Information being transferred from point to point over a public communications carrier or stored on portable media can be protected, by the use of cryptography, from accidental or intentional disclosure. Control functions are required to ensure synchronization of the process. In a communications environment, the control functions become logically part of the network architecture. IBM's Systems Network Architecture (SNA) has been extended to allow the use of cryptography when sensitive information is being processed. Architectural similarities for the file environment are discussed.

Cryptography architecture for information security

by R. E. Lennon

The value of information can be measured in terms of the financial loss that would be incurred if the information were lost. In data processing and data communications, security measures are used to protect against such loss. When information cannot be physically secured from unauthorized access, as when it is stored in shared systems or transmitted over insecure communications links, the information can be protected from disclosure or modification through the use of cryptography. Information passing across a public telecommunications carrier, whether a telephone line, microwave link, or satellite link, is not physically secure. Similarly, information stored on a portable medium such as a reel of tape or a disk pack, which then must be or might be removed from the normal control of its owners or managers, may not readily be secured. The threat of disclosure or modification can be countered, however, by using cryptography to transform data into an unintelligible form that is useless to all but authorized persons who can reconstruct the original data.

Certain control functions must be established to ensure that the use of cryptography will not lead to disruption of normal activity. In a communications environment, control functions must be included in the network architecture, which, by definition, establishes the rules, or protocols, for the orderly movement of information through the network. Similarly, control functions are required at the architectural level to allow consistent implementations of cryptography by the various processes that manipulate information retained on secondary storage media, such as tapes and disk packs.

The inclusion of cryptography at the architectural level allows it to be woven into the fabric of the system. The security of the cryptographic function is then equal to that of the system.

Cryptography is not the total solution to the general data security problem. Cryptography can provide a reasonable measure of protection against such threats to security as eavesdropping, misrouting, masquerading, and scavenging by active or passive techniques. To properly address the general data security question, one needs a total security plan, within which cryptography can play only a part. For example, the use of cryptography without meaningful authorization and access controls will not result in increased data protection.

A fundamental building block on which cryptography is based is the algorithm used in preparing cryptograms. The algorithm used in IBM products is the Data Encryption Standard (DES),[1] the output of which is a function of a secret, variable quantity termed a *key*. A key is represented by 56 independent key bits and eight dependent parity bits. There are 2^{56} different possible keys.

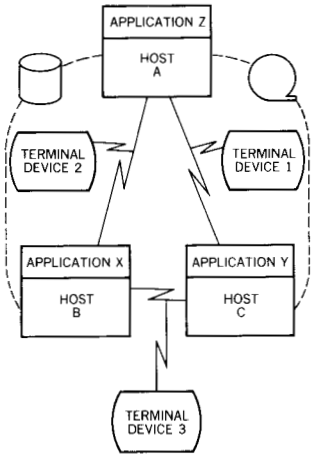
While cryptography can be applied to data, other applications are feasible. Cryptography can serve as an adjunct to access control, in that tables containing identification and perhaps password data can themselves be enciphered. The IBM 3600 Banking System uses such an approach with a private form of cryptography.[2]

A number of IBM products support the use of cryptography.[3] In a communications environment, the operation of these products is governed by Systems Network Architecture (SNA).[4] A major part of this paper discusses the changes and additions to SNA that enable it to support cryptography.

Communications networks

A general communications network is illustrated in Figure 1. Disregarding cryptography for the moment, it is apparent that some form of network management is required for data to move efficiently through the network. SNA regulates the orderly flow of data between an application program in a host computer and either a terminal or another application program.[4] When cryptography is introduced into such a network, there are some special requirements that must be considered. For example, it is desirable that the cryptography function be provided in a manner that is transparent to the end user so that the function can be used without requiring changes to application programs. Transparency is provided, with maximal implementation flexibility, by designing cryptography into the transmission control element of SNA.[5] In addition, the use of cryptography during a particular session

Figure 1 General network



must be communicated to each node, and a means of agreeing or disagreeing on its use is required. A key to be used during the session must be derived and distributed to each node. A verification procedure is required to validate the cryptographic competency of the nodes once a session is established. Finally, a mechanism is required to report and recover from any errors or failures during cryptographic operations.

definitions of keys

Since data is intended to be protected by secret cryptographic keys, data security is a function of key secrecy. The approach taken here is that keys must never appear in the clear outside the cryptographic facility except for a brief moment during key generation and while a key is being installed in a specific implementation of the DES algorithm. Therefore, when a key has to be stored or distributed to a remote location, it is protected by being enciphered under another key.

