Integrating the Data Encryption Standard into Computer Networks

MILES E. SMID

Abstract—**The NBS Data Encryption Standard may be integrated into computer networks to protect personal (nonshared) files, to communicate securely both on- and off-line with local and remote users, to protect against key substitution, to authenticate system users, to authenticate data, and to provide digital signatures using a nonpublic key encryption algorithm. Key notarization facilities give users the capability of exercising a set of commands for key management as well as for data encryption functions. The facilities perform notarization which, upon encryption, seals a key or password with the identities of the transmitter and intended receiver. Thus, in order to decrypt a message, the receiver must authenticate himself and supply the correct identity of the transmitter. This feature eliminates the threat of key substitution which must be protected against to attain a high level of security.**

INTRODUCTION

IN 1977 the National Bureau of Standards (NBS) published a completely defined encryption algorithm known as the Data Encryption Standard (DES) which became a Federal standard for the protection of unclassified data [2]. The International Business Machines Corporation had made the DES specifications available to NBS, and had provided nondiscriminatory and royalty free licensing for building DES devices. Since publication of the standard, several manufacturers have produced DES implementations, and there has been an increased awareness that, in certain applications, encryption offers the only effective means of protecting information. The first applications of the encryption of unclassified data appeared in the area of electronic funds transfer, but the passage of the Privacy Act of 1974 (5 USC 522a) and Transmittal Memorandum No. 1 by the Office of Management and Budget to its Circular A-71 placed added responsibilities on Federal data systems for the protection of nonfinancial data as well.

Even before the DES was adopted, it was clear that there was more to cryptographic security than a secure encryption algorithm. Efforts were initiated by NBS to have additional standards, based on the DES, developed. One area which needed to be addressed was the integration of cryptography into computer networks. In order to gain experience, NBS decided to design and implement cryptography into its experimental computer facility. This decision led to a consideration of likely cryptographic requirements and the development of new techniques for their solution. The resulting system, called the key notarization system (KNS), is discussed in this paper.

Manuscript received February 13, 1980; revised November 18, 1980. This paper was presented at the National Telecommunications Conference, Washington, DC, November 1979.

The author is with the Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, DC 20234.

REQUIREMENTS

Whenever a cryptographic system is to be implemented, consideration should be given to the requirements. What protection is desired? What capabilities must the system have? How are the keys to be managed? (DES keys are 64-bit vectors composed of eight parity bits and 56 bits that are individually selected in order to provide the unknown quantity necessary for security of the DES function.)

Secure Data Storage

The first requirement is to provide the computer user with the capability of encrypting files for off-line storage and recovery at a later time. Once encrypted, personal files can only be decrypted by the original owner. They are encrypted for secure storage rather than for secure communication. In this case, encryption is used to protect against accidental disclosure, such as spillage, and intentional disclosure, such as scavenging. In order to lessen the number of keys which must be managed by the system, it is often desirable to store the data encrypting key with the cipher. Of course, the key must be encrypted under another long term key which is kept for the user in a highly protected location. Since only the file owner should be able to recover the plain text once encrypted, we are naturally led to the consideration of the second requirement.

User Authentication

One must decide how the true owner of a particular file is to be identified. The owner could directly supply a key for each file he wishes to decrypt. Thus, he would be implicitly authenticated by his ability to decipher the data. If the file encrypting keys are unencrypted they cannot be stored in unprotected memory and so the user is usually given the burden of remembering and protecting his keys. On the other hand, it might be preferable to allow the user no knowledge of unencrypted keys. An authentication system is used to correctly identify users, and data access is based on the user's identity and system encrypted keys. At present the most common user authentication systems employ passwords. In the future more secure and more convenient techniques such as fingerprints and signature analyzers may evolve. The system described in this paper uses password authentication but could be updated to more advanced methods in the future.

Secure Data Communications

If the user has the capability of encrypting files for secure storage, he will certainly want to encipher a file and send it to another user for decryption. This other party may be on the

same computer or he may be linked to the transmitter by means of a network. Thus, we are now considering a capability for secure data communications. Secure communications involve preventing the disclosure of plain text, detecting fraudulent message modification, detecting fraudulent message insertion or deletion, and detecting fraudulent replay of a previously valid message. Any cryptographic system should be consistent with these goals and yet operate at speeds sufficient for normal network communications.

Secure Communications Via Encrypted Mail (Off-Line)

With mail encryption, data are encrypted and then sent via mail, or some means which cannot provide an immediate response. The data are stored in encrypted form until decryption at some later time. In this situation one cannot have an interactive system for exchanging keys because no real-time response is possible. Therefore, protocols must be devised so that the receipt of keys need not be immediately acknowledged.

A Digital Signature Capability

If a computer user may receive an encrypted file or encrypted key from another user across the network, then it would be desirable to have a signature system whereby the receiver could prove that the file or key did in fact come from a particular transmitter. Digital signatures were developed in conjunction with public key algorithms (see Diffie and Hellman [4] and Rivest, Shamir, and Adleman [12]). In such systems the decryption key is not equal to, and cannot be computed from, the encryption key. Encryption keys may be made public while decryption keys are kept secret. A digital signature is transformed using the secret decryption key and sent to the receiver. The receiver may encrypt, using the public key, and verify the signature, but the signature cannot be forged since only the transmitter knows the secret decryption key. (The cryptoalgorithm must have the property that decryption of the signature followed by encryption equals the original signature.) Denning [3] has studied the implementation of public key algorithms in computer networks. Nonpublic key algorithms use the same key for both encryption and decryption. In the NBS system, a new method is used to implement digital signatures with the DES nonpublic key algorithm. In this system every message may be regarded as a signature.

