CHAPTER SEVEN

Incorporation of Cryptography into a Communications Architecture[1]

Today's versatile, powerful, and complex computer networks have evolved from a modest beginning in the 1950s when users first accessed computer resources from remote terminals. As the evolution proceeded, so did attempts to replace ad hoc network designs with systematic approaches based on defined parameters [2–6].

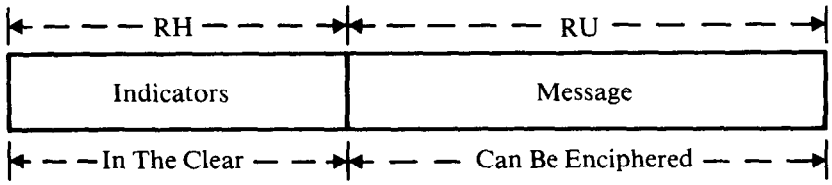
Fundamental to this approach is a *protocol* (set of agreements) which presents a basis for controlling information transfer within a communications network. Collectively, such protocols, referred to as a *communications architecture*, put the parties served by a network into communication with each other. One architectural approach, IBM's Systems Network Architecture (SNA) [2, 3], is used here to show how cryptography can be incorporated into a communications network. The specifics, discussed here to illustrate how the SNA architecture can be structured to support cryptography, lead to broader concepts which are applicable to other architectures.

In SNA terminology, application programs and terminal devices equate to logical units (LUs). Data transfer between two LUs may occur after a logical connection, or *session*, has been established [4]. Cryptography can be specified as a session parameter at the time a session is established. When this method of protection is in effect, data are enciphered by the originating LU and deciphered only by the destination LU; thus end-to-end protection is achieved (see Chapter 4).

Three architectural levels of cryptography are defined within SNA: session, end user, and private. In *session-level cryptography*, SNA protocols are used by the system to manage cryptography during a session between communicating LUs. In *end-user cryptography*, SNA protocols are used by the system for key distribution, but the end user provides his own rules and protocols regarding the use of cryptography. In *private cryptography*, key selection and distribution, as well as management of the use of cryptography, is performed by the end user according to his own rules and protocols. Because, in this latter case, the use of cryptography is known only to the end user and not to the system, it is transparent to and not in conflict with SNA.

The basic information element that flows between LUs during the LU-LU

[1] Reprinted from *IBM Systems Journal* **17**, No. 2, 1978 [1].



RH - Request/Response Header (contains RU descriptor flags)
RU - Request/Response Unit

Figure 7-1. RH/RU Relationship

session is a request/response unit (RU). It contains either user data or control information (Figure 7-1). The RU is preceded by a request/response header (RH). Only a data request unit is enciphered; the RH remains in the clear. A bit in the RH indicates that the RU contains enciphered data.

The installation, through a definition process, specifies each LU's cryptographic capability, that is whether the LU is equipped with or has access to a cryptographic facility (see The Cryptographic Facility, Chapter 4). Furthermore, particular LUs may be declared secure components, thus making cryptography mandatory for every session in which they participate. The terminal operator may, via the LOGON procedure, select cryptography as a session option. An application program, as part of the OPEN process of a system teleprocessing access method, such as IBM's ACF/VTAM (advanced communication function/virtual telecommunications access method), may request the use of cryptography for the pending session (Figure 7-2). Once cryptography is selected for communicating LUs, it cannot be disabled for the duration of the session.

Three levels of session cryptography are defined: (1) selected cryptography in a transparent mode of operation, (2) selected cryptography in a non-transparent or application-directed mode of operation, and (3) mandatory cryptography.

Transparent cryptography results when the selection of cryptography is unknown to the participating end users. Cryptography may be specified by the installation, perhaps based upon the physical characteristics and not necessarily the logical characteristics of an LU. If selected, cryptographic services are provided by the system, transparent to the end users.

Application-directed cryptography results when the end user makes a specific request for the use of cryptography during a given session. An application program may select which outbound messages are to be enciphered and which are not. Similarly, by using the indicator in the RH, inbound messages can be identified as being enciphered, thus requiring decipherment before being processed.

Mandatory cryptography is a subset of transparent cryptography. As the name implies, this level requires both participating LUs to encipher all outbound messages and to decipher all inbound messages. In this case, the indicator bit in the RH is ignored (as to what or what not to decipher), although it may continue to be set to maintain consistency with other system services.

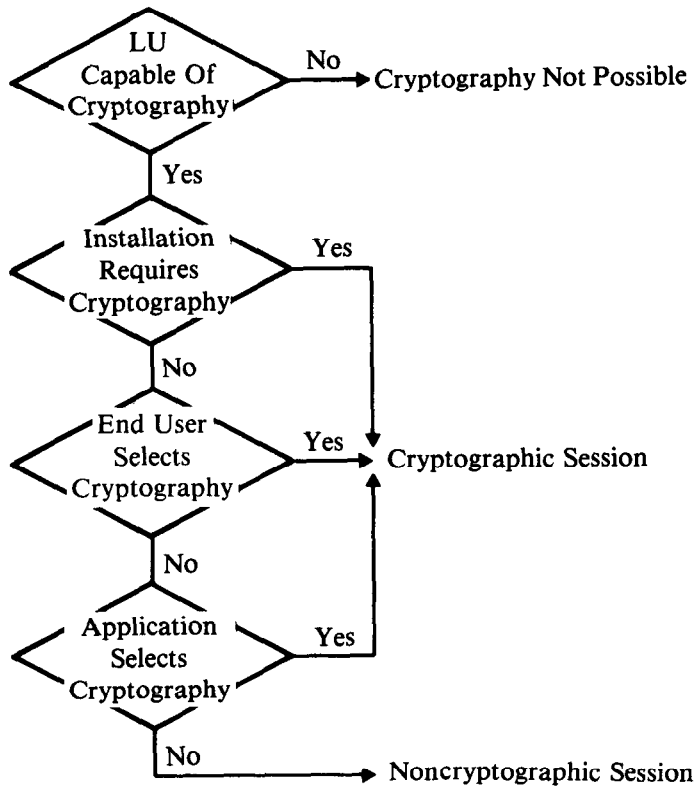


Figure 7-2. Cryptography Selection Process

SESSION-LEVEL CRYPTOGRAPHY IN A SINGLE-DOMAIN NETWORK

Transparent Mode of Operation

SNA provides a set of commands to allow LUs within the communications network to specify and agree on the manner in which the orderly transfer of information from one LU to another will be accomplished. The addition of cryptography, as a means of protecting information from disclosure during its passage through the network, affects the SNA communication network from the standpoint of selection, distribution, and verification of the function. The SNA commands affected by the implementation of communication security will be described.

Figure 7-3 provides a logical view of both commands significant to session initiation, and the network elements between which they flow. The notation **PLU** (primary logical unit) and **SLU** (secondary logical unit) is used because the clarifiers secondary and primary aid in establishing the relationship between the two nodes [3].

Typically, in a terminal-to-application program communications session, the application is the PLU and the terminal (with an installed master key,

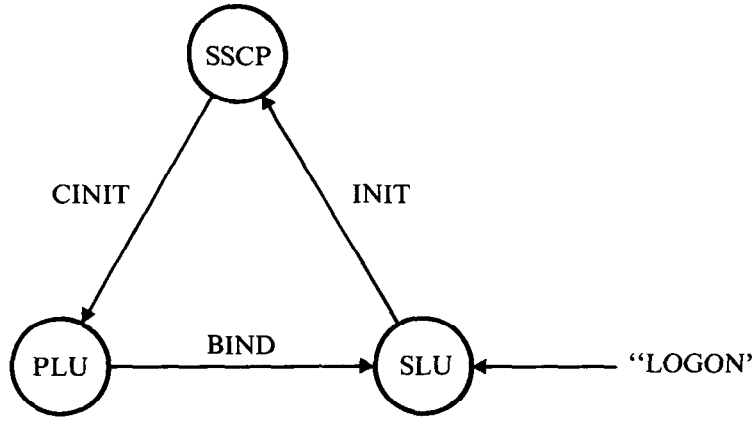


Figure 7-3. SNA Session Initiation Command Flow

KMT) is the SLU. The session initiation process begins with the receipt of an INITIATE (INIT) command at the system services control point (**SSCP**). The SSCP resides in a host node and is the manager of all sessions between communicating LUs within its *domain* of control. The INIT command is typically generated as the result of a LOGON request (to establish contact with the system) entered at an SLU. An INIT command can also result from actions taken by the host operator or host application programs.