Two classes of keys are defined: *data-encrypting keys*, or *data keys*, and *key-encrypting keys*. Data keys are used to encipher and decipher data. Key-encrypting keys ensure that a key never appears in the clear outside the cryptographic facility.

There are two forms of key-encrypting keys, *primary* and *secondary*. A primary key-encrypting key, also called a *master key* and designated *KM*, protects data keys and other key-encrypting keys. It is stored in the clear within a cryptographic facility. A master key associated with a host processor is designated *KM*0, and a master key associated with a terminal is designated *KMT*. Variants of the host master key are designated *KM*1 and *KM*2. Secondary key-encrypting keys are used to encipher data keys. Secondary key-encrypting keys are stored in enciphered form, enciphered under a primary key-encrypting key.

SNA defines a *system services control point* (SSCP) as the control function for all communications activity within a portion of a network called a *domain*. Through a definition process, tables are constructed that describe to the SSCP the configuration and characteristics of its domain, including the cryptographic capability of appropriately equipped resources. A key management function also is located at the SSCP, in part by providing a key table within which are stored, enciphered under either *KM*1 or *KM*2, the master keys associated with each implementation of the DES algorithm within the domain. Through key transformation functions defined in Reference 6, the SSCP can assume the role of cryptographic resource manager for its domain.

For interdomain communication—that is, for communication between SSCPs—it is necessary to exchange data keys securely between the SSCPs. For this purpose, a secondary key-encrypting key, *KNC*, is used to encrypt the data keys. In SNA, this second-

ary key-encrypting key is called a *cross-domain key*. Usually cross-domain keys are defined in pairs, one key being used in transmitting data keys, and the other in receiving them. By definition, then, these keys are unidirectional; therefore a data key cannot be compromised by the application of an inverse transformation.

With cross-domain keys, one SSCP need not know the host master key associated with another SSCP. Therefore the security of the data exchanged between the systems is enhanced.

For a full discussion of key management, see the paper by Ehrsam et al.[6] in this issue.

Communications security

architectural levels

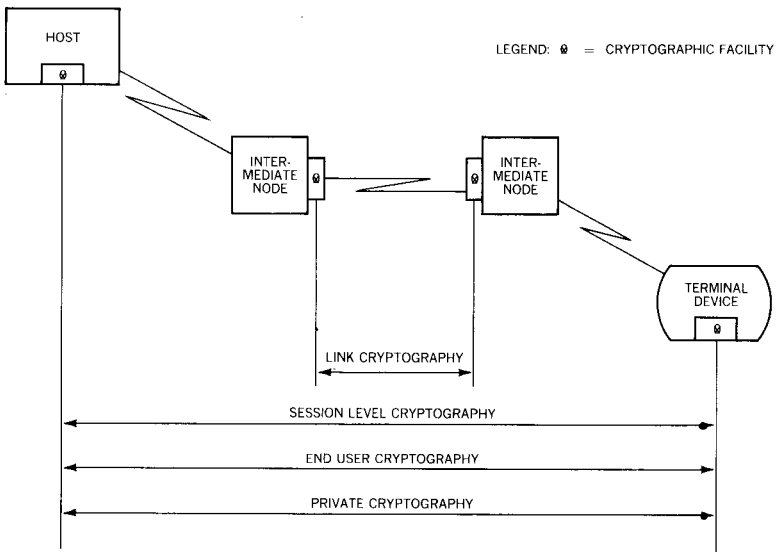
Key selection and management, as well as deciding when and how often to use cryptography, give rise to several architectural levels of cryptography:

- In *session-level cryptography,* SNA protocols are used to manage cryptography during a session between communicating nodes.
- In *end-user cryptography,* SNA protocols are used for key selection and key distribution, but the end user provides his own rules and protocols regarding the application of cryptography.
- In *private cryptography,* key selection and distribution, as well as management of the application of cryptography, are performed by the end user according to his own rules and protocols. The use of cryptography is known only to the end user, not to the system; it is transparent to, and not in conflict with, SNA. (An example is the use of cryptography in the IBM 3600 system, cited earlier.)