Key Management

Underlying all of the requirements mentioned above is key management. Key management involves the secure generation, distribution, storage, and destruction of cryptographic keys. Early key management techniques for computer networks were developed by Branstad [1], Ehrsam *et al.* [5], and Everton [6]. A key management system must be both secure and flexible enough to be consistent with the requirements. If the key management is weak, then the most sophisticated cryptoalgorithm will be of little value. In fact, a very strong cryptoalgorithm used in a weak key management system can give a false sense of security. To satisfy these requirements, NBS designed the key notarization system (KNS), and is currently in the process of implementing it in the NBS experimental computer facility.

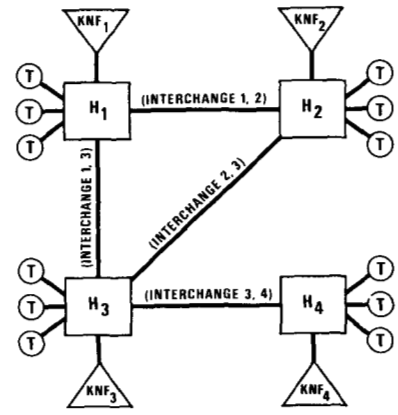


Fig. 1. A four host network.

SYSTEM DESIGN

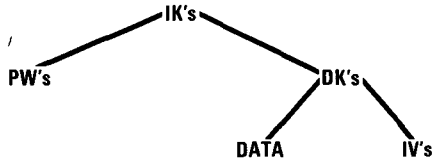
The Network

The key notarization system (KNS) is designed for computer networks which consist of host computers, user terminals, and key notarization facilities (KNF's). Fig. 1 shows a four host network. The host controls the normal operation and communication of the terminals. Terminals have the capability of communicating with the host, with other local terminals through the host, and with terminals of other hosts via communication channels. The communication channels between hosts are called *interchanges.* Interchanges may be electronic communication lines, microwave links, courier routes, etc., or combinations of more than one medium. Each terminal will be able to use the host KNF by means of KNF commands. These commands will be implemented in the KNF, and every KNF will have the capacity to generate keys for distribution to other hosts or facility users.

The lines between the KNF and its host and the lines between each terminal and its host must be protected. They could be physically secured or they could be secured by the addition of cryptographic devices on each end of the links. A cryptographic capability may also be implemented so that the entire communications path from the terminal to the KNF, including the path through the host, is cryptographically protected. This technique can provide for a minimum reliance on the host for security, especially when the host is merely acting as a communications switch which contains no sensitive data. When a user is editing a file in the host, it is in plain text form, and the host will have to protect the data from other users. Once the user has finished editing, he may command the KNF to encrypt the data and store the resulting cipher in unprotected memory or send it to a remote user over an interchange.

The Key Hierarchy

Two distinct types of keys, shown in Fig. 2, form the key hierarchy, *interchange keys* (IK's) and *data keys* (DK's). Interchange keys encrypt passwords (PW's) and data keys, while data keys encrypt both data and *initialization vectors* (IV's). IV's are randomly generated values which initialize the DES



IK = INTERCHANGE KEY
PW = PASSWORD
DK = DATA KEY
IV = INITIALIZATION VECTOR

Fig. 2. The KNF key hierarchy.

algorithm in certain applications called modes of operation [7].

Interchange keys are used for the exchange of keys between users. One interchange key, called the *facility interchange key*, is used for the encryption of facility user passwords. Other interchange keys are available for the exchange of data keys between facilities or subgroups of a facility. IK's are generated outside the network and are entered, unencrypted, directly into the KNF. This permits two facilities to enter the same IK. With the addition of another KNF command, one could encrypt the IK's in the facility interchange key to reduce the number of interchange keys needing protection when stored outside of the KNF. One IK can be used to connect all the users of two hosts since a user may not decrypt a data key shared by two other users. This is because the identifiers of the two parties are involved in the encryption of the shared key.

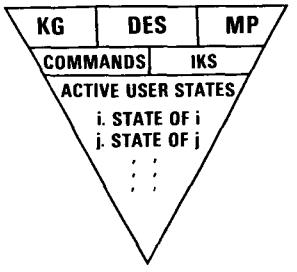
Data keys are used to encrypt data belonging to one particular user. DK's are generated by the key generator and are immediately encrypted. When encrypted, DK's may be sent, kept in unprotected memory, etc. DK's are also used to encrypt initialization vectors. All IV's are encrypted, before they leave the KNF, under the data key which enciphers the corresponding data.

The Host

The host computers have two types of memory: that which is not normally accessible to users, called system memory, and that which is accessible to users, called user memory. User i's memory is core, disk, etc., where user i is permitted to store and recall data. The encrypted keys of user i are stored in user i's memory, and encrypted passwords to which no user needs access will be stored in system memory. Most computers have a means of protecting system memory from users, and some computers protect one user from another to a certain degree. Nevertheless, we will assume that any user can gain unauthorized read and write access to both encrypted keys and encrypted passwords. In this situation, encryption alone is not sufficient. A method is required to protect against key substitution and to insure that each user correctly identifies the user with whom he is communicating.

The Key Notarization Facility (KNF)

The KNF contains a DES encryption device. It will have a control microprocessor and memory to implement commands and data transfers. The KNF must also store the unencrypted interchange keys and the states of active users. (See Fig. 3.) An *active state* consists of a user identifier along with two DES initilization vectors (IV's) and two unencrypted data keys (one



KG=KEY GENERATOR
DES=DATA ENCRYPTION STANDARD
MP=MICROPROCESSOR
IKS=INTERCHANGE KEY STORAGE
STATE=TRANSMIT KEY, RECEIVE KEY

Fig. 3. The key notarization facility.

IV-key pair for transmitting and the other pair for receiving). Separate transmit and receive key storage permits alternating encryption and decryption without key loading on each transition. A user is *active* as soon as his identifier is loaded into *active user memory* in the KNF. He may then proceed to load the rest of his state. Multiple users may be active at the same time.

The KNF has the capability of using the DES to encrypt, decrypt, and authenticate data via the cipher feedback (CFB) or cipher block chaining (CBC) modes of operation described in the NBS proposed DES modes of operation standard [7].

