The implementation of a cryptography-based secure office system

by CHRISTIAN MUELLER-SCHLOER
Siemens Corporation
Cherry Hill, New Jersey
and
NEAL R. WAGNER
Drexel University
Philadelphia, Pennsylvania

ABSTRACT

A cryptography-based secure office system is discussed, including design criteria and a specific implementation. The system is intended to be practical, simple, and inexpensive, but also highly secure. The implementation uses a hybrid scheme of conventional (DES) and public-key (RSA) cryptography. Randomly generated DES keys encrypt messages and files, and the DES keys themselves and a one-way hash of the messages are encrypted and signed by RSA keys. The system provides secure electronic mail (including electronic registered mail and an electronic notary public), secure two-way channels, and secure user files. Timestamps and a special signed file of public keys help decrease the need for an online central authority involved in all transactions.

INTRODUCTION

This article discusses the design and experimental implementation of a secure data processing system for an office or similar organization. The basic design should be usable by any number of users (from one to thousands or more), and the hardware may be distributed. We have attempted to design a practical, simple, inexpensive system whose underlying method of implementation is transparent to users. However, the principal design goal has been security. We also wanted to limit the need for an online (i.e., "always-present") central authority.

We visualize users each having access to a *personal work station* (PWS), i.e., a station with local computing power. While in use the hardware of each PWS will be assumed secure, but no security is assumed for the connections between PWSs or for an individual PWS when not in use. Users are not limited to a single PWS, but may sign on at any convenient one.

The system described here is intended as a supplement to the functions normally provided by operating systems and network managers. Our system provides the following basic functions:

1. Secure electronic mail, including electronic registered mail and an electronic notary public (one or more users who can authenticate a signature, provide a timestamp, and save a copy of the message)
2. Secure two-way communications channels (to give simultaneous interactive dialogue)
3. A secure user file system

Such a secure distributed system requires some use of cryptography.[1, 2] In order to achieve simplicity and low cost along with high security and ease of use, we used a hybrid system of conventional and public-key cryptography.

FUNCTIONAL DESCRIPTION

First we list the system's user-level security-related commands, suppressing other commands needed for an electronic mail system, such as "SearchMailbox," etc. The commands are independent of the particular form of implementation (whether conventional, public-key or hybrid like ours), and the user need not know anything about cryptography.

1. SignOn. (Input: Username, Password. Result: User is authenticated by the Password, and the PWS is initialized and thereby dedicated to the user.)
2. SignOff. (Input: None. Result: The PWS is deinitialized by overwriting sensitive areas and keys, etc.)
3. NewUser. (Input: Username, Password. Result: A new user is enrolled in the system with the password as the means of subsequent authentication.)
4. UpdatePublicKey. (Input: Username, Password. Result: A new public and secret key pair is substituted for the old.)
5. UpdatePassword. (Input: Username, New Password, Old Password. Result: A new password is substituted for the old.)
6. SendSecure. (Input: Destination-name, File [i.e., a "message"]. Optional input: Intermediate-destination-name, Request to register or notarize. Result: The file is timestamped, signed, and encrypted for the user whose name is Destination-name. In case of optional input, the file will first be routed to Intermediate-destination-name for registration or notarization.)
7. ReceiveSecure. (Input: Sender-name. Result: File from user implied by Sender-name is decrypted and authenticated.)
8. Register. (Input: Destination-name, Encrypted file. Result: The file is timestamped, signed, and forwarded.)
9. Notarize. (Input: Destination-name, Encrypted file. Result: The file is timestamped and signed, and a copy is retained before forwarding.)
10. AcknowledgeSecure. (Input: Sender-name, Encrypted file. Result: The file is timestamped, signed, and sent back to sender. This is for use with registered mail.)
11. OpenSecure. (Input: Destination-name. Result: A secure channel is created for immediate interactive use.)
12. CloseSecure. (Input: Destination-name. Result: The secure channel is closed.)
13. SaveSecure. (Input: Filename. Result: The file is saved in the user's mass storage in a secure way.)
14. RestoreSecure. (Input: Filename. Result: The saved encrypted file is made available as unencrypted cleartext.)