The SSCP resides in a host processor and has available to it tables, built from definition parameters, which completely describe the network or portion of the network that it manages (i.e., the domain of the SSCP). From these tables, the SSCP can determine if cryptography is supported by an LU. Additionally, from the same tables or from LOGON parameters, the SSCP determines if the cryptography function is required or requested for the session corresponding to the INIT. (Refer to Figure 7-2 for clarification of SSCP actions during session initiation.) An error condition occurs if one or both of the candidate LUs cannot support a requested function.

If the use of cryptography is possible, the SSCP obtains a randomly generated number via GENKEY (see Key Management Macro Instructions, Chapter 4), which it defines to be the session key enciphered under the host master key (i.e., in the form $E_{KM0}(KS)$). $E_{KM0}(KS)$ is then inserted in the appropriate field of the CONTROL INITIATE (CINIT) command. If the SLU happens to be a terminal device, then the session key is additionally enciphered under the terminal master key ($E_{KMT}(KS)$) and this quantity is then inserted in the BIND image (which consists of a number of fields in the CINIT command). (The BIND image contains information which specifies the characteristics of the session when established.)

The SSCP forwards the CINIT command to the specified PLU, thus indicating to the PLU that there is a request for a session to be established with an SLU. A PLU, as implemented in SNA, is a host-resident application program. An SLU may be another host-resident application program, or an LU residing in a control unit or terminal.

Upon receipt of CINIT, the PLU can either accept or reject the invitation to go into session with the SLU, regardless of the cryptography level specified. When the PLU accepts, implying also that it acknowledges the use of cryptography, it extracts the session key in the form $E_{KM0}(KS)$ from CINIT and saves it for later use. The PLU uses this quantity for ciphering data which is communicated during the session. The PLU converts the BIND image into a BIND command. The BIND command is then transmitted to the SLU. Upon receipt of BIND, the SLU (if it accepts BIND) extracts from it the quantity $E_{KMT}(KS)$, saving it for later use in the session. The result of this dialogue is that the two participating LUs are each provided with a copy of an identical session key in a form suitable for use with their respective cryptographic facilities.

One additional step is required to complete the process of session initiation (Figure 7-4).[2] It involves an action on the part of the SLU to verify that the PLU has an identical copy of the session key, and that both the PLU and SLU have the ability to encipher and decipher data correctly. If cryptography is specified, a randomly chosen 64-bit number (N) enciphered under the session key (KS) is appended to the positive BIND response. (Note that by SNA protocol, the SLU is required to respond to the BIND command positively if in agreement, negatively if not.) Representing N as the concatenation of two 32-bit quantities, N1 and N2, the resulting quantity, $E_{KS}(N1, N2)$, is thus returned to the PLU in the BIND response.

For those sessions bound using cryptography, the PLU is required to initiate an additional command, the CRYPTO VERIFICATION (CRV) command. The CRV command is used to send the quantity $E_{KS}(\overline{N1}, N2)$ to the SLU, where $\overline{N1}$ denotes the complement of N1. The PLU produces this quantity

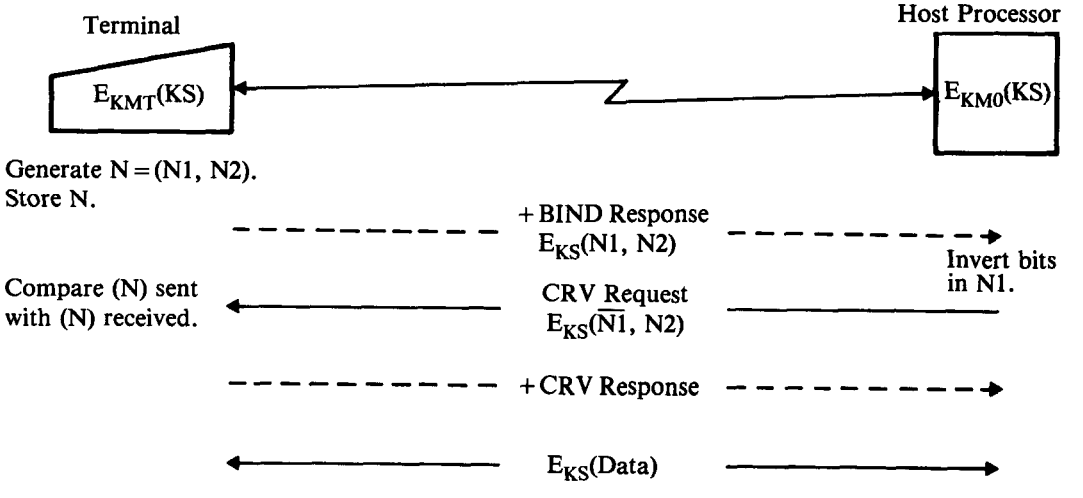


Figure 7-4. Cryptographic Verification Procedure for Session Keys

[2] In this and all subsequent figures, SNA commands are denoted by solid arrows (⟶) and responses by broken arrows (- - -▸).

by deciphering $E_{KS}(N1, N2)$, which it received from the SLU in the BIND response, inverting N1, and enciphering $(\overline{N1}, N2)$. The quantity $E_{KS}(\overline{N1}, N2)$ is then sent to the SLU (via CRV). Upon receipt of the CRV, the SLU deciphers $E_{KS}(\overline{N1}, N2)$, inverts $\overline{N1}$, and compares the result (N1, N2) with the random number (N) that it originally sent to the PLU in the BIND response. If equal, the SLU responds positively to the CRV (completing the session initiation procedure), and messages may now flow between the two bound LUs, within the constraints of SNA and subject to the agreed upon session cryptographic protocol. If unequal, the SLU overwrites the stored copy of the session key and responds negatively to the CRV. This action then causes the PLU to terminate the session.

Besides verifying that the SLU and PLU have identical session keys and that both LUs can correctly encipher and decipher data, the described handshaking procedure prevents an attack known as the midnight attack, where the data from an entire session is intercepted, recorded, and then later played back into the terminal. (For a more detailed analysis of handshaking, see Authentication Techniques Using Cryptography, Chapter 8.)

Furthermore, random number N is subsequently used by the LUs in session as the initial chaining value (ICV) required with block chaining (see Communication Security and File Security Using Cryptography, Chapter 4). Note that SNA specifies block chaining with ciphertext feedback as the default mode of data encryption.

Figure 7-5 illustrates the session concept in a single domain network where cryptography has not been implemented. Figures 7-6 and 7-7 illustrate session-level cryptography in a single-domain network where the cryptographic system operates in a transparent mode and employs system managed keys (KMTs and KSs).[3]

By employing system keys, the architecture discussed so far allows cryptography to be used in a manner that is transparent to both terminal users and application programs. If personal keys (KPs) are employed, cryptography can still be used in a manner that is transparent to application programs (by treating KP as a master key of a terminal), but requires an action on the part of the user to install the key. For example, KP could be stored on a magnetic stripe card and entered into the terminal during session initiation. A *write master key* operation would then allow KP to be loaded as the terminal's master key.