In each of these levels, data is enciphered at the point of origin and deciphered at the destination, thus providing *end-to-end* data protection. Provisions have been made in SNA to allow these levels to be specified. More than one level can be specified, depending on the user's requirements.

Other architectural levels of cryptography are possible. One, called *transparent link cryptography*, involves the attachment of a stand-alone cryptographic device at either end of a communications link over which cryptographic protection is required. If the link is bracketed in this fashion, all traffic passing over it is subjected to cryptographic processing. This mode of operation sometimes is referred to as *point-to-point* cryptography.

Figure 2 Communication security architectures



Generally speaking, the stand-alone devices are self-synchronizing, and their operation is transparent to the operating system that manages the link. The devices must use the same algorithm, with identical keys. On a multipoint line, every drop point would require a cryptographic device, each containing the same key. As a rule, such a device can be made independent of a line discipline if a stream cipher mode of operation is used. However, a stream cipher, whose underlying generator could be the DES, requires an initializing vector to ensure synchronization between the sending and receiving devices.[7] IBM products that support link encryption are implemented in this manner.

With link cryptography, information is protected only when passing over the specified link. End-to-end protection, as provided by SNA, is not possible. Link cryptography can coexist with SNA-supported cryptography, with the result that information becomes doubly encrypted, under different keys, while on the specified link.

Figure 2 illustrates the paths through a network that are protected by encryption based on the architectural levels described above.

operating modes

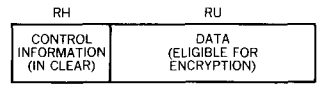
Session-level cryptography can operate in any of three modes:

- *Transparent cryptography.* If implemented, the cryptography function as provided by the network nodes is transparent to the end user; that is, cryptography is provided automatically. Cryptography might be implemented, for example, because of

the location of a particular terminal, or because of the nature of the data to be processed by a particular application program.

- *Application-directed cryptography*. If authorized by the installation, a particular end user can request the use of cryptography for a given session. The requesting user can then select which outbound messages are to be enciphered and which are not, while incoming data requiring decipherment can be identified by means of a flag accompanying the data.
- *Mandatory cryptography* is a special case of transparent cryptography. This operating mode requires both participating nodes to encipher all outbound and decipher all inbound messages. The accompanying flag is set to maintain consistency with other features such as *trace*. Enciphered data should not be traced because, in its enciphered form, it has no value as a debugging aid.

Figure 3 Request response header and request response unit

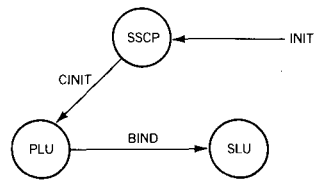


SNA cryptography architecture

Communication between network nodes (application programs and terminals) is based on logical connections called *sessions*. Once a session is established between two nodes, information can begin to flow.[8] Cryptography can be specified as a session parameter during session initiation.

The basic element of data exchanged between network nodes during a session is a *request-response unit* (RU), which can be thought of as a logical message. It represents the unit of encryption. The RU is preceded by a *request-response header* (RH), which contains routing information for the associated RU. The RH is not enciphered. A flag bit in the RH indicates whether the associated RU is enciphered. The RH-RU relationship is illustrated in Figure 3. Cryptography can be specified as a session parameter through an extension to the SNA BIND command, which establishes a session. Bits in BIND specify the various session characteristics, including cryptography, that can exist for the session being bound.