Initially, users must give the "NewUser" command with a password that they can remember. "NewUser" requires the physical presence of users at the central authority if authentication that a username corresponds to a particular physical individual is desired. "SignOn" must be given with a password matching the username. Messages or files of any sort can be sent to other users with the "SendSecure" command and can be received with the "ReceiveSecure" command. Various encryptions, decryptions, signatures, timestamps, and authentication steps are built into these commands at a lower level, as described below.

OVERVIEW OF OUR IMPLEMENTATION

In addition to the PWSs, our specific design uses a special component called the *crytoprocessor* (CP) to perform the various encryption/decryption functions, to store and generate keys, and to authenticate users. The CP is part of the PWS. We also use a special file called the *public-key file* (PKF). These public keys are signed with a *network secret key*. (The PKF also contains each user's secret key in encrypted form, as described later.) To replace the memorized username and password as entry to the CP, we can also use a data storage device called a *personal data card*. (Described in "Cryptographic Protection of Personal Data Cards," by C. Mueller-Schloer, submitted to the Seventh International Conference on Computer Communication.) The system design allows the simultaneous use of CPs implemented in either hardware or software. A software CP would be less expensive, perform more poorly and offer lower security than one in hardware. For a production system, a hardware CP could be made difficult to modify (for example, it could be embedded in epoxy), and this should increase security considerably.

Later sections describe the CP and the public-key file in more detail. Figure 1 gives a picture of these components.

We have chosen to use the data encryption standard (DES)[3] for the bulk of the encryption/decryption and to use the RSA public-key cryptosystem[4] for exchange of keys and signatures. DES is a natural choice because of its speed when implemented by inexpensive special hardware. If DES did not seem secure enough, one could switch to triple DES encryption[5] or to some other conventional system.

It is clear that DES alone would suffice for the complete implementation,[6] although with some considerable complications for key distribution and signatures. We have included public keys in a hybrid system because it places less burden on a central authority and allows more autonomy to users. We chose the RSA public-key scheme because it has been thoroughly studied and because of the symmetry between secrecy- and signature-encryption in that system. If RSA ever proves