In this case, KP is used as a key-encrypting key, and session keys are thus sent to the terminal in the form $E_{KP}(KS)$, as shown in Figure 7-8. Except for the fact that KP is identified by the user's ID, whereas KMT was identified by the terminal's ID, the basic protocol has not changed.

[3] The following legend applies to this and all subsequent figures: GENKEY (Generate Key) produces a session key (KS) enciphered under the host master key (KM0) and transforms it to encipherment under the appropriate secondary communication key(s) (KMT, KNC). RETKEY (Retrieve Key) transforms a session key (KS) from encipherment under a secondary communication key (KNC) to encipherment under the host master key (KM0). CIPHER performs data enciphering and data deciphering on behalf of an application program residing in the host. CKDS (Cryptographic Key Data Set) denotes a table of enciphered keys.

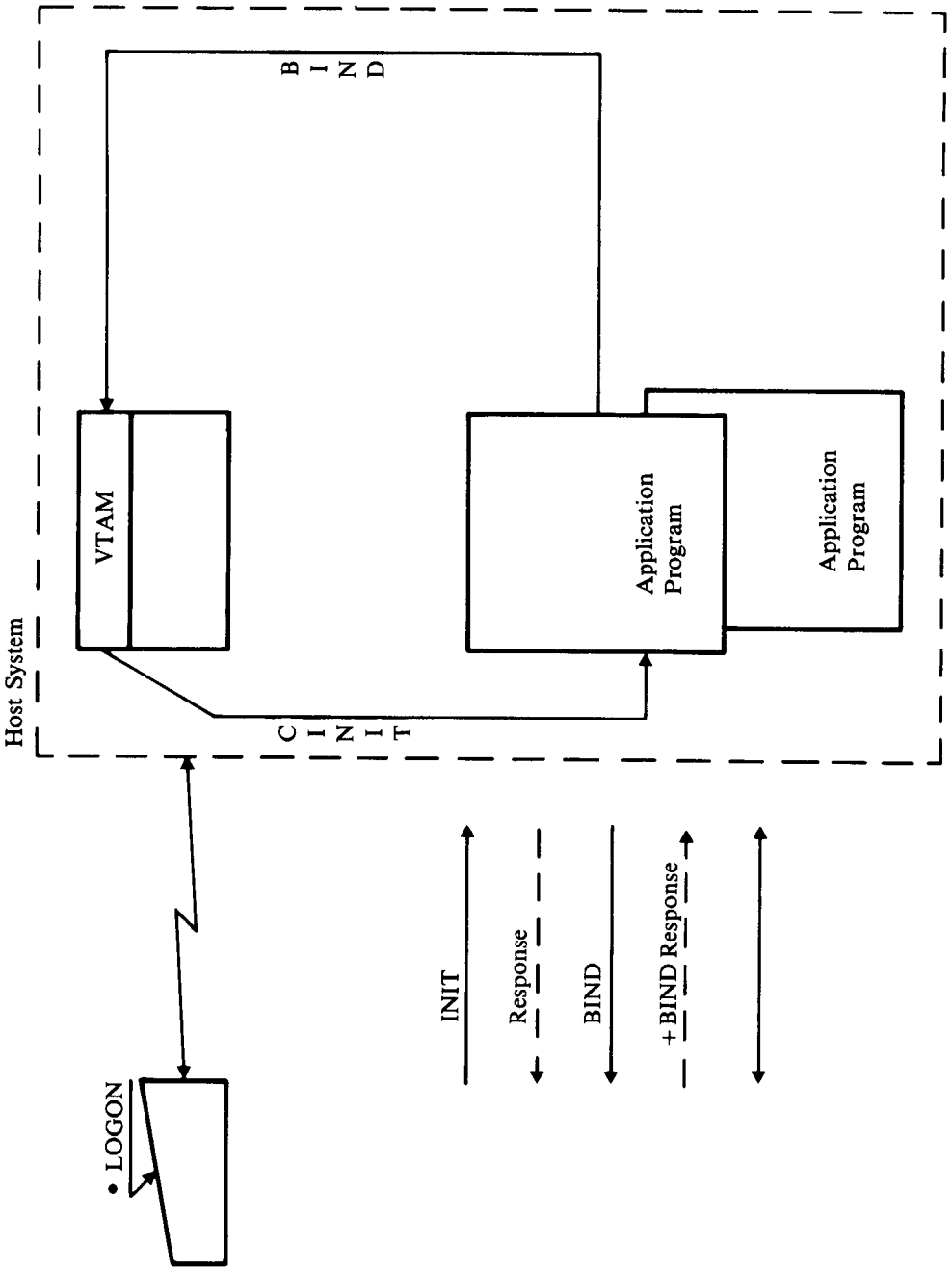


Figure 7-5. Session Concept in a Single-Domain Network (No Cryptography)

Note: ● denotes cryptographic facility in this and subsequent figures.

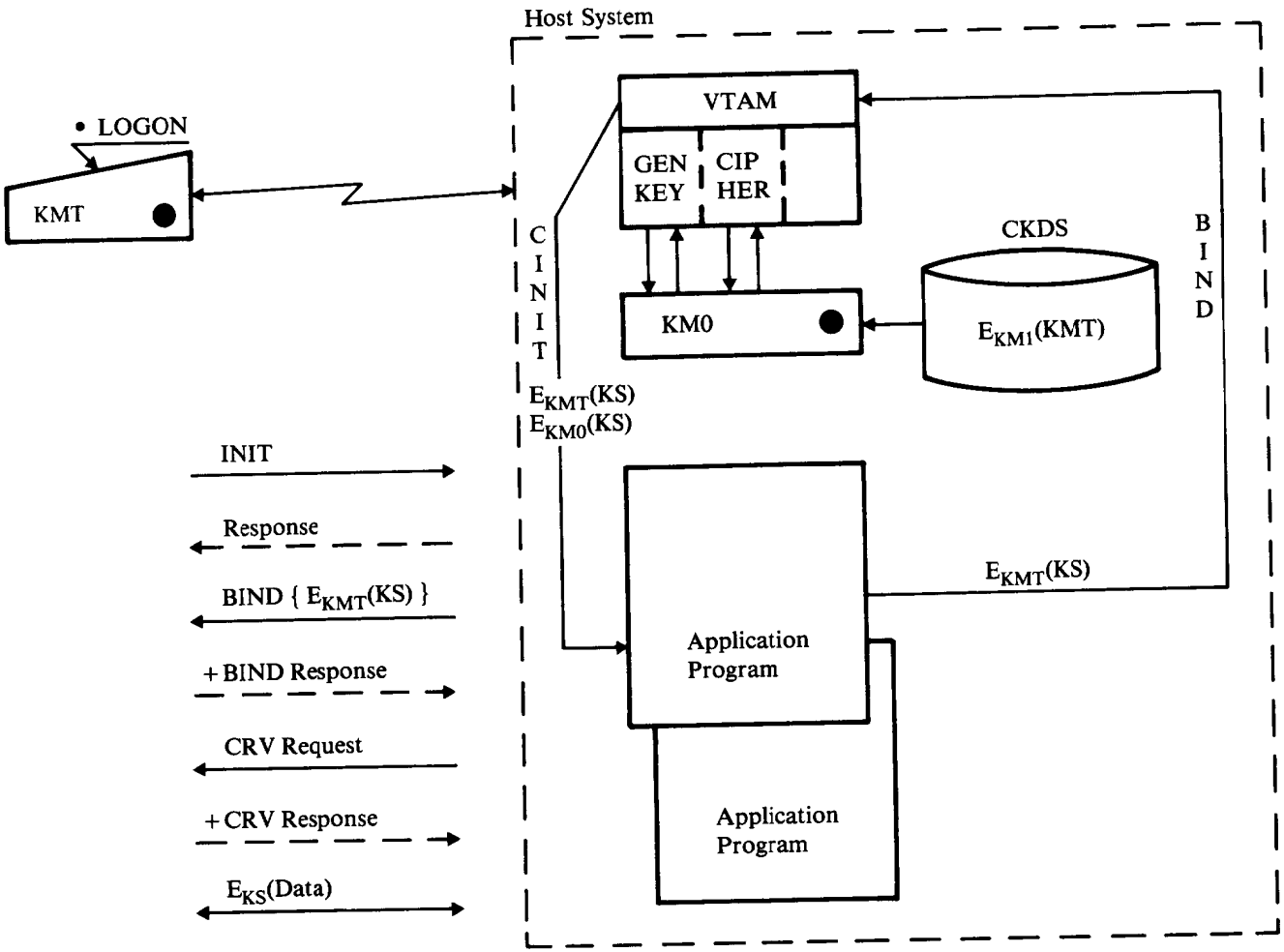


Figure 7-6. Session-Level Cryptography in a Single-Domain Network (Transparent Mode, Systems Keys, System Managed)