Figure 4 SNA session initiation command flow



SNA command flow

Session initiation is the process by which one network node notifies another that it wishes to establish a communications session and by which the nature of the session is agreed upon. SNA provides a set of commands that allow network nodes to specify and agree on the manner in which an orderly transfer of data will be accomplished. Extensions have been added to this existing set of commands to satisfy the requirements of cryptography, including the selection, distribution, and verification of cryptographic keys. A logical view of the commands and the direction of their flow are illustrated in Figure 4, in which SSCP indicates the system services control point, and PLU (the primary logical unit) and SLU (the secondary logical unit) represent network nodes.

session initiation

The session initiation process begins when an INITIATE (INIT) command is received by the SSCP. Typically, at an operator controlled terminal, INIT is created when the operator logs on. (Other methods of creating INIT are not relevant to this discussion.) The SSCP, which resides in a host processor, has a table available that provides a complete description of its domain—the network or portion of the network that it manages. From the descriptive table, the SSCP can determine whether cryptography is possible as a session characteristic. From the same table or from LOGON parameters, the SSCP determines whether the cryptography function is required or requested for the session corresponding to INIT. (The cryptography selection process is discussed in detail later in this paper. The reader may wish to refer to Figure 7 for clarification of SSCP actions during session initiation.) An error condition is created if a function is requested and one or both of the candidate logical units cannot support it.

Where cryptography is possible, the SSCP uses a pseudorandom-number generating function[6] to obtain a data key and prepares to distribute it to the logical units involved in the session. This key is called a *session key* (*KS*). Since it is chosen pseudorandomly, it is different for every session established. Further, it is defined as enciphered under the host master key (*KM*0), so it never appears in the clear outside the protection of the cryptographic facility in which it is used to encipher and decipher data. The encipherment (E) of *KS* under *KM*0 is denoted by $E_{KM0}(KS)$. Another form of *KS* must be sent to the secondary logical unit (SLU). Assuming that a terminal master key (*KMT*) has been installed in the SLU, a transformation is performed, using the host's representation of *KMT* and the enciphered session key just generated, to yield $E_{KMT}(KS)$.[6]

The quantities $E_{KM0}(KS)$ and $E_{KMT}(KS)$ are then inserted into defined fields in a CONTROL INITIATE (CINIT) command and the BIND image (which is imbedded in CINIT) for distribution to the PLU and, eventually, the SLU, as illustrated in Figure 4. In transmitting CINIT to the PLU, the SSCP notifies the PLU that a request has been received for a session with an SLU. As implemented by SNA, a PLU is a host-resident application program. An SLU can be either another host-resident application program or a logical unit residing in a cluster controller or terminal.

Upon receiving CINIT, the PLU can accept or reject the invitation to go into a session with the SLU, regardless of the cryptographic level specified. When the PLU accepts, implying that it agrees to the use of cryptography, the quantity $E_{KM0}(KS)$ is extracted from CINIT and maintained in storage for use later in the session. The BIND image may or may not be altered, depending on the session characteristics specified and those the PLU is required to enforce.

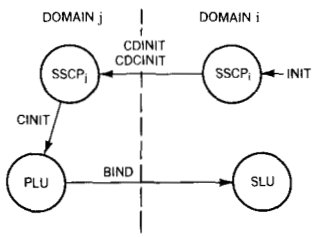
In either case, the PLU converts the BIND image to a BIND command, which, containing the quantity $E_{KMT}(KS)$, is routed to the SLU.

Upon receving BIND, the SLU can accept or reject it according to the session parameters specified and those it is required to enforce. Should it accept BIND, it must extract the quantity $E_{KMT}(KS)$ and save it for use later in the session.

By means of this dialog, each of the logical units participating in the session is provided with a copy of the session key, in a form suitable for use with the cryptography facility at each location. Similarly, through the session-level cryptography bits in BIND, the logical units agree to the use of cryptography as a session characteristic.

One further step is required to complete the session initiation process. The use of cryptography must be verified for the newly bound session. SNA protocol requires the SLU to respond to BIND, positively if in agreement and negatively if not. If cryptography was specified, the positive BIND response is appended with a pseudorandomly chosen 64-bit number *RN*, enciphered under the session key just received. The resulting quantity, expressed $E_{KS}(RN)$, is returned to the PLU with the BIND response. For sessions bound as cryptography sessions, the PLU is required to initiate a CRYPTOGRAPHY VERIFICATION (CRV) command, which contains the partial inverse (first 32 bits) of the pseudorandom number derived at the SLU and enciphered under the established session key. That quantity, represented as $E_{KS}(\overline{RN})$, is sent from the PLU to the SLU.