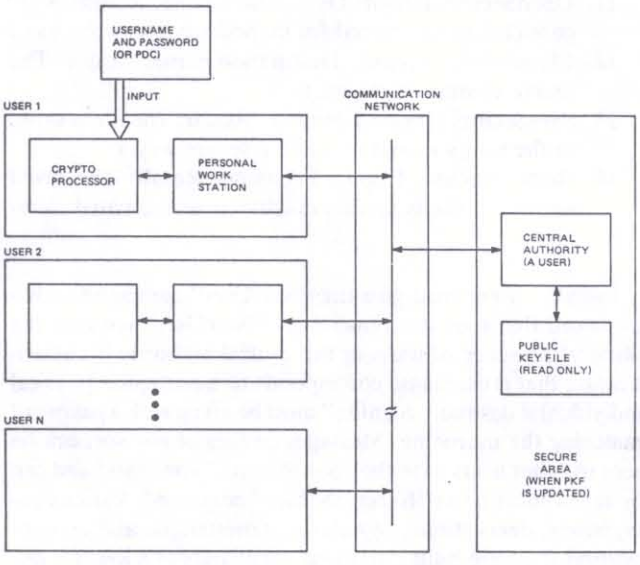

Figure 1—System overview

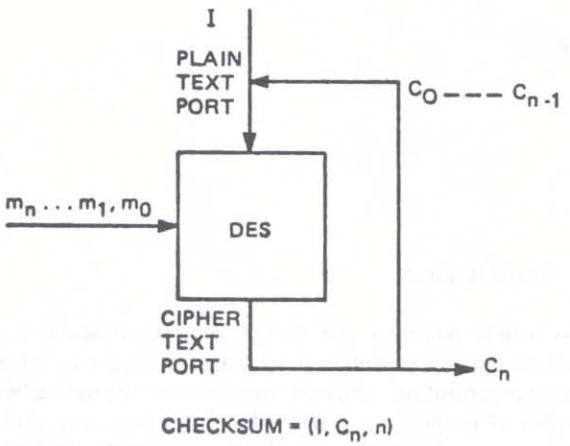

Figure 2—Checksum calculation (I = initialization vector, m_0, m_1, ..., m_n = 56-bit blocks of message, c_0, c_1, ..., c_{n-1} = 64-bit blocks of ciphertext)

insecure, we could switch to some other secure public-key system. The RSA scheme is slow even with special hardware,[7] and that is why we use a hybrid approach.

Because we wish to sign entire messages or files and yet want to apply the RSA signature only to a few blocks (to save encryption time), some sort of one-way hash function[8] is required. We use the DES to convert any text to a single 64-bit result, which we call a *checksum*. The method is illustrated in Figure 2. Since 64 bits are used, it is not feasible (assuming DES secure) for an opponent to construct an alternate text with the same checksum as the given text.[9, 10]

THE PUBLIC KEY FILE AND ENROLLMENT

The public RSA keys of all users are stored in a special *public-key file* (PKF).[11] (Also described in "The Cryptoprocessor: Hardware for a High-Level Cryptographic Instruction Set," by C. Mueller-Schloer, in preparation.) This file is accessible for reading except when created or updated by a distinguished user called the *central authority* (CA). The CA first generates one pair of RSA public and secret keys, called the *network public key* (PK.N), and the *Network Secret Key* (SK.N). When a user U executes "NewUser," the CA receives the username, the user public key (PK.U), the user secret key (SK.U) encrypted under a random local DES key (Kloc), and a special codeword (CW) for recovering the DES key. The CA forms the checksum (CS) of everything, adds a timestamp (TS), and signs these two with the network secret key. Thus a PKF entry looks like this:

Username, PK.U, $<$SK.U$>$Kloc, CW, {CS, TS}SK.N.

(Here $<$ ... $>$ is used for DES encryption and { ... } for RSA encryption.) When decrypted under PK.N, the precise format of the timestamp will serve to authenticate the entry.

As long as the SK.N remains secure, it will not be feasible for anyone to generate fake PKF entries, since the decrypted checksum must match the checksum generated from the first part of the entry. The timestamp is the time the entry was made, or the time the PKF was reconstructed. This timestamp

will prevent an old entry from being substituted for a current one. The time of each PKF reconstruction should be widely distributed; therefore each PKF entry must be timestamped no earlier than the overall timestamp.

An opponent can create fake network secret and public keys and then create a whole fake PKF. This can be defeated by wide dissemination of the network public key. If the personal data card is available, each user will have his/her own copy of the network public key, supplied when he/she enrolls.

Notice that we are not claiming for this system perfect security in the face of the "pervasive deceit" described by Simmons.[11] It is instead a practical approach.

THE CRYPTOPROCESSOR

As mentioned earlier, the cryptography functions of our system are all concentrated in a component called the *cryptoprocessor* (CP). Initially, we implemented this component in software, but we are now proceeding with a multibus compatible hardware implementation based on the Intel 8086 microprocessor and a Western Digital DES chip. The CP performs the basic functions of encryption/decryption and key generation for both DES and RSA schemes. It provides a well-defined, high-level, crypto-oriented instruction set that cannot be modified from the outside and therefore helps prevent interference by an intruder. Certain sensitive data like passwords and keys are stored internally in CP and can be manipulated only by using the CP's instruction set. (A command "OutputSecretKey," for example, does not exist!) Since RSA key generation on a microcomputer is relatively slow, it will occur as background activity in the CP.