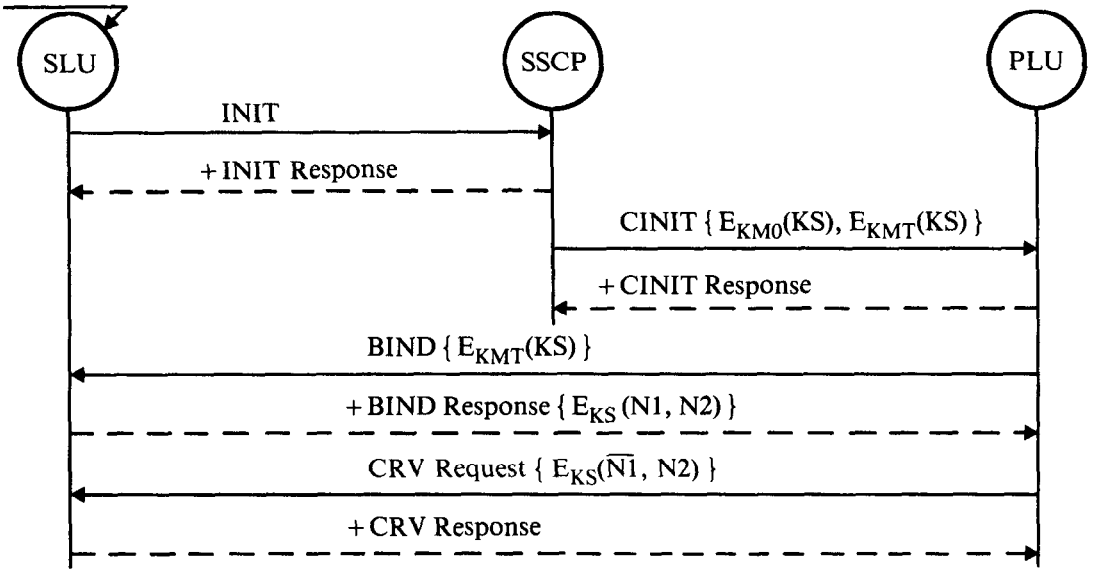


Figure 7-7. Session-Level Cryptography in a Single-Domain Network (SNA Command Flow, Transparent Mode)

Nontransparent Mode of Operation

In this mode, requests for cryptographic services originate with an application program instead of a system routine, and positive action is required on the part of the terminal user to load a personal key in the terminal (Figure 7-9). KP is entered in the terminal at session initiation, and a *write master key* operation is used to load it as the terminal's master key. Session keys are thus sent to the terminal in the form E_{KP}(KS).

A private protocol can be established between the terminal and application program by using the private bit in BIND (provided for this purpose by SNA). But to process the BIND properly, the terminal must be programmable (i.e., the user must be able to program for private cryptography).

In the described protocol, key distribution is accomplished with BIND, but there is no CRV issued. Keys are managed by the application program, not the system, and if handshaking is desired it must be provided by the private protocol.

PRIVATE CRYPTOGRAPHY IN A SINGLE-DOMAIN NETWORK

If a personal key (KP) is used to encipher and decipher data instead of a session key (KS), then *private cryptography* must be used rather than session-level cryptography. Although this mode of operation is not defined by SNA, neither is it precluded. What is needed is a means of synchronization similar to that provided by session-level cryptography (key selection, key generation, cryptography selection mechanism, etc.) that has been agreed to and imple-

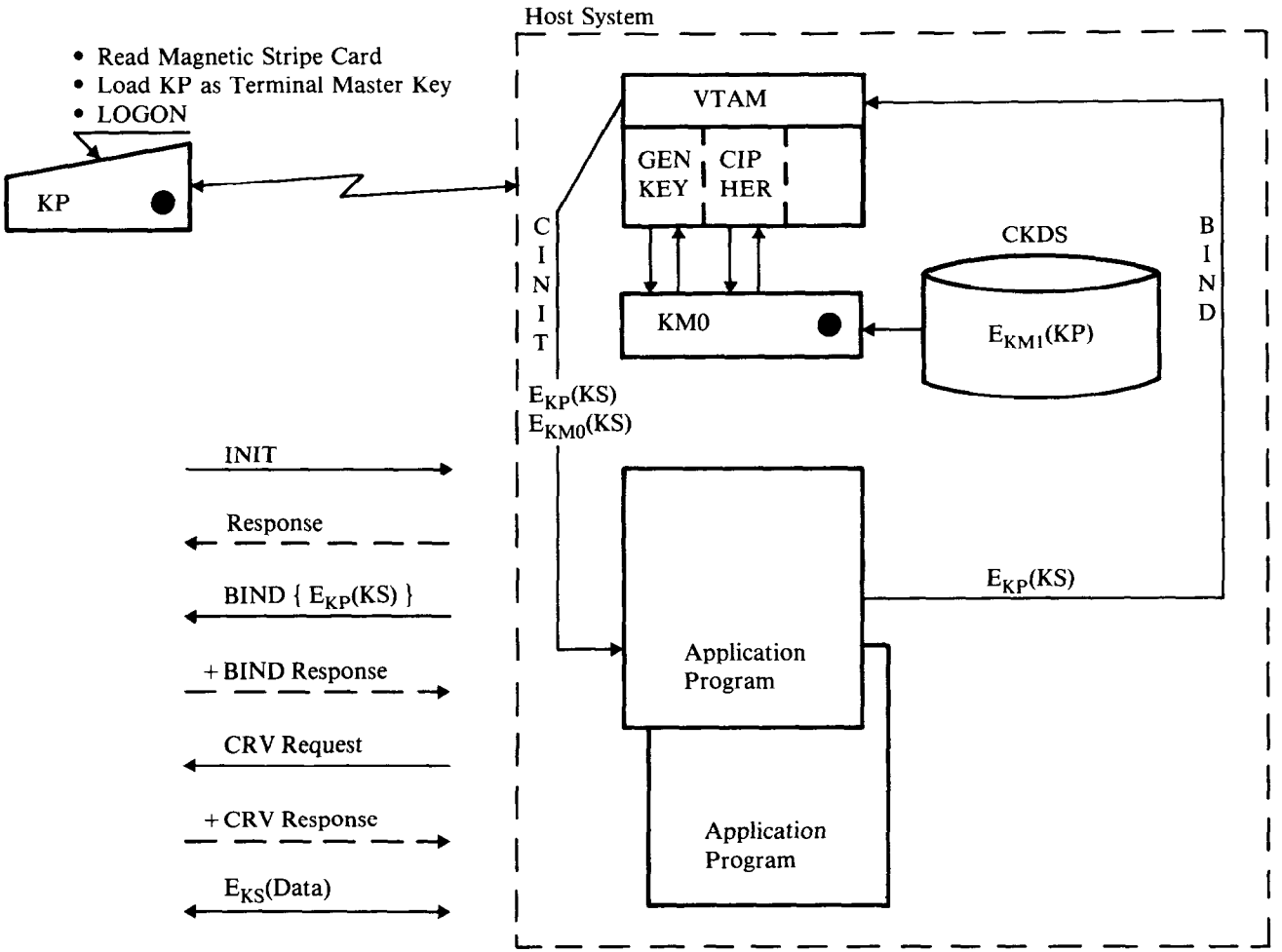


Figure 7-8. Session-Level Cryptography in a Single-Domain Network (Transparent Mode, Personal Keys, System Managed)