When the SLU receives CRV, it compares the received pseudorandom number *RN* with that sent in the BIND response to verify that they are the same. Because *RN* is pseudorandomly chosen and therefore nonrepeatable, this procedure, called *handshaking*, defeats the use of a recording device to play back enciphered information through a terminal in hopes of having the terminal decipher the information. Handshaking also verifies that the PLU's cryptographic capability is operating correctly and is equipped with an identical session key. Having verified the validity of its communicating partner, the SLU can respond positively to CRV, enabling information to flow between the two bound logical units, within the constraints of SNA and subject to the cryptographic protocol agreed upon for the session.



interdomain communications

The addition of another domain, managed by another SSCP, adds a step to the session initiation process. Figure 5 illustrates the multidomain case (compare Figure 5 with the single-domain case illustrated in Figure 4). The significant difference between Figures 4 and 5 is the addition of a path called a *cross-domain link* be-

tween SSCPs. It is over this link that SNA supports a special session known as a *cross-domain session*, or *SSCP-SSCP session*, which plays an important part in establishing cryptographic sessions between network nodes in different domains.

As in a single domain, the cross-domain session initiation process begins when an SSCP receives an INIT command, which designates the desired logical unit. The SSCP determines that the logical unit is not in the immediate domain and initiates cross-domain communication to the logical unit's SSCP. The CROSS DOMAIN INITIATE (CDINIT) command allows one SSCP to notify another that a network node in one domain wishes to go into session with a network node in another domain. No extensions to CDINIT are required for cryptography.

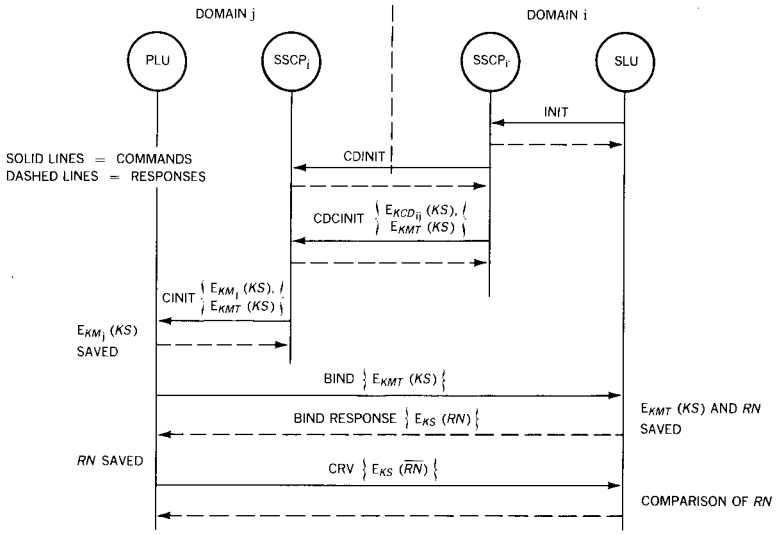
A positive acknowledgment results in the creation and transmission of the CROSS DOMAIN CONTROL INITIATE (CDCINIT) command. Like CINIT in a single domain, CDCINIT contains information describing the desired session. It is through CDCINIT that session keys are transferred from one domain to another.

As described in Figure 5, $SSCP_i$, receiving INIT, being the logical "owner" of the SLU, is responsible for key selection and distribution. Having the SLU's master key in its key table, the initiating SSCP can manage the required key transformations. In addition to creating the value $E_{KMT}(KS)$, the initiating SSCP must transform the session key so that it can be transmitted to the other SSCP involved in the session initiation process. The result of this second transformation is encipherment of the session key under the cross-domain key, KCD_{ij}, which was established for communication between $SSCP_i$ and $SSCP_j$. The notation $E_{KCD_{ij}}(KS)$ describes the session key enciphered under the cross-domain key. This quantity is inserted into the CDCINIT command, as is the BIND image containing $E_{KMT}(KS)$, described earlier. Upon receiving CDCINIT, $SSCP_j$ must extract $E_{KCD_{ij}}(KS)$ and transform it for use with $SSCP_j$'s cryptographic facility. This transformation yields the quantity $E_{KM0_j}(KS)$, which is passed to the PLU in CINIT. Remaining processing is the same as that described for a single domain. Figure 6 presents an overview of the logical command flow and the keys involved in establishing a cryptographic session between two logical units in different domains.

