CHAPTER EIGHT

Authentication Techniques Using Cryptography

FUNDAMENTAL CONCEPTS

Authentication is a process which proves that someone or something is valid or genuine. In a computer system or communications network, authentication is an important part of good data security. Cryptography offers a highly secure means to authenticate transmitted messages, stored data, and the identity of people and devices.

Generally, all authentication schemes have a common step in which the validity of one or more parameters must be checked. An authentication scheme is characterized by the nature of the preestablished relationships existing between the checked parameters and the quantities to be authenticated.

For example, authentication of a person's identity requires a special test of legitimacy in which a secret or nonforgeable parameter is supplied by the identified individual together with a claimed identity (ID). By checking the validity of the supplied parameter, the system can decide whether the identified individual is the person he claimed to be. (Personal authentication is discussed in greater detail in Chapters 10 and 11, which treat electronic funds transfer systems.)

Data parameters that remain constant for relatively long periods can be authenticated using bit patterns which are pregenerated under secure conditions. Data parameters which are not constant must be authenticated using bit patterns generated dynamically within the system during normal operations.

The authentication techniques discussed in this chapter are based on a conventional algorithm like DES, and it is assumed that:

1. The cryptographic algorithm is strong, which implies a property of noninvertibility.
2. The cryptographic system can maintain the secrecy and/or integrity of its cryptographic keys.

However, many of the concepts and principles apply to public-key algorithms as well.

As previously defined, encipherment of plaintext X under cipher key K,

results in ciphertext $Y = E_K(X)$, and decipherment of ciphertext Y under cipher key K results in plaintext $X = D_K(Y)$.)

A cryptographic algorithm has the *property of noninvertibility* if it is computationally infeasible to determine cipher key K given knowledge of plaintext X and ciphertext $E_K(X)$. Many cryptographic authentication techniques take advantage of this property by treating certain data as cryptographic keys. The idea is similar to that of using one-way functions in the process of personal identification [1–3], in which case the function's input parameter(s) cannot be reconstructed from the function's output parameter.

HANDSHAKING

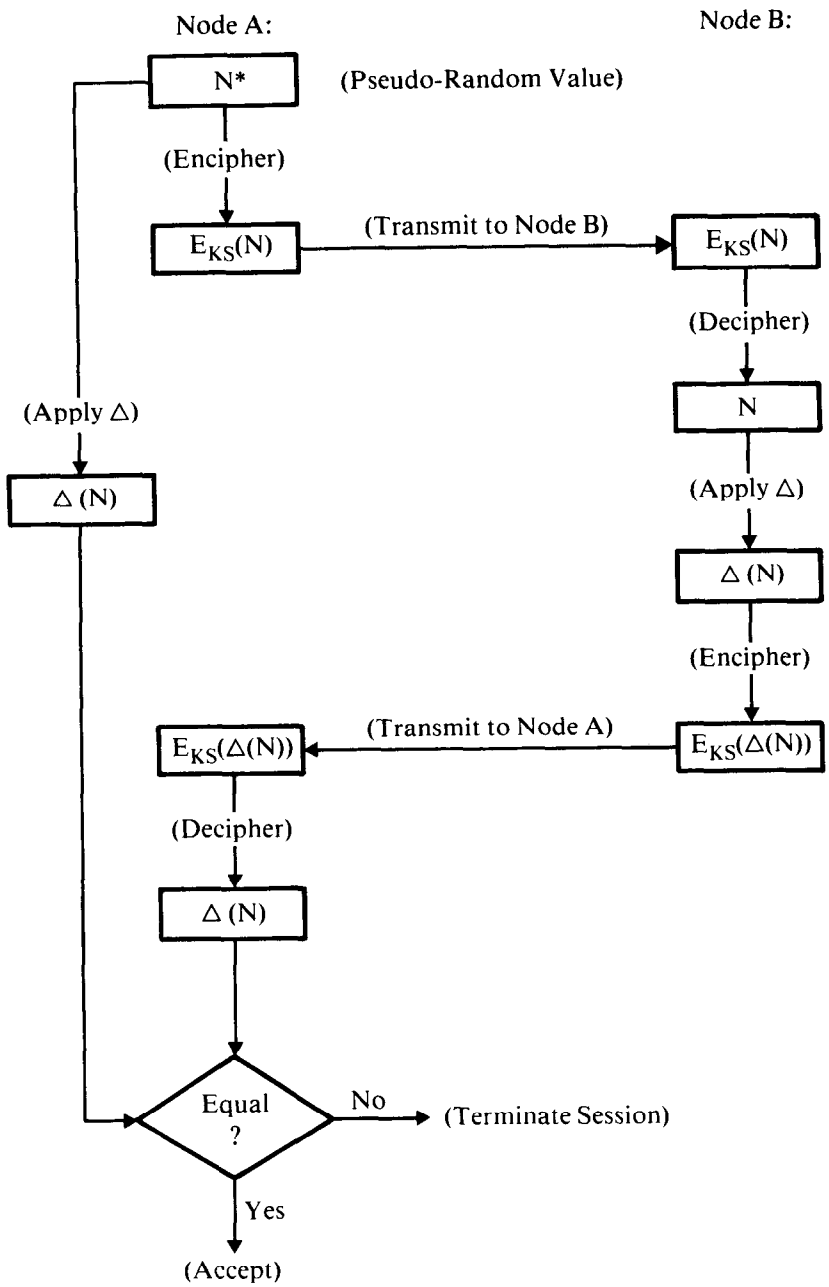
Handshaking is a procedure to ensure that communication has been established between two genuine nodes (devices) within a communications network. Ordinarily used when communications are initiated, it permits a node to prove that its communicating partner possesses the same data-encrypting key. This is accomplished by testing that enciphered data can be communicated successfully between the two nodes.