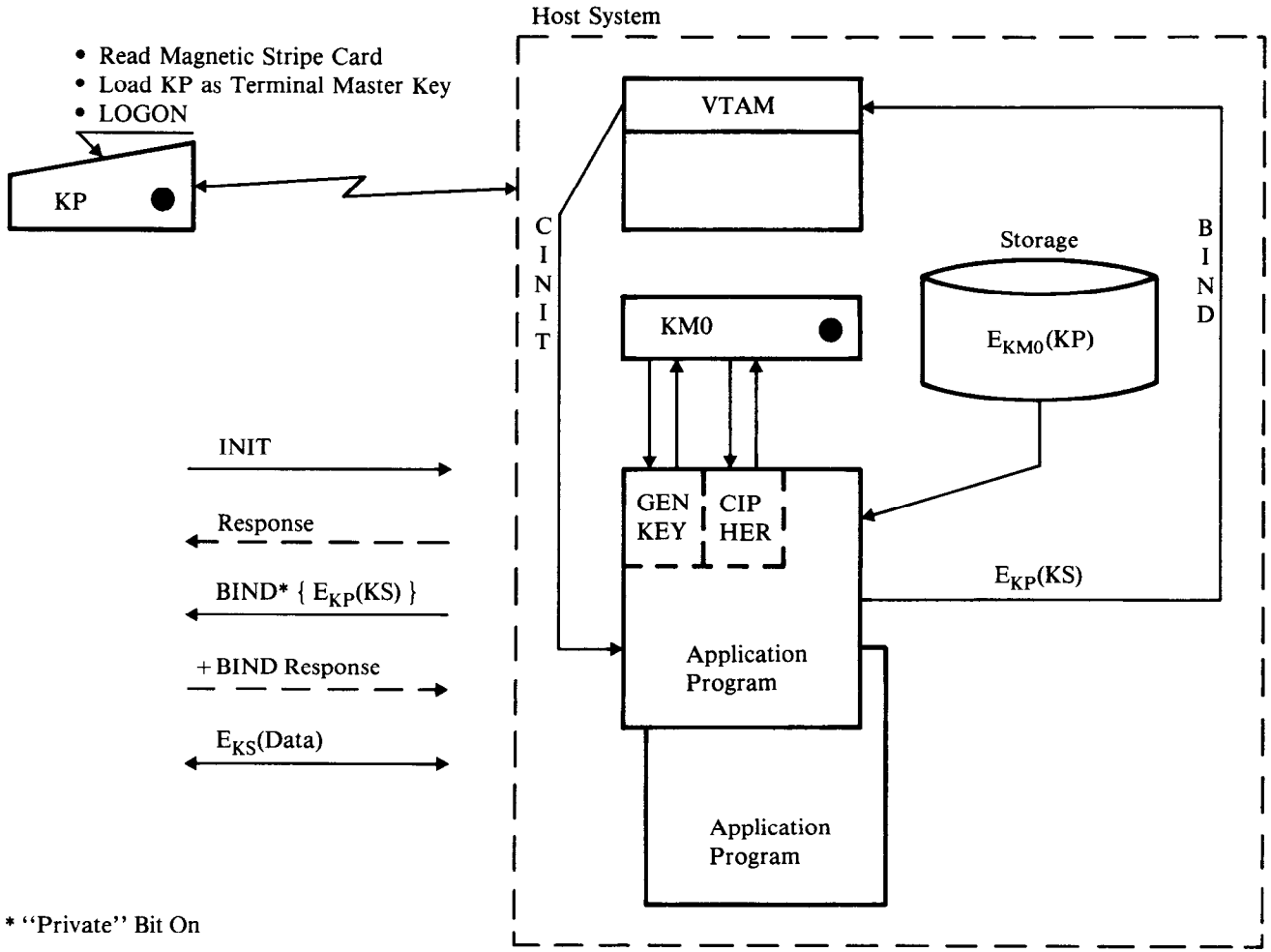


Figure 7-9. Session-Level Cryptography in a Single-Domain Network (Nontransparent Mode, Personal Keys, Privately Managed)

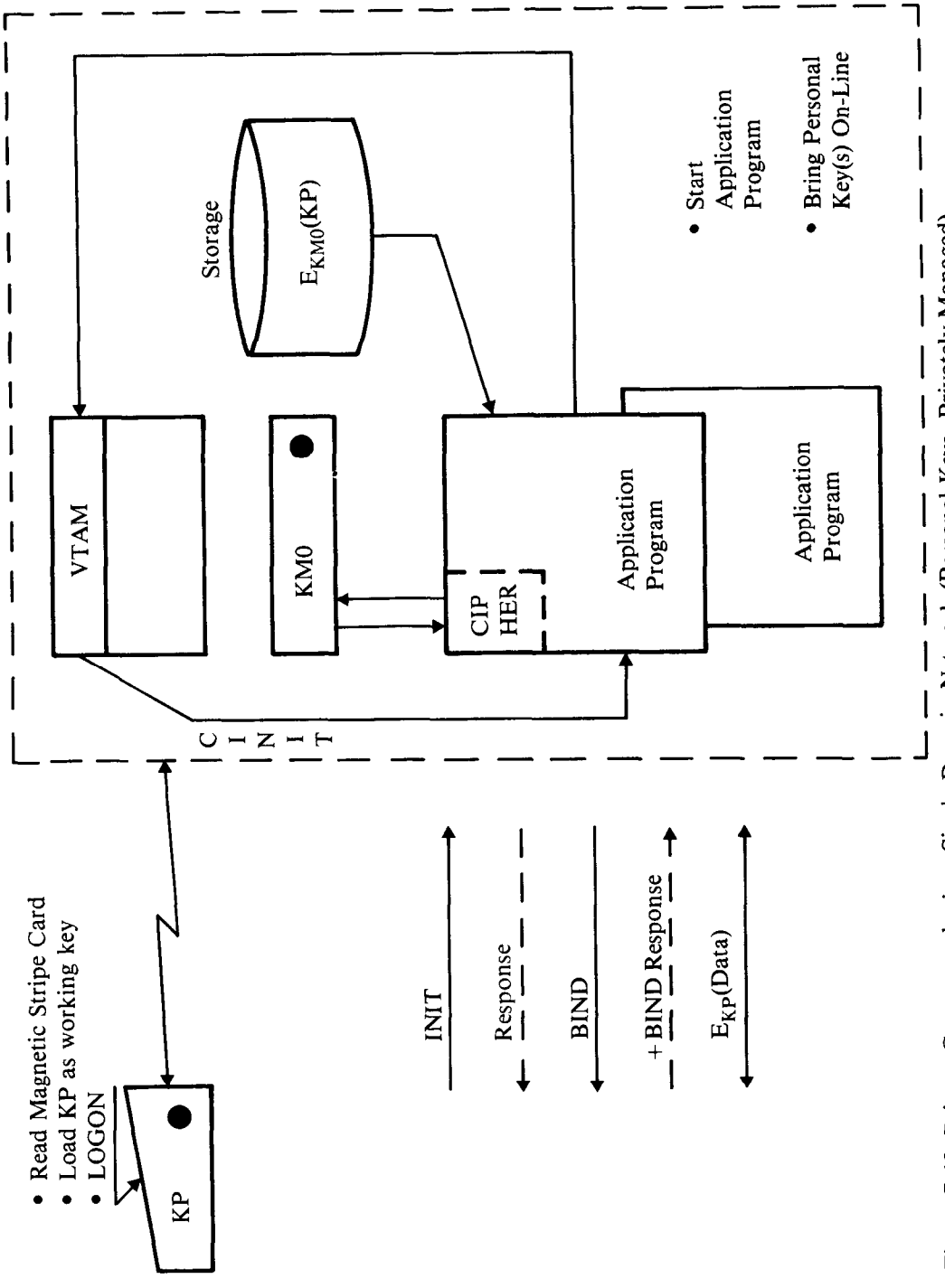


Figure 7-10. Private Cryptography in a Single-Domain Network (Personal Keys, Privately Managed)

mented by the communicants in advance (i.e., the nature of the *private protocol must be defined*).