sessions between application programs

It is possible to have cryptographic sessions between two application programs residing in different host processors. Key management functions require an additional transformation, however, because application programs do not have cryptographic facilities of their own, as do terminals. An application program shares the cryptographic facility of the host in which it resides. The operations required for the additional transformation are provided in SNA.

Figure 6 SNA command flow—cryptographic session initiation



When an application program functions as an SLU, it must receive and process the BIND command the same way a terminal does. Similarly, the SSCP that processes the INIT command must create a session key and encipher it under the equivalent of a terminal master key. Therefore the master key table must appear logically the same for application programs as for terminals.

In reality, the key associated with an application program is a dummy key, which is used only for key transformations and is not associated with a specific cryptographic facility. It follows, then, that the key value extracted from BIND cannot be used directly, but must be transformed to make it suitable for use with the cryptographic facility at the application program's host.

The selection of cryptography

When implemented, cryptography can be selected at any of three levels: by installation management, by a terminal operator, or by an application program. The system definition process allows installation management to specify the cryptographic capability of network nodes. Moreover, management can define a given node as secure, implying that it has cryptographic capability which is to be used in all sessions in which the node participates. The terminal operator can select cryptography as a session option when he logs on. The use of cryptography can be selected by an application program as part of the OPEN process to a system teleprocessing service such as ACF/VTAM (Advanced Communications Function Virtual Telecommunications Access Method).[8, 9]

Figure 7 Cryptography selection process

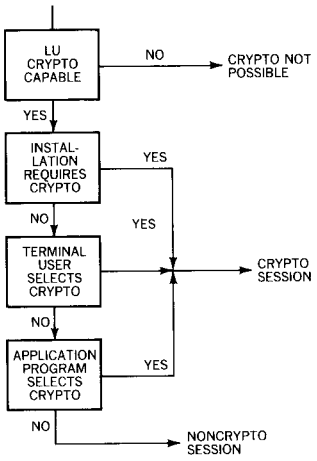


Figure 7 illustrates a natural hierarchy of decisions involved in the selection of cryptography. If cryptography is possible, once it is selected it cannot be deactivated for the duration of the session. Generally speaking, the selection process shown in Figure 7 is the same as that for other session characteristics allowed in SNA. Note that the session initiation function is controlled by the SSCP.

Padding

The DES algorithm accepts as input discrete 64-bit (eight-character) blocks of cleartext and converts them into 64-bit blocks of ciphertext, the result being a function of the specified data key.[6] Streams of data are broken down into eight-character blocks, and each block is processed by the algorithm separately. Fractional remainders, or blocks of less than eight characters, cannot be processed directly by the DES algorithm. In a communications environment, this restriction can be overcome by padding messages to a length divisible by eight.

SNA calls for adding up to seven random pad characters to a message, the last character being a count of the number of pad characters added, before the message is presented to the algorithm for encipherment. A special flag bit is then set in the accompanying request-response header (RH) to indicate to the receiver that, after decipherment, the message must be stripped of pad characters. Thus irregularities in message length are made transparent to cryptographic processing.

Although padding is suitable for a communications environment, storage restrictions may not tolerate padding—that is, the expansion of data—in a file environment. Therefore, in a file environment, an alternative technique is used for enciphering fractional and short blocks of data.

File security architecture

A disk pack or reel of tape is not, in itself, a secure medium. Unless a disk pack or tape reel is physically protected in a safe or vault, the information contained on the recording surface can be exposed. Cryptographic processing of data transcribed to such portable media will prevent exposure of the data if it is accidentally or intentionally probed.