The KNF contains a key generator which is capable of generating a 56-bit random number. Once the 56-bit number is generated the proper parity is determined and the entire 64-bit key is encrypted before it is returned to the host. The key generator is also used to generate the IV's used by the DES algorithm. It is important that the key generator is contained within the KNF so that its operation cannot be subverted. Since the KNF contains clear keys, the encryption algorithm, the commands program, and the key generator, it must be physically protected.

IDENTIFIERS AND KEY NOTARIZATION

A special feature of the KNS is the support of key notarization. This feature increases security, permits a simple system design, and provides a means of implementing signatures with a nonpublic key system. *Identifiers* are nonsecret binary vectors of up to 28 bits which uniquely identify each user in the network. When a user first attempts to call the KNF he must submit his identifier along with the correct password to establish an active state in the KNF. Both the host and the KNF employ identifiers to "recognize" the users.

Key notarization is similar to the actions of a notary public who first requires his customer to identify himself via a driver's license, etc., before he seals (notarizes) the customer's signature on a document with his notary stamp. In addition to the notary's function of authenticating the creator of a message, the KNS authenticates the message itself and the person requesting decryption. Key notarization is similar to having a notary public on each end of a secure communication channel.

Let i and j be identifiers and IK be a DES interchange key. Then $(i \| j)$ represents the concatenation of i and j. IK, a 64-bit key, consists of eight bytes, each with seven information bits and a parity bit. IK XOR $(i \| j)$ is a special function defined as follows. The leftmost seven information bits of IK are exclusive or'ed with the leftmost seven bits of i. The eighth

bit, a parity bit, is then appended so that the modulo 2 sum of all eight bits is odd. Then the next seven information bits of IK are exlcusive or'ed with the next seven bits of i and the correct parity bit is appended. This continues until the last seven information bits of IK have been exclusive or'ed with the last seven bits of j and the final parity bit has been set. Therefore, IK XOR $(i \| j)$ is a valid DES key with 56 information bits and eight parity bits.

All passwords and data keys are notarized by being encrypted under IK XOR $(i \| j)$ for some IK and some i, j pair. The identifier of the user requesting the key, who is also the transmitter, is always the left identifier and the identifier of the intended receiver is the right identifier in the identifier pair. In the case of passwords and private files, $i = j$. Security is increased because one user cannot substitute his password or keys for those of another user and be able to authenticate or decrypt as that user. This will be explained in detail in KEY AND PASSWORD PROTECTION. The security is also increased because both parties in a conversation must know the other's correct identity to communicate. Since the identities of the sender and receiver are combined in a notarized key, a signature system may be devised. This will be discussed further in DIGITAL SIGNATURES.

When key notarization is used, keys and passwords are sealed, upon encryption by the KNF, with the identifiers of the transmitter (key generator) and the receiver. To generate a notarized key the transmitter must identify himself to the KNF and provide proof of his identity by supplying his correct password. He must also identify the intended receiver of the key. Once encrypted, the correct key cannot be decrypted unless the correct identifier pair is again provided. To decrypt the key the receiver identifies himself and provides password proof of his identity. The receiver must also supply the identity of the transmitter which may have been sent unencrypted. If the identification information is not the same as that provided by the transmitter to his KNF, then the decrypted key will not equal the original key and no information can be correctly decrypted. Thus, the receiver must know the correct transmitter and be the intended receiver.

USER AUTHENTICATION

Each user will have a password which is used to authenticate the user and permit him to invoke KNF commands. The plain password is passed through an encryption function, involving the user's identifier, and the result is compared with a stored value before the user is activated. Therefore, a user cannot exercise any other command until his identity has been authenticated. The password of each user is stored in system memory encrypted under the facility interchange key (see KEY HIERARCHY) combined with the user's identifier.

If the host can maintain the correct identity of a user once he has been authenticated, the user need not resubmit his password for each command he executes while he is active. His authenticated identifier which has been loaded into active user memory will automatically be used as his identifier. If the host cannot be trusted to maintain user identities, then cryptographic keys may be used instead of passwords. These keys are unique to each user and provide for secure communications between the user at his terminal and his local KNF.

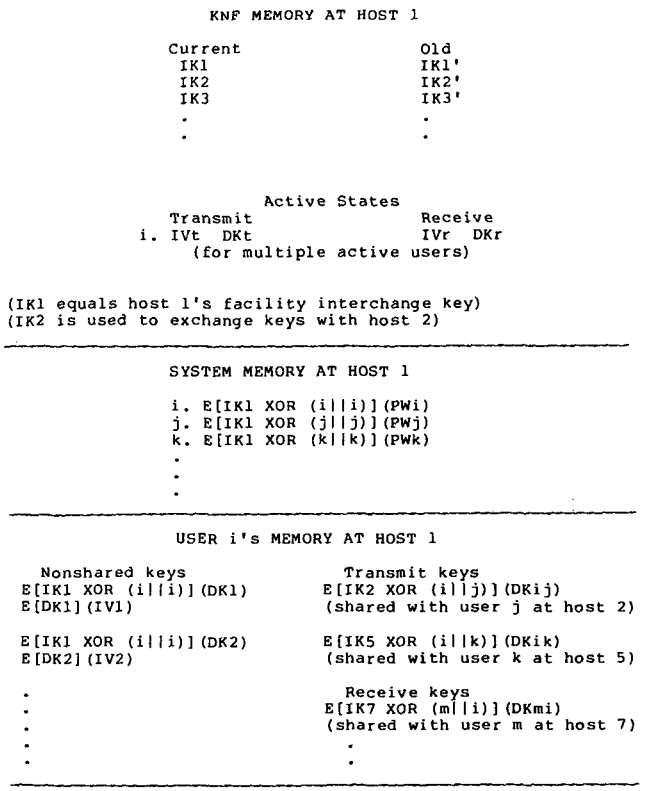


Fig. 4. Password and key storage.