With private cryptography (Figure 7-10), KP is entered in the terminal at session initiation, and a *load key direct* (LKD) operation is used to transfer it to the terminal's working key storage. KP can then be directly used to encipher and decipher data with the ENC and DEC operations.

At the host, KP is maintained enciphered under the host master key (KM0). In this form, the personal key can be directly used in an *encipher data* (ECPH) or *decipher data* (DCPH) operation. But because the DCPH operation is available to any authorized user of the system, anyone having *access* to KP can decipher data. Thus data security is enhanced if KP is stored off-line and entered as an input parameter only when the using application is executed.

Table 7-1 gives a brief overview of the possible implementations using system managed keys and personal keys.

Approach	Secondary Communication Key	Data-Encrypting Key	Storage of Key at Host
Fig. 7-6	Terminal Master Key (KMT)	Session Key (KS)	E_{KM1}(KMT) Stored On-Line
Fig. 7-8	Personal Key (KP)	Session Key (KS)	E_{KM1}(KP) Stored On-Line
Fig. 7-9	Personal Key (KP)	Session Key (KS)	E_{KM1}(KP) Stored On-Line
Fig. 7-10	None	Personal Key (KP)	E_{KM0}(KP) Typically Stored Off-Line

Table 7-1. Summary of Approaches Using System Managed Keys and Personal Keys

SESSION-LEVEL CRYPTOGRAPHY IN A MULTIDOMAIN NETWORK

The addition of another domain managed by another host requires additional commands to flow in the session initiation process. Figure 7-11 describes the multidomain case. Note that the difference between the multidomain case and the single-domain case is the addition of a cross-domain link. It is over this link that SNA supports a special session known as the *cross-domain session*. The cross-domain session plays an important role in establishing session-level cryptography between two LUs residing in different domains.

As in the single-domain case, the process might begin with the receipt of an INIT command at the SSCP logically owning the SLU. Again, this is typi-

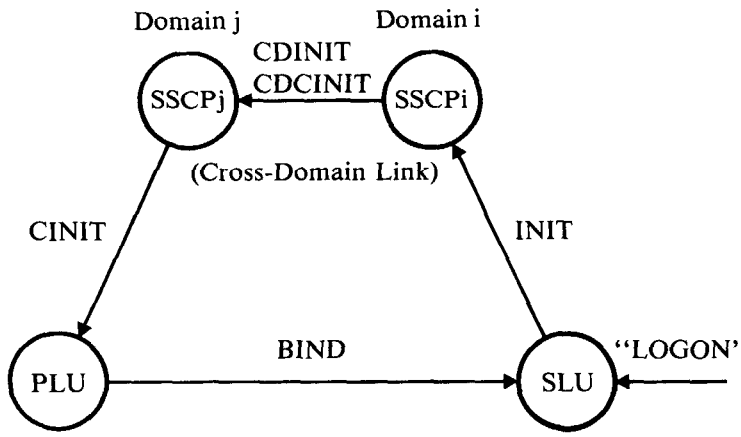


Figure 7-11. SNA Session Initiation Command Flow in a Multi-Domain Network

cally the result of a LOGON sequence entered at a terminal. The PLU (via its SSCP) may also request a session with an SLU. If the SSCP receiving the INIT determines that the requested LU is not in the immediate domain, it initiates cross-domain communication with the appropriate owning SSCP.

The CROSS-DOMAIN INITIATE (CDINIT) command is sent between SSCPs to indicate that an LU in the sender's domain wishes to establish a session with an LU in the receiver's domain. Transmission of CDINIT and its response allows each SSCP to define completely the communicating LUs. The protocol permits session requests to originate in the domain of either LU. No extension to CDINIT is required for cryptography.

A positive acknowledgment to CDINIT results in the creation and transmittal of the CROSS-DOMAIN CONTROL INITIATE (CDCINIT) command. It is through this command that the session key, enciphered under a secondary communication key (KNC), is passed from one domain to another. (SNA defines this key as a *cross-domain key*). In addition to enciphering KS under the cross-domain key, the SSCP must also encipher KS under the SLU's key. This is because the SLU's key is known only to its owning SSCP, and the SSCP therefore is responsible for managing session key initialization. The SLU's key is either a terminal master key (KMT), if the SLU is a terminal, or a special secondary communication key called a *node application key* (KNA), if the SLU is a host application program. (For the distinction between KMT and KNA, see Application Program-to-Application Program Cryptography.) In either case, the session key (KS) is enciphered under the SLU's key and then placed in the BIND image. Both KS (enciphered under the cross-domain key) and the BIND image are included in the CDCINIT command.

The receiving SSCP extracts from CDCINIT the value of KS which was previously enciphered under the cross-domain key. KS is then reenciphered via a RETKEY macro instruction (see Key Management Macro Instructions, Chapter 4) under the host master key of the SSCP, and the result is placed