Cryptography cannot be applied arbitrarily, as is demonstrated earlier in this paper, because there is a danger that enciphered data might become undecipherable and therefore lost. Note that the effort required to recover encrypted data, when the data-encrypting key has been lost, is equal to the effort that would have

to be expended if the key were unknown.[6] Using the fundamental methodology described above for a communications environment, a cryptography architecture can be developed for the file environment.

An analogy exists between units of encryption in communications and file environments. Entire messages and fields within messages are analogous to logical records and fields within logical records. It follows, then, that a secure session, in which all messages are cryptographically processed, is equivalent to an enciphered file, in which all logical records are enciphered. From a cryptographic point of view, the size of a file does not present a limitation. Enciphering and deciphering operations are not affected by a file that exceeds the storage capacity of a single volume, or several volumes.

Having recognized an equivalency between units of encryption in the communications and file environments, we can define the following file security architectural levels:

- *Private file security*. Key selection and management are the responsibility of the application program. The use of cryptography is transparent to system file management services.
- *End-user file security*. Key selection is provided by system cryptography services. Key management is the responsibility of the application program. The use of cryptography remains transparent to system file management services.
- *System file security*. System file management services also provide cryptography services which may be either transparent to or directed by the application program.

These levels of file security can be used in combination, possibly resulting in double encryption, with either the same or different keys.

It is possible to define a master file key that is the analog of a terminal master key. This secondary key-encrypting key is used to protect a file key, which is analogous to a session key. The main difference between a session key and a file key is the length of time during which the key has significance. A session key is relatively short-lived; it is used only for the duration of a session. A file key, on the other hand, must be retained for the life of the file it protects, possibly for years. Since the file key usually is a pseudorandom value generated by the system, it cannot be recreated. Therefore a form of the file key must be maintained in a secure manner for the life of the file. The chosen form must allow for the recreation of the file key to permit deciphering when desired. Enciphering the file key with a secondary key-encrypting key protects the true value of the file key and allows retrieval by means of the same functions used to transform a session key from

encipherment under a cross-domain key to encipherment under a host master key. This relationship between the master file key and the file key simplifies the key retrieval process without weakening the security of the file key.

Summary

SNA provides the foundation for the orderly movement of information from source to destination in a communications network. The addition of a cryptography function, as a means of protecting information from disclosure or deliberate modification during passage through the network, affects the architecture in terms of selection, distribution, and verification of the function. By associating cryptography and an architecture, an otherwise unprotected communications facility or portable storage medium becomes practically as secure as a physically protected host processing center.

ACKNOWLEDGMENT

The author acknowledges Mr. J. Oseas for his direct assistance in the preparation of this paper, as well as his many contributions to SNA.

CITED REFERENCES

1. *Data Encryption Standard*, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington, DC (January 1977).
2. *IBM 3600 Finance Communication System—3614 Programmer's Guide and Reference Manual*, IBM Systems Library order number GC66-0002, IBM Corporation, Department A60, 5600 Cottle Road, San Jose, California 95193.
3. *Data Security Through Cryptography*, IBM Systems Library order number GC22-9062, IBM Corporation, Department 63T, Neighborhood Road, Kingston, New York 12401 (1977).
4. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15,** No. 1, 4–23 (1976).
5. P. G. McCullum, "The transmission subsystem in Systems Network Architecture," *IBM Systems Journal* **15,** No. 1, 24–38 (1976).
6. 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 Systems Journal*, **17,** No. 2, 106–125 (1978, this issue).
7. C. H. Meyer and W. L. Tuchman, "Data security for communications systems," *Proceedings of the First Southeast Asia Regional Computer Conference (SEARCC)* (Singapore), North Holland Publishing Company, New York (1976), pp. 557–573.
8. H. R. Albrecht and K. D. Ryder, "The Virtual Telecommunications Access Method: A Systems Network Architecture perspective," *IBM Systems Journal* **15,** No. 1, 53–80 (1976).
9. *Introduction to Advanced Communications Function, Multiple System Data Communication Networks*, IBM Systems Library, order number GC30-3033, IBM Corporation, Data Processing Division, 1133 Westchester Avenue, White Plains, New York 10604.

Reprint Order No. G321-5068