The cryptoprocessor has a protected memory section called the *security status table* (SST). The SST is mostly to be filled in at "SignOn" time, using the input username and password or, if available, the hardware *personal data card* (PDC). (See the next section.) The PDC and the CP hardware solution will be described in detail in subsequent papers ("Cryptographic Protection . . ." and "The Cryptoprocessor . . .," both by C. Mueller-Schloer, cited previously).

PROCEDURAL DETAILS

We have tried to use simplified versions of standard protocols. In particular, we have chosen a public key file[11] and timestamps[13] instead of more elaborate protocols.[6]

In most cases the timestamps on messages that are sent, received, and acknowledged will be relatively close in time, so both parties will agree on the time of a message. Timestamps on registrations or notarizations applied by a third party will serve to settle any disagreements.

Let us go over the actions that occur at "SignOn" time. As Figure 3 shows, the PKF entry contains the username; the user public key (PK.U); the user secret key (SK.U), encrypted under a special Local DES key (Kloc); and a special codeword used to hide Kloc. The Local DES key is an *xor* combination of the codeword and the input user password, so an opponent with access to the PKF cannot recover Kloc and hence cannot calculate SK.U. It is important that we allow passwords of arbitrary length (and encourage long, easily-

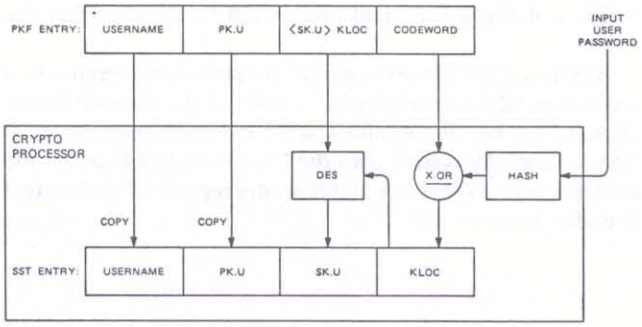

Figure 3—Use of public key file entry at "SignOn" time to initialize security status table of the cryptoprocessor

remembered ones), so Kloc is formed by first getting the checksum of the password and then *xor*ing with the codeword. The codeword itself is formed during the "Newuser" command by *xor*ing the password checksum and Kloc.[14] Thus

$$\text{codeword} = \text{checksum } \textit{xor } \text{Kloc},$$

where Kloc is randomly chosen. Since *xor* is self-inverse,

$$\text{Kloc} = \text{checksum } \textit{xor } \text{codeword}.$$

In case a hardware personal data card (PDC) is used, everything is handled in the same way except that the SST information originates from the PDC rather than the PKF. "SignOn" results in the work station's being temporarily dedicated to the user performing the operation.

Now consider the "SendSecure" command. User U starts with a file F and destination name D. U generates a random DES key KT and forms $<$F$>$KT, the file F encrypted under KT. Then U signs and encrypts for D the DES key KT. Finally U forms CS1, the checksum of everything up to this point, adds a timestamp TS1, and signs this. Thus

$$\text{U} -> \text{D}, <\text{F}>\text{KT}, \{ \{\text{KT}\}\text{SK.U} \}\text{PK.D}, \{\text{CS1}, \text{TS1}\}\text{SK.U}$$

is sent to D, where U $->$ D serves as cleartext routing information.

In case the file is routed through a notary public N, N forms CS2, the checksum of everything, adds a timestamp TS2, and signs the result. The notary public N can also recover CS1 and check that it is the checksum of $<$F$>$KT. Thus N adds

$$\{\text{CS2}, \text{TS2}\}\text{SK.N}.$$

Finally the destination D can acknowledge by forming CS3, the checksum of everything received, and signing and encrypting this for U (along with a new timestamp TS3). Thus U receives back what N added on and

$$\{ \{\text{CS3}, \text{TS3}\}\text{SK.D} \}\text{PK.U}.$$

Of course there is no need for D to send back what U originally sent out. The notary public will keep a copy of what he added on, but not of the original encrypted file. Thus all

traffic is of a relatively small size except for the encrypted file itself.