In this case, a terminal encryption unit is required. Rather than authenticating the password, the KNF decrypts the encrypted key of the user and then employs it for all subsequent communications between the user and the KNF. With this implementation, the host could not cause a security compromise by switching two user's identities with respect to the commands passed to the KNF.

KEY AND PASSWORD PROTECTION

Let $E[X](Y)$ indicate the encryption of Y under key X as defined by the DES. Fig. 4 shows how keys and passwords appear in KNF memory at host 1, in host 1 system memory, and in the memory of user i at host 1.

KNF Keys

KNF memory contains interchange keys and active states. When the interchange keys are changed, the old interchange keys are securely stored outside of the KNF along with their effective dates. The current IK's become the old interchange keys and the new IK's are stored as the current IK's. After such a change, the passwords are reencrypted under the current (new) facility interchange key, and the users are told to reencrypt their data keys.

Fig. 4 shows only one transmit (IVt, DKt) pair and one receive (IVr, DKr) pair for user i. Implementations may be devised which permit a user to have several transmit and receive keys active simultaneously.

Passwords

Host memory contains the encrypted passwords for every user. E[IK1 XOR $(i \| i)$](PWi) denotes the encryption of PWi under IK1 XOR'ed with user i's identifier pair, $(i \| i)$. IK1 is used because the encrypted passwords are from the system

memory of host 1 and IK1 is the facility interchange key for host 1.

Although passwords are enciphered so as to protect them from disclosure, they should also be protected against substitution. The password is encrypted under IK1 *X*OR'ed with the appropriate identifier pair to provide the necessary protection. If identifiers were not used, system memory might appear as follows:

i E[IK1] (PWi)

j E[IK1] (PWj)

•
•
•

If user j could gain access to system memory, he might alter it as follows:

i E[IK1] (PWj)

j E[IK1] (PWj)

•
•
•

User j could then authenticate as user i by submitting his own password while claiming to be user i. If identifiers were used in Fig. 4, then E[IK1 *X*OR ($i \| i$)] (PWj) would be calculated upon authentication and it would not compare with E[IK1 *X*OR ($j \| j$)] (PWj) which was substituted as user i's encrypted password.

Data Keys

User i's memory contains personal and shared data keys. Personal data keys protect private data and are encrypted under the facility interchange key *X*OR'ed with the user's identifier pair. User i's memory also contains shared data keys. Keys used for transmitting data are encrypted under an interchange key *X*OR'ed with the concatenation of user i's identifier and another user's identifier. For receiving data, user i's identifier appears second in the identifier pair. ($i \| j$) uniquely identifies the communication parties. If ($i \| j$) were not used, another user could substitute his own data key encrypted under the interchange key and then be able to decrypt any subsequent cipher. Similarly, when user j receives E[IKp *X*OR ($i \| j$)] (DKij), he must know that he is communicating with i, over interchange p, to correctly decipher DKij. Thus, the transmitter is prevented from posing as someone else. Since several users may all use the same IKp to communicate, this protection is critical. IV's which need to be reused may be stored in encrypted form with the notarized data key. The IV for a shared key may be regenerated by the transmitter each time the key is loaded into the active memory.

One could argue that substitution protection is not needed for system memory because if the system cannot protect system memory, it probably cannot prevent users from changing identity, from invoking system commands, or other security threats. This may be true but encryption should not add additional possibilities for attacks and it is desirable to minimize the protections required of the operating system. In user memory the substitution threat is very real because many systems cannot protect one user's memory from another user, and even if they could, the encrypted keys will not be protected because encrypted data keys may be stored with cipher on unprotected tapes and disks, and they may even be sent out over unprotected communication channels. Rekeying techniques, whereby keys are transmitted in encrypted form over an unprotected channel, must consider the substitution threat whenever an active tap (one which permits data alteration) is feasible.

The protection provided by notarization is protection against a security loss of sensitive data by means of key substitution. It does not prevent the disruption of communications or garbling of data which may result from the modification of cipher text.

REMOTE KEY DISTRIBUTION

In Fig. 1, host 1 and host 3 share an interchange (direct means of communication). If their KNF's also share an interchange key, then they may exchange data keys by enciphering them under the interchange key. This exchange is called remote key distribution. After a common data key is held by both KNF's, data may be passed between the two KNF's encrypted under the data key. However, only host 3 shares an interchange with host 4. If KNF 1 shares a common interchange with KNF 4, then host 1 may communicate with host 4 through host 3 without intermediate decryption and reencryption. Host 3 would merely act as a switch. If KNF 1 does not share a common key with KNF 4 but does share a key with KNF 3, and if KNF 3 shares a key with KNF 4, then host 1 may communicate with host 4 via host 3. The data key, which is used to encipher the file at host 1, is encrypted and sent to KNF 3. KNF 3 decrypts the data key and reencrypts it for transmission to KNF 4. The file, which is enciphered under the common data key, is passed through host 3 but is only decrypted by the intended receiver at host 4. Using this method of key distribution, n hosts may be connected with as few as $n - 1$ keys. (See Fig. 5.) It is this feature which permits the connection of a large number of hosts with a roughly equivalent number of keys. When a key is reencrypted at an intermediate KNF, the intermediate KNF must be able to recognize a key reencryption message, and the parties must rely on the physical security of the intermediate KNF. If no intermediate KNF's are to be trusted, then a secret interchange key must be manually exchanged between the transmit and receive KNF's.

DISTRIBUTED VERSUS CENTRALIZED KEY GENERATION

Branstad [1] describes how network security centers (NSC's) may be used for key generation and distribution. Upon request, the NSC generates a key for use by each of the parties in a conversation. One copy is encrypted under a key shared between the NSC and the first party and another copy is encrypted under a key shared between the NSC and the second

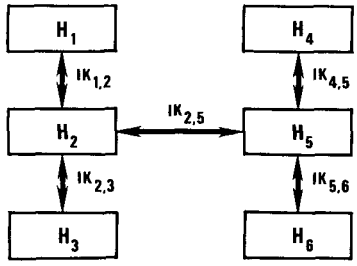


Fig. 5. Six hosts connected by five interchange keys.

party. The encrypted forms of the key are then sent to the appropriate receivers.