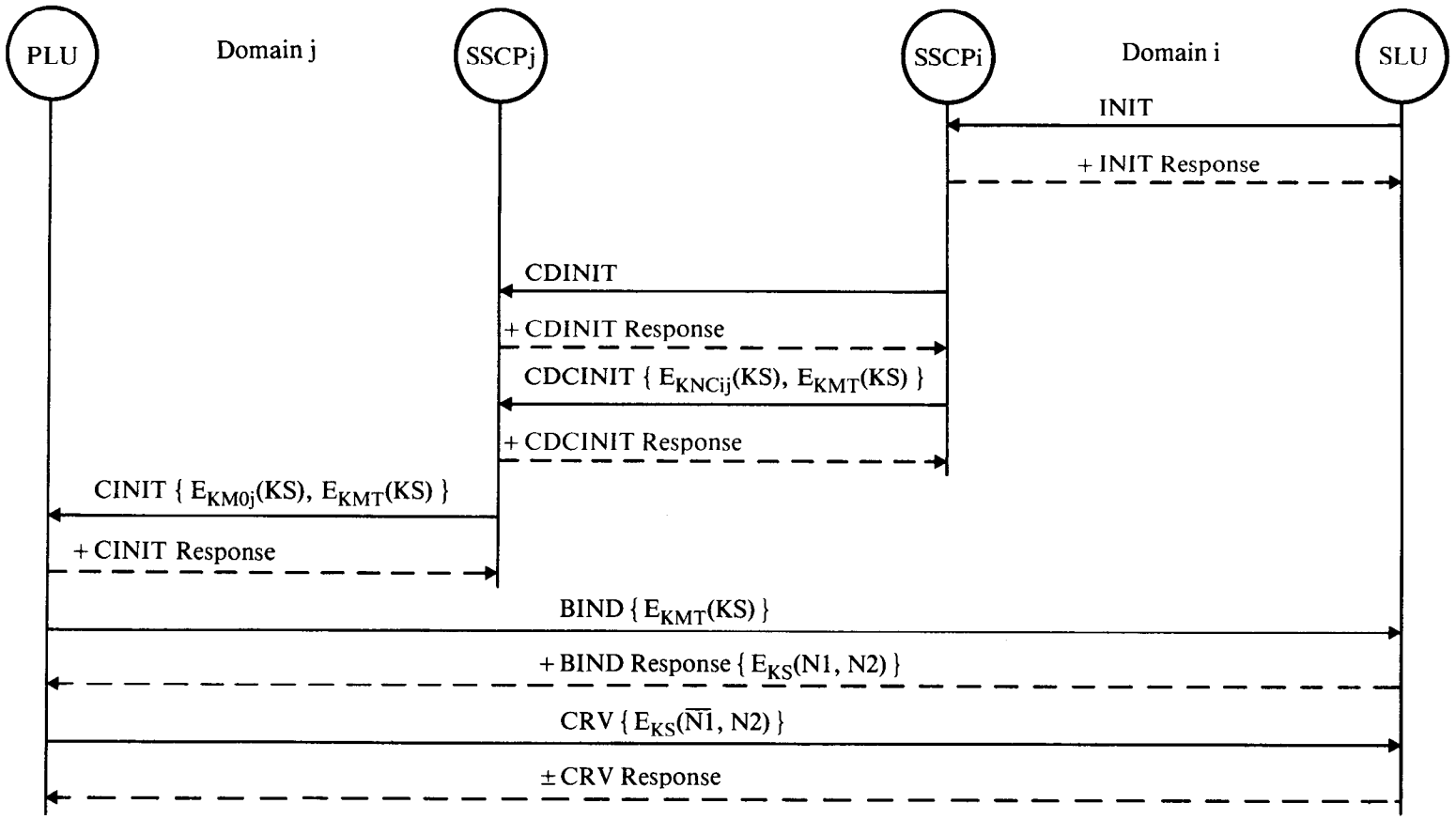
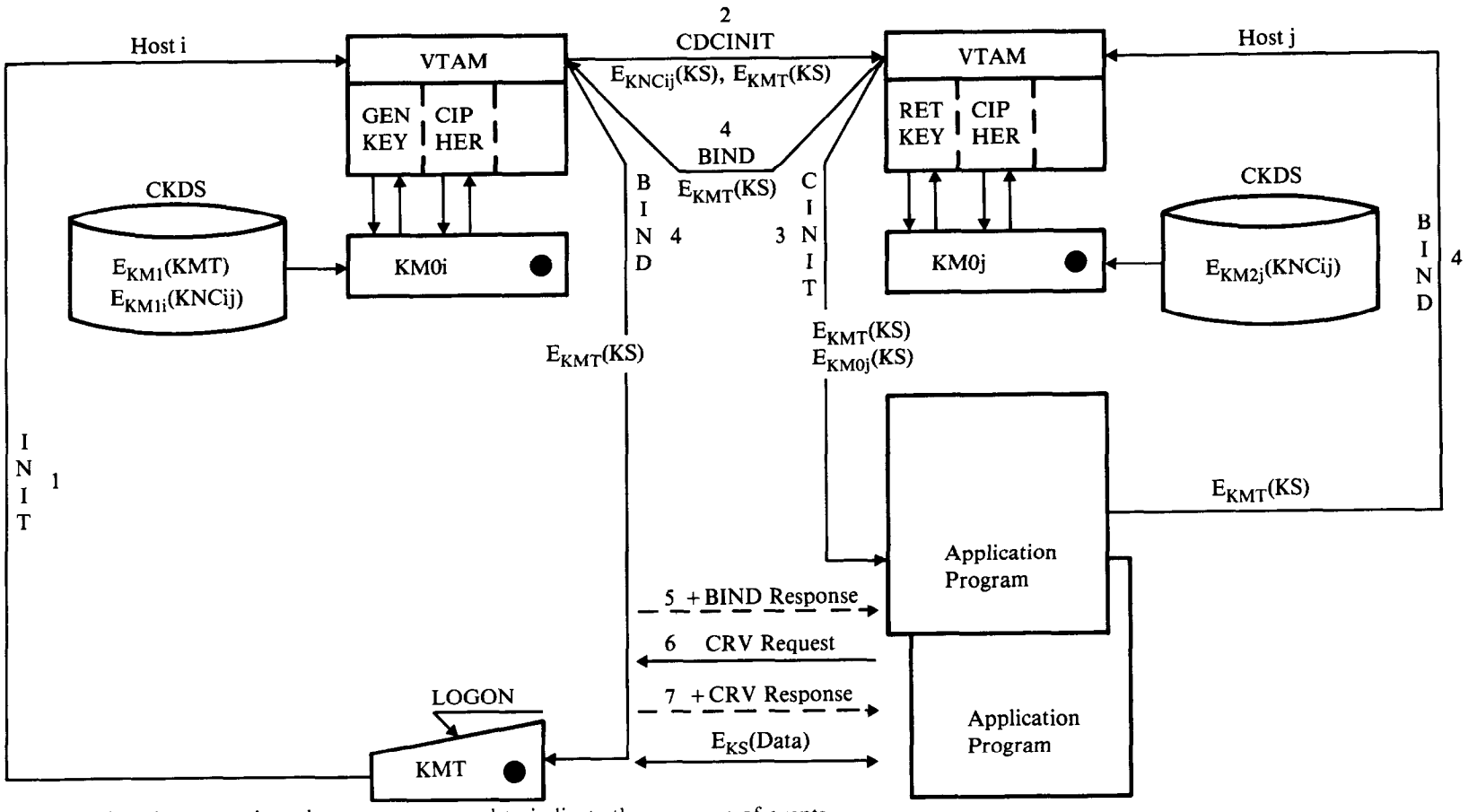


Figure 7-12. Session-Level Cryptography in a Multi-Domain Network (SNA Command Flow, Transparent Mode)



Note: numbered commands and responses are used to indicate the sequence of events.
Some responses are not shown to avoid unnecessary complexity.

Figure 7-13. Session-Level Cryptography in a Multi-Domain Network (Transparent Mode, System Keys, System Managed)

in the appropriate field of a CONTROL INITIATE (CINIT) command. The BIND image, which contains KS enciphered under the SLU's key, is also copied to CINIT, and CINIT is then passed to the requested PLU (application program).

From this point, the action taken by the PLU and SLU is the same as discussed earlier (see Session-Level Cryptography in a Single-Domain Network.) To summarize, the receiving PLU extracts from CINIT the value of KS which was previously enciphered under the host's master key, and stores it for later use during the session. It also extracts the BIND image, which contains KS enciphered under the SLU's key, and transmits the enciphered session key to the SLU via BIND. At the end of this exchange, both the PLU and SLU have identical copies of the session key.