When a secure two-way channel is opened, the result is that a common DES key resides in the SSTs of the communicating stations. For key distributions public key encryption is used.[15] The file security system uses the local DES key Kloc for file encryption/decryption. A highly secure registered mail system is under development.

CONCLUSION

We have designed and implemented an experimental secure communications system for a network of personal work stations. The underlying cryptographic mechanisms guarantee a high level of security but are totally transparent to the user. The functionality matches that of today's paper mail security procedures. The confinement of security-related processing to one hardware device (the cryptoprocessor) with a well-defined high-level instruction set allows for higher speed and better protection of sensitive areas. The use of public-key cryptography limits the need for a heavily involved central authority. Other than publishing one network public key, no predistribution of keys is necessary. Users are not restricted to their own workstation but are assured full mobility in the network.

ACKNOWLEDGMENT

During part of this research the second author was supported by a grant from Siemens Corporation.

REFERENCES

1. Lagger, H., C. Mueller-Schloer, and H. Unterberger. "Security Aspects of Computer Controlled Communications Systems" (in German). *Elektronische Rechenanlagen,* 22 (1980), pp. 276–280.
2. Denning, D. E. "Secure Personal Computing in an Insecure Network." *Communications of the ACM,* 22 (1979), pp. 476–482.
3. "Data Encryption Standard." Federal Information Processing Standard (FIPS) Publication No. 46, National Bureau of Standards, January 1977.
4. Rivest, R. A., A. Shamir, and L. Adleman. "A Method of Obtaining Digital Signatures and Public-Key Cryptosystems." *Communications of the ACM,* 21 (1978), pp. 120–126.
5. Merkle, R. C., and M. E. Hellman. "On the Security of Multiple Encryption." *Communications of the ACM,* 24 (1981), pp. 465–467.
6. Needham, R. M., and M. D. Schroeder. "Using Encryption for Authentication in Large Networks of Computers." *Communications of the ACM,* 21 (1978), pp. 993–999.
7. Rivest, R. L. "A Description of a Single-Chip Implementation of the RSA Cipher." *Lambda,* 1 (1980), pp. 14–18.
8. Merkle, R. C., and M. E. Hellman. "Hiding Information and Signatures in Trapdoor Knapsacks." *IEEE Transactions on Information Theory,* IT-24 (1978), pp. 525–530.
9. Mueller-Schloer, C. "The Usage of DES-generated Checksums for Electronic Signatures." Internal Report CRT-81-TM-043, Siemens Corporation, Cherry Hill, New Jersey, September 1981.
10. Davies, D. W., and W. L. Price. "The Application of Digital Signatures Based on Public Key Cryptosystems." In J. Salz (ed.), *Proceedings of the Fifth International Conference on Computer Communication,* The International Council for Computer Communication, 1980, pp. 525–530.
11. Simmons, G. J. "Secure Communications in the Presence of Pervasive Deceit." In *Proceedings of the 1980 Symposium on Security and Privacy.* IEEE Computer Society, 1980, pp. 84–93.
12. Merkle, R. C. "Protocols for Public Key Cryptosystems." In *Proceedings of the 1980 Symposium on Security and Privacy.* IEEE Computer Society, 1980, pp. 122–134.
13. Denning, D. E., and G. M. Sacco. "Timestamps in Key Distribution Protocols," *Communications of the ACM,* 24 (1981), 8, pp. 533-536.
14. Wagner, N. R., "Practical Approaches to Secure Computer Systems," Technical Report UH-CS-81-3, Computer Science Department, University of Houston, Texas, April 1981.
15. Popek, G. J., and C. S. Kline, "Encryption and Secure Computer Networks," *Computing Surveys,* 11 (1979), 4, pp. 331-356.