The KNS uses distributed rather than centralized key generation as employed by an NSC. The KNS gives each host the capability of key generation in its own KNF. Thus, two hosts do not even have to be electronically connected in order to communicate securely. The KNS requires fewer protocols because parties do not have to send a remote key generation request and they do not have to respond to the receipt of a key. The transmitter always generates the shared data key. Fewer protocols mean fewer ways an enemy can attempt to trick or confuse the communicating parties by altering or playing back the protocol messages. (See Needham and Schroeder [9].)

If a KNF is compromised, only communications involving the comprised facility are exposed. If an NSC is compromised, and there is only one NSC for the network, then the whole network is exposed. With a local key generator, one can encrypt personal (nonshared) files without having to depend on any remote site. The KNS approach does require that the key generation capability and the KNF physical security be replicated at each host.

EXTENDING DATA KEY ACCESS

The KNF allows the owner of an encrypted file to extend access to the file without having to reencrypt the data. For example, user i may have encrypted a personal file so that the encrypted data key has the form

$$ED = E\,[\mathrm{IK1}\ XOR\ (i \,||\, i)](\mathrm{DK}).$$

If he then wishes to share this file with user j on interchange 5, he may request that the KNF reencrypt the data key to

$$ED' = E\,[\mathrm{IK}\,5\ XOR\ (i \,||\, j)](\mathrm{DK}).$$

If user j has access to IK5, user i may transfer read access to j by providing j with the original IV, the original cipher, and ED'. The KNF will not permit i to change his identity as the original encryptor of the file. Therefore, user j will be able to read the file but he will not be able to modify the file and encrypt it under DK. In order to have several users each with the capability to modify a single encrypted file a special identifier would have to be implemented for the entire group.

DIGITAL SIGNATURES

Rationale

Digital signatures are possible with public key algorithms because one cannot decrypt another person's data even though anyone with the public key can encrypt data intended for that person. This is because the decrypt key is not shared. Since the KNF combines identifiers with interchange keys for protection against substitution and employs separate encryption and decryption key storage, one cannot encrypt data in a key that was generated by another user.

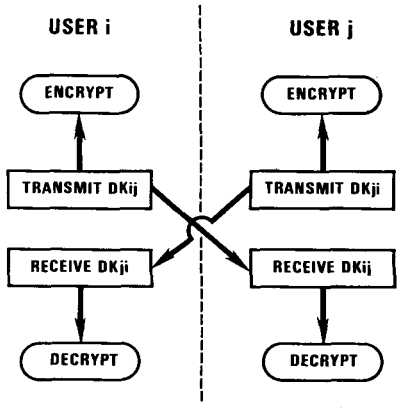


Fig. 6. Separate transmit and receive key storage.

Suppose user i generates a key and sends it to user j. The encrypted data key would be of the form

$$ED = E\,[\mathrm{IK}p\ XOR\ (i \,||\, j)](\mathrm{DK}ij)$$

where IKp is the interchange key for interchange p connecting i's host with j's host, and DKij indicates a data key generated by i for transmission to j. Whenever i generates a key his identifier is always leftmost in the identifier pair used in the encryption of the key. The only way user j can load DKij is by loading it as a receive key. If j tries to load DKij as a transmission key for the encryption of data going to i, the KNF will use $(j \,||\, i)$ instead of $(i \,||\, j)$ when decrypting ED. If j tries to load the key as a personal key, then $(j \,||\, j)$ will be used. When DKij is loaded as a receive key, only the decryption commands have access to it. (See Fig. 6.)

Example

Suppose user i generates ED as before. He may then generate an encrypted IV of the form

$$EI = E\,[\mathrm{DK}ij](\mathrm{IV}).$$

Next, he may encrypt a signature, S, under DKij and send ED, EI, and the encrypted signature to j. User j may load IV and DKij in the active receive state and decrypt the encrypted signature to recover S. There is no way that j can alter S to a particular S' and encrypt it under DKij because there is no way for j to get DKij into the transmit data key active storage.

If user j generates his own encrypted data key, it will be of the form: $E[\mathrm{IK}p\ XOR\ (j \,||\, i)](\mathrm{DK}ji)$. He may encrypt a signature S' under DKji but he cannot claim that it came from i because he could be challenged to decrypt the encrypted signature. To do so j would have to load DKji by submitting $E[\mathrm{IK}p\ XOR\ (j \,||\, i)](\mathrm{DK}ji)$ to the KNF for loading in the receive key register. The KNF would not load the correct DKji because it would use $(i \,||\, j)$ instead of $(j \,||\, i)$ as the identifier pair. Thus, the signature would be garbled. (The correct signa-

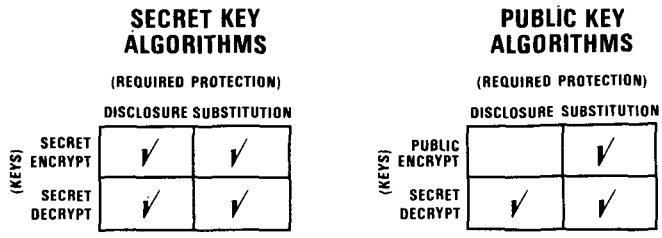


Fig. 7. Secret key versus public key protection.

ture must be something meaningful so that it may be distinguished from a garbled signature.) Of course, user j may send a signature S' to user i encrypted under the data key, DKji, in a similar manner as described above.

The signature procedure is the same as the protocols used when i sends an encrypted file or message to j except that the signature takes the place of the message. Any message may be regarded as a signature. No additional keys or commands are required. All user j needs to do is keep E[IKp XOR (i || j)] (DKij), E[DKij] (IV), and the encrypted signature in order to be able to prove that the signature was sent to him from i. With these parameters j can decrypt the signature at any later time. Signatures should be large enough to provide adequate security. At least 64 bits are recommended.