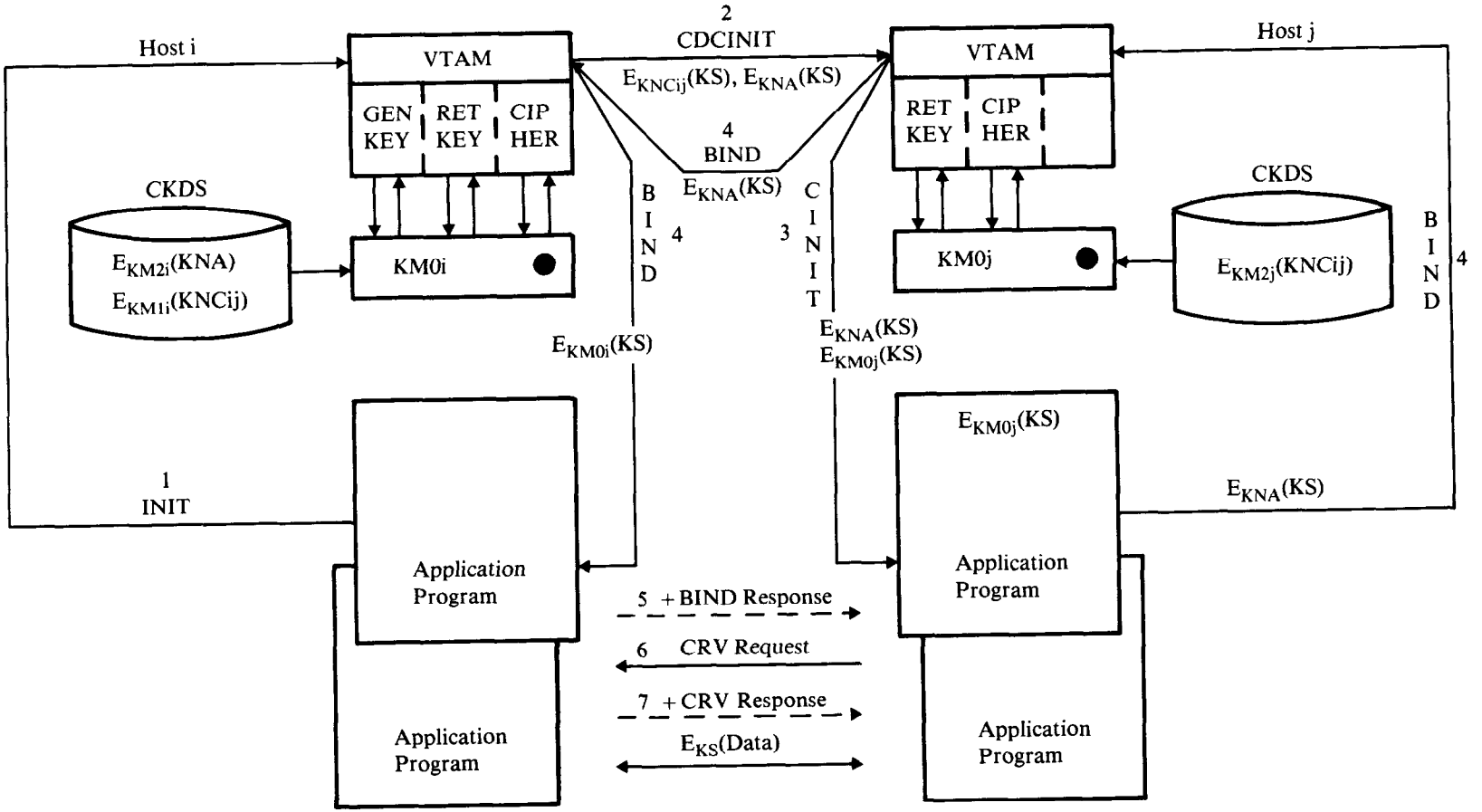
Figures 7-12 and 7-13 provide an overview of the command flow and keys contained therein for the establishment of a cryptographic session between two LUs in different domains.

APPLICATION PROGRAM-TO-APPLICATION PROGRAM CRYPTOGRAPHY

Unlike a terminal, an application program has no cryptographic facility of its own; it must use the host's cryptographic facility. Thus to perform encipher and decipher operations, the session key used by an application program must be enciphered under the host master key, that is in the form $E_{KM0}(KS)$. Because of this, the nature of the key transformations that support application-to-application communications vary slightly from those described for application to terminal communications. (In SNA terms, an application program is referred to as an outboard LU.)

Suppose that an SLU residing in host i has requested a session with a PLU in host j. The session key to be used by the SLU thus appears as $E_{KM0i}(KS)$, and is analogous to an outboard SLU's session key which appears in the form $E_{KMT}(KS)$. The SNA protocol for establishing a common session key requires that the key be received by the SLU in BIND. In other words, it must be sent to the SLU by the PLU. In the case of an outboard SLU, the key is directly usable in the form $E_{KMT}(KS)$. However, in the case of an inboard SLU, the equivalent approach of routing $E_{KM0i}(KS)$ to the SLU in domain i via the PLU in domain j is not desirable from a security viewpoint. A session key in the form $E_{KM0i}(KS)$ could be used directly in a *decipher data* operation at host i. Thus if an opponent were able to obtain $E_{KM0i}(KS)$ and data enciphered under KS (via an external wiretap), and obtain subsequent access to host i, then the data could be recovered. The problem is overcome, however, by enciphering the session key under an SLU key (KNA) associated with the specific application program. The quantity $E_{KNA}(KS)$, instead of $E_{KM0i}(KS)$, is sent to the PLU and returned via BIND to the host owning the SLU. After being received, the quantity $E_{KNA}(KS)$ is transformed via a RETKEY macro instruction[4] to the quantity $E_{KM0i}(KS)$, which can then be

[4] To ensure that the transformation cannot be used indiscriminately, the key manager could enforce a requirement that the program invoking RETKEY be privileged (see Protection of Host Keys, Chapter 4).



Note: some responses are not shown to avoid unnecessary complexity.

Figure 7-14. Session-Level Cryptography in a Multi-Domain Network (Application Program-to-Application Program Cryptography)

used by the application program at host i to encipher and decipher data. Application-to-application cryptography is illustrated in Figure 7-14.

Personal keys, whether managed privately or by the system, can be implemented in multidomain networks in a manner similar to that described for single-domain networks. The interested reader should have no difficulty in making the necessary extensions.

PADDING CONSIDERATIONS

When the DES algorithm is used in a block cipher mode of operation (e.g., block chaining with ciphertext feedback), a requirement for padding arises because communicated data quite often are not a multiple of 8 bytes (8 bits per byte). (The DES algorithm operates only with blocks of 8 bytes). Therefore, the communications architecture must accommodate a requirement to pad data to 8 bytes.

The cryptographic extensions to SNA provide for message padding. When padding is used, the last message block contains n data bytes (n = 1, 2, . . . , 7), 7-n random pad bytes, and a 1 byte count indicating the number of added (nondata) bytes. A defined bit in the accompanying RH is then set to indicate to the receiver that the message must be stripped of pad characters after decipherment. However, RH is not available to application programs. Thus a private protocol that pads messages must provide a means to identify when padding is used, and identify the number of added pad bytes.

REFERENCES

1. Lennon, R. E., "Cryptography Architecture for Information Security," *IBM Systems Journal,* **17**, No. 2, 138–150 (1978).
2. McFadyen, J. H., "Systems Network Architecture: An Overview," *IBM Systems Journal,* **15**, No. 1, 4–23 (1976).
3. Cypser, R. J., *Communications Architecture for Distributed Systems,* Addison-Wesley, Reading, MA, 1978.
4. Schwartz, M., *Computer-Communication Network Design and Analysis,* Prentice-Hall, Englewood Cliffs, NJ, 1977.
5. Davies, D. W. and Barber, D. L. A., *Communications Networks for Computers,* Wiley, New York, 1973.
6. *Proceedings of the IEEE,* Special Issue on Packet Communication Networks, **66**, No. 11, 1301–1588 (November 1978).
7. Albrecht, H. R. and Ryder, K. D., "The Virtual Telecommunications Access Method: A Systems Network Architecture Perspective," *IBM Systems Journal,* **15**, No. 1, 53-80 (1976).
8. *Advanced Communication Function for VTAM (ACF/VTAM) General Information,* IBM Systems Library, Form No. GC27–0462.

Other Publications of Interest

9. Barber, D. L. A., Davies, D. W., Price, W. L., and Solomonides, C. M., *Computer Networks and Their Protocols,* Wiley, New York, 1979.
10. Kent, S. T., *Protecting Externally Supplied Software in Small Computers,* Doctoral Thesis, Massachusetts Institute of Technology, 1980. Department of Electrical Engineering and Computer Science.