, D.C., Jan. 1977.
12. Needham, R., and Schroeder, M. Security and authentication in large networks of computers. *Comm. ACM 21*, 12 (Dec. 1978), 993–999.
13. Popek, G.J., and Kline, C.S. Design issues for secure computer networks. In *Operating Systems, an Advanced Course*, R. Bayer, R.M. Graham, and G. Seegmuller, Eds., Springer-Verlag, New York, 1978.
14. Rivest, R.L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM 21*, 2 (Feb. 1978), 120–126.
15. Rivest, R.L., Adleman, L., and Dertouzos, M.L. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, R. DeMillo, D. Dobkin, A. Jones, and R.L. Lipton, Eds., Academic Press, New York, 1978.
16. Saltzer, J. On digital signatures. *Operating Syst. Rev. 12*, 2 (April 1978), 12–14.
17. Simmons, G.J. Computational complexity and secure communications. *Comptg. Surveys* (to appear).
18. Tanenbaum, A. A distributed interactive computing system. IR-20, Vrije Universiteit, The Netherlands, June 1977.