True and Arbitrated Signatures

Matyas [8] distinguishes between two types of signatures: true signatures and arbitrated signatures. Both types require a trusted arbitrator to resolve disputes but true signatures do not require the arbitrator to take part in the normal calculation and verification of a signature. Public key algorithms can be used to provide true signatures. Lamport [4] and Rabin [11] have shown that nonpublic key algorithms may also implement true signatures, but their techniques require the exchange of predetermined validation patterns in advance of the signatures. In addition, there is data expansion of the signatures upon transmission. Popek and Kline [10] and Matyas [8] have proposed arbitrated signature techniques which include the identities of the transmitter and receiver in the signature data. The key notarization system provides arbitrated signatures whereby the identities of the parties involved are employed in the notarization of the data key rather than the data themselves. A key notarization facility does not distinguish signatures from other messages because their implementation is the same.

Secret Key Versus Public Key Protection

Fig. 7 illustrates the protection required for the encryption and decryption keys of secret and public key algorithms. Keys used by secret key algorithms like DES must be protected from both disclosure and substitution. The secret keys used for decryption in public key algorithms must also be protected against both disclosure and substitution while the public keys require substitution protection. The fact that public keys need to be protected from substitution is often overlooked when public key distribution systems are described. For example, if an intruder is permitted to substitute his own public key for that of another user, he may send false signatures posing as the other user.

COMMANDS

The KNF will contain the code to implement the required commands. The commands are used for:

1) data encryption, decryption, and authentication;
2) password initialization, notarization, change, and reencryption;
3) reserving and logging out of active user states;
4) generating notarized data keys and initialization vectors;
5) loading notarized data keys and initialization vectors;
6) reencrypting data keys.

A user must authenticate and reserve an active state before he can execute any other commands. A description of the KNF commands is provided in the Appendix.

ILLUSTRATIVE EXAMPLE

Initialization

Suppose cryptography were to be added to a computer network. First, each host would have to be provided with a KNF and the necessary interface. Then interchange keys would have to be generated and distributed. Once the interchange keys are loaded directly into the KNF's and the authorized users are assigned unique identifiers and passwords, the security officer at each facility can initialize the passwords of the authorized users. (If encrypted terminal to KNF communications were desired, secret keys could be used in place of passwords.)

The Transmitter

A user may then authenticate and become active by submitting the password which corresponds to his identifier. He could also change his password to a secret value known only to himself. Next he may want to generate data keys for encrypting either personal or shared files. When he has an encrypted data key, say E[IKp XOR (i || j)] (DKij), user i can load the key into his active transmit key state. The user or his host must keep track of whether an encrypted key is a transmit, personal, or received key; the sharing party; and for what interchange it is intended. Once the transmit key is loaded, user i may then generate an IV and load it into the transmit active IV storage. After he sends the encrypted DKij and IV to j, he is ready to encrypt data intended for user j. Of course, if he is on-line with user j, he must establish contact with j, identify himself, and send him the encrypted DKij and IV. If he is on-line he should also require an appropriate response from j to ensure that he is being received. He should include a message number, the date, and the time in his plain text so that old valid messages from i to j cannot be played back to j.

User i may generate and load a notarized data key, E[IK1 XOR (i || i)] (DKij), and an encrypted IV, E[DKii] (IV), for the protection of a personal file. He may then encrypt his data. The notarized data key and encrypted IV are either stored with the cipher or in the user's key-IV file. Finally, user i can log out of active status. If not, the system should

automatically log him out after a specified time period or when he logs off the system, whichever comes first.

The Receiver

Once user j is active, and has received the encrypted DKij, IV, and data, he may load his receive active storage. He can then decrypt the cipher. He may also keep the cipher, the notarized data key, and the encrypted IV for signature purposes. If j wishes he can generate a DKji to communicate securely to i, but communiations from j to i will not be encrypted with the same data key as communications from i to j.

SUMMARY

The DES can provide secure authentication and encryption with limited protocol requirements in a variety of network configurations. Rekeying techniques whereby keys are transmitted in encrypted form over an unprotected channel must consider the substitution threat whenever an active tap is feasible. Host operating systems must protect unencrypted files, but hosts need not be relied upon to protect keys from either disclosure or substitution. Key notarization facilities implement key management functions as well as the encryption, decryption, and authentication operations. The secure distribution of data keys is attained by encryption and the use of identifiers for key notarization. The KNS features on-line and off-line applications, local key generation, and a digital signature capability.

APPENDIX

This Appendix describes the commands which implement the KNF for key management and data encryption purposes. Besides encryption, decryption, and authentication, they are used to generate encrypted keys which are given to the user and to provide for the supersession of the keys which are controlled by the system. The commands are invoked by a command name followed by a parameter list of passed and returned values. The user's identifier is shown as a parameter only when it must be supplied by the user of the command. For some commands the system automatically supplies the KNF with the user's identifier. Interchange keys must be loaded into the KNF before commands are executed. User interface programs in the host will prompt the user and call the commands as required. Alternate command sets may be devised to meet varying security applications.

1) INITIALIZE PASSWORD (IPW)
IPW: $\{ui, pw, ps\}$

ui = user identifier

pw = user password

ps = security officer password.

This command is used when a user is first put on the system. The password is encrypted and stored in host system memory. The original password is known to the user and the security officer. The user submits the original password when he first authenticates himself to the KNF, then he immediately changes his password to a secret value, known only to himself, by using the change password command, CPW. Only the security officer who is responsible for putting new users on the system should be capable of initializing the password.

2) DELETE PASSWORD (DPW)
DPW: $\{ui, pw, ps\}$

ui = user identifier

pw = user password

ps = security officer password.

This command is used by the security officer in conjunction with a user to remove a user from the system. The user must supply his password to the security officer before he can be removed. This feature prevents the security officer from removing and reinitializing a user's password without his knowledge.

3) REENCRYPT PASSWORDS (RPW)
RPW: $\{ps\}$

ps = security officer password.

The security officer executes this command after the interchange keys have been changed. Each encrypted password stored in system memory is decrypted using the old facility interchange key and encrypted using the new facility interchange key. The result is then stored back in system memory. This permits a user to authenticate even though the interchange keys have been changed. After he is authenticated and active, it will be the user's responsibility to reencrypt his data keys before using them for encryption, decryption, or data authentication.

4) REENCRYPT FOR INTERCHANGE SWITCHING (RIS)
RIS: $\{ti, ri, oi, ni, ok, rk\}$

ti = transmitter id

ri = receiver id

oi = old interchange

ni = new interchange

ok = old encrypted data key

rk = returned reencrypted data key.

The purpose of this command is to permit a data key to be decrypted and reencrypted under a new interchange key at an intermediate host so as to permit communications between two hosts which do not share a common interchange key. Only the data key and not the data itself needs to be reencrypted. Transmitter and receiver identifiers are not changed.

This command is performed by the operating system or by the host security officer.

5) RESERVE ACTIVE STATE (RAS)
RAS: {*ui*, *pw*, *ss*, *ua*}

ui = user identifier

pw = user password

ss = *y* if active memory is available

= *n* otherwise

ua = 0 if *ss* = *n*

= *y* if *ss* = *y* and PW authenticates

= *n* if *ss* = *y* and no authentication.

This command activates the user by loading the user's identifier into the KNF. Active user memory must be available and the user must authenticate before the identifier is loaded. No other commands may be executed by the user until he has successfully executed RAS. Once authentication is complete, the system must ensure that other users cannot execute commands in place of an authenticated user.

6) LOGOUT ACTIVE USER (LAU)
LAU: {*ui*}

ui = user identifier.

This command may be used by the user when he has finished using the KNF. In this case *ui* is optional. The command removes the user identifier from the active user list maintained in the KNF. All active DK's and IV's belonging to the specified user are lost. The host may also keep a list of active users and the time of the last command executed for each one. If a user has not executed a command after a reasonable time period, then the host may use LAU to log out the user. The user may still be logged on the system but he will have to repeat the RAS command to use the KNF. The system may also periodically decide to challenge a user by requiring him to reauthenticate. Whenever the user logs off the system, the LAU command should automatically be executed.

7) CHANGE PASSWORD (CPW)
CPW: {*op*, *np*}

op = old password

np = new password.

This command is used to change passwords. The old password is authenticated before any change is made. The user identifier must be loaded into active user memory; otherwise an error message is returned.

8) GENERATE DATA KEY (GDK)
GDK: {*in*, *sp*, *ed*}

in = interchange name

sp = identifier of sharing party

ed = returned encrypted data key

ex. (command executed by user *i*)

in = *p*

sp = *j*

ed = *E* [IK *p*XOR (*i* || *j*)] (DK *ij*).

This command is used to generate new keys. The identifier of the user invoking the command, user *i* in the example, is always the leftmost value in the concatenation of the sending and receiving identifiers. If the two identifiers are equal, then the key is personal and cannot be shared. This command may not be executed unless the user is active. Otherwise an error message is returned.

9) LOAD DATA KEY (LDK)
LDK: {*kf*, *in*, *sp*, *ed*}

kf = *t* if key is for transmitted data

= *r* if key is for received data

= *s* if key is for personal data

in = interchange name

sp = identifier of sharing party

ed = encrypted data key

ex. (command executed by user *i*)

kf = *t*

in = *p*

sp = *j*

ed = *E* [IK *p*X OR (*i* || *j*)](DK *ij*)

ex. (command executed by user *i*)

kf = *r*

in = *p*

sp = *j*

ed = *E* [IK *p*X OR (*j* || *i*)] (DK *ji*)

ex. (command executed by user *i*)

kf = *s*

in = *f* (facility interchange identifier)

sp = *i*

ed = *E* [IK *f* X OR (*i* || *i*)] (DK *ii*).

This command loads a data key, either shared or personal, into the user's active state in the KNF. The key is stored at the transmit key address if *kf* = *t*, and at the receive key address if *kf* = *r*. If user *i* executed the command, then *kf* = *s* if and only if *sp* = *i*. Otherwise an error message will be returned. When *kf* = *s* and *sp* = *i* the data key will be loaded into both the transmit and receive locations. The user must be active before this command can be executed.

10) GENERATE INITIALIZATION VECTOR (GIV)
GIV: {*ei*}

ei = returned encrypted initialization vector

ex.

ei = *E* [DK] (IV).

This command is used to generate new initialization vectors. The KNF key generator generates 64 bits and then encrypts them under the data key which must be previously loaded at the transmit address in active user memory. The encrypted IV is returned to the user. The data key may be either personal or shared.

11) LOAD INITIALIZATION VECTOR (LIV)
LIV: $\{kf, ei\}$

$kf = t$ if IV is for transmitted data
$= r$ if IV is for received data
$= s$ if IV is for personal data
ei = encrypted initialization vector
ex.
$kf = t$
$ei = E$ [DK](IV).

If $kf = t$, then the data key at the transmit address is used to decrypt the encrypted IV. The IV is then stored at the transmit IV address. If $kf = r$, then the data key at the receive address is used to decrypt the encrypted IV, and the IV is stored at the receive IV address. When $kf = s$, the transmit data key is used to decrypt and the IV is placed in both the transmit and receive IV locations.

12) REENCRYPT DATA KEY (RDK)
RDK: $\{kf, in, sp, ok, rk\}$

$kf = t$ if data key is for transmitted data
r if data key is for received data
s if data key is for personal data
in = interchange
sp = identifier of shared party
ok = old encrypted data key
rk = returned reencrypted data key

ex. (user j reencrypting a key sent by user i)
$kf = r$
$in = p$
$sp = i$
$ok = E$ [OIK XOR $(i \| j)$](DKij)
$rk = E$ [NIK XOR $(i \| j)$] (DK ij)
OIK = old interchange key for interchange p
NIK = new interchange key for interchange p

This command is used when interchange keys are changed. It reencrypts data keys under the new interchange key so that the data protected by the key does not have to be reencrypted. The user must be active. Also, $kf = s$ if and only if $sp = i$ and user i invoked the command.

13) CHANGE KEY ACCESS (CKA)
CKA: $\{oi, ni, os, ns, ok, rk\}$

oi = old interchange
ni = new interchange
os = old sharing party
ns = new sharing party
ok = old encrypted data key
rk = reencrypted data key
ex. (user i extending access to user j)
$oi = p$
$ni = q$
$os = i$
$ns = j$
$ok = E$ [IKpXOR $(i \| i)$] (DK)
$rk = E$[IKqX OR $(i \| j)$](DK).

This command is used to change the sharing party and interchange of a notarized key. Using this command, the owner of an encrypted file may extend access to additional users without having to reencrypt the file itself. Since the KNF employs the user's identifier as the first element of the identifier pair, only keys owned by the user can be successfully reencrypted by this command.

14) GENERATE AUTHENTICATION CODE (GAC)
GAC: $\{da, nb, md, ac\}$

da = data address
nb = number of bytes of data
md = CBC for Cipher Block Chaining mode
= CFB for Cipher Feedback mode
ac = returned authentication code (eight bytes)

This command uses DES to calculate an eight-byte authentication code on nb bytes of data at da. The data key and IV which have been previously loaded into transmit active storage will be used. The value of md indicates which of two DES modes of operation are desired (see [7]). If the IV and DK are not loaded an error message is returned.

15) VALIDATE AUTHENTICATION CODE (VAC)
VAC: $\{da, nb, md, ai\}$

da = data address
nb = number of bytes of data
md = CFB for cipher feedback
CBC for cipher block chaining
ai = returned authentication indicator.

This command validates the authentication code which follows nb bytes of data stored at address da. md indicates the DES mode of operation. The data key and IV, which have been previously loaded into receive active storage, will be used. The authentication code is calculated and compared with the stored value. The returned value of ai equals 1 if the codes

compare and 0 otherwise. If the IV and DP are not loaded an error message is returned.

16) ENCIPHER DATA (ENC)
ENC: $\{pt, ct, nb, md\}$

pt = plain text

ct = cipher text

nb = number of bytes

md = CBC for cipher block chaining mode
CFB for cipher feedback mode.

This command enciphers nb bytes of data stored at pt and returns the cipher to ct. The mode of operation is indicated by md. The transmit DK and IV which must be previously loaded into active memory are used by the encryption algorithm. If the required DK and IV have not been loaded an error message will be returned.

17) DECIPHER DATA (DEC)
DEC: $\{pt, ct, nb, md\}$

pt = plain text

ct = cipher text

nb = number of bytes

md = CBC for cipher block chaining mode
CFB for cipher feedback mode

The command deciphers nb bytes of data stored at ct and places the results in pt. The mode of operation is indicated by md. The receive DK and IV which must have been previously loaded into active memory are used by the decryption algorithm. If the required IV and DK values have not been loaded an error message will be returned.

ACKNOWLEDGMENT

The author would like to express his appreciation to Dr. D. Branstad for his many valuable suggestions. The referees also provided several comments which greatly enhanced the paper.

REFERENCES

[1] D. K. Branstad, "Encryption protection in computer data communications," presented at the IEEE 4th Data Commun. Symp., Oct. 7–9, 1975.

[2] *Data Encryption Standard*. National Bureau of Standards (U.S.), Federal Information Processing Standards Publication (FIPS PUB) 46, National Technical Information Service, Springfield, VA, 1977.

[3] D. E. Denning, "Secure personal computing in an insecure network," *Commun. Ass. Comput. Mach.*, vol. 22, Aug. 1979.

[4] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, Nov. 1976.

[5] W. F. Ehrsam, S. M. Matyas, C. H. Meyer, and W. L. Tuchman, "A cryptographic key management scheme for implementing the data encryption standard," *IBM Syst. J.*, vol. 17, no. 2, 1978.

[6] J. Everton, "A hierarchical basis for encryption key management in a computer communications network, trends and applications," in *Proc. 1978 IEEE Comput. Soc.*, May 1978.

[7] *Information Processing Standards*; Federal: DES Modes of Operation; Inquiry, Federal Register, vol. 45, no. 101, book 1, pp. 34322–34336.

[8] S. M. Matyas, "Digital signatures—an overview," *Comput. Networks*, vol. 3, Apr. 1979.

[9] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Commun. Ass. Comput. Mach.*, Dec. 1978.

[10] G. J. Popek and C. S. Kline, *Encryption Protocols, Public Key Algorithms and Digital Signatures in Computer Networks, Foundations of Secure Computations*. New York: Academic, 1978.

[11] M. Rabin, *Digitalized Signatures, Foundations of Secure Computation*. New York: Academic, 1978.

[12] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. Ass. Comput. Mach.*, Feb. 1978.

★

Miles E. Smid was born in Chicago, IL, in 1944. He received the B.S. degree in mathematics from the University of Chicago, Chicago, IL, in 1966 and the M.A. degree in mathematics from the University of Maryland, College Park, in 1969.

From 1967 to 1978 he worked for the Department of Defense specializing in computer cryptography and computer security. He evaluated existing tactical data systems as well as proposals for future computer networks. His work included a security assessment of the Software Status Authentication System for the Minuteman missile program. He is currently employed by the National Bureau of Standards to develop standards and techniques for the application of the Data Encryption Standard (DES). He is a member of the American National Standards Institute's working group on Financial Message Authentication (X.9.E8). Of special interest to him is the integration of DES into computer systems for secure data storage and communications.