In the key management scheme described in Chapter 4, a common session key, KS, is established between a host and terminal by generating KS at the host and transmitting it to the terminal. In this case, KS is protected by enciphering it under the terminal's master key, KMT, which is resident within the terminal's cryptographic facility.

Even though the secrecy of KS is ensured by enciphering it with KMT, without handshaking the procedure is exposed to the so-called midnight attack. In a midnight attack launched against a terminal, the data associated with an entire communication session, including the session key, which is sent to the terminal under KMT encipherment, are wiretapped and recorded (figuratively, by day). Later, (figuratively, by night), the opponent gains unauthorized access to the terminal room and plays back the recording into the terminal. During playback, the terminal is unaware that it is in communication with a fictitious node. The session key is deciphered and initialized in the cryptographic facility. The ciphertext is then deciphered and presented to the opponent who is stationed at the terminal. Handshaking prevents the midnight attack.[1]

Figure 8-1 illustrates a procedure that allows node A (the terminal) to authenticate node B (the host). (By reversing the procedure, node B could also authenticate node A.) Assume that node A generates a pseudo-random number N which it enciphers under the session key KS and transmits to node B. N is padded with null characters if necessary, although it is assumed that N is long enough to ensure that the probability of the same value recurring is very low. N is recovered at node B by deciphering the transmission under

[1] Other cryptographic methods could be used to defend against a midnight attack; for example, a protocol of composite session keys (see Extended Cryptographic Operations, Chapter 5). A midnight attack would also be blocked if (user-supplied) personal keys were used at terminals instead of (system-managed) terminal master keys.



Note: Δ could be a function which inverts certain bits in N.

*Padding with zeros is indicated for the case where N does not have enough bits. N must have enough combinations to prevent a successful attack by trial and error.

Figure 8-1. Authentication of System Nodes via Handshaking

KS. Note that this step is possible only if node B is a genuine node and a copy of the correct KS is available. A nonsecret function Δ is applied to N, and the quantity $\Delta(N)$ is enciphered under KS and sent back to node A. At node A, Δ is similarly applied to the stored or saved value of N so that $\Delta(N)$ can be compared with the corresponding value returned from node B. Node B is accepted as genuine only if the comparison is successful. Otherwise, the session is aborted.

The following example illustrates why an opponent is blocked from carrying out a midnight attack when handshaking is used. Let N1 be a pseudo-random number generated at node A during a session that is wiretapped by an opponent. Assume that node A is a remote terminal. Later, when the recording is played back into the terminal, a value N2 (not equal to N1) is generated within the terminal as part of the handshaking procedure. This means that the prior value $E_{KS}(\Delta(N1))$ returned from node B is no longer correct, and hence the comparison for equality between $\Delta(N1)$ and $\Delta(N2)$ will fail. In an actual implementation, Δ could be a function that inverts certain bits in N.

It is important that N be generated within the terminal and stored in a protected memory. If N is entered by a human operator, or if the stored value of N can be overwritten by a value of the user's choosing, the procedure can be attacked. For example, during an attempted playback, the opponent stationed at the terminal enters $\Delta(N2)$ instead of N2 (Δ is a nonsecret function) and wiretaps the quantity $E_{KS}(\Delta(N2))$, which is sent to node B instead of $E_{KS}(N2)$. The original recording is now modified so that $E_{KS}(\Delta(N1))$ is replaced by $E_{KS}(\Delta(N2))$. A second playback is made. This time the opponent enters N2 into the terminal and the authentication check therefore succeeds.

To block this kind of attack, N can be defined as an operator-entered number RN which is enciphered under the session key KS:

$$N \equiv E_{KS}(RN)$$

Since N is a function of the unknown session key KS, there is no way to subvert the checking procedure unless it is possible to encipher or decipher under the same session key. For example, if an opponent could gain access to an unattended terminal during an actual session, then N2 could be entered as data and the ciphertext $E_{KS}(N2)$ intercepted via a wiretap. Since Δ is a nonsecret function, $\Delta(E_{KS}(N2))$ could be computed from $E_{KS}(N2)$. Then, by entering $\Delta(E_{KS}(N2))$ as data during the same session, the quantity $E_{KS}(\Delta(E_{KS}(N2)))$ could be obtained via a wiretap. Thus to defeat the system, the opponent replaces $E_{KS}(\Delta(E_{KS}(N1)))$ with $E_{KS}(\Delta(E_{KS}(N2)))$ in the original recording, and enters N2 at the terminal.

In an actual implementation, this attack may already be blocked. For example, the attack would not be possible if certain characters in $\Delta(E_{KS}(N2))$ had no corresponding keyboard characters.

Whenever possible, the pseudo-random number N should be generated within a terminal. In theory, this requires a nonvolatile storage in the terminal; otherwise, the terminal will always start from the same initial conditions when power is first supplied to it. However, from a practical viewpoint,

a free-running counter starting from the same value (after power-on) may be acceptable provided that the resolution of the counter is such that synchronizing the recorded playback with the power-on sequence is impractical.

MESSAGE AUTHENTICATION

Message authentication is a procedure which, when established between two communicants, allows each communicant to verify that received messages are genuine. Communicants can be people, devices, or processing functions. Typically, a communicant acts as both a sender and receiver of messages, although it is possible for a communicant to participate only as a sender or receiver. Message authentication must allow the receiver of a message to determine that:

1. The message originated with the alleged sender.
2. The contents of the message have not been accidentally or intentionally changed.
3. The message has been received in the same sequence that it was transmitted.
4. The message was delivered to the intended receiver.

In other words, message authentication permits the receiver to validate a message's *origin, contents, timeliness,* and *intended destination.*[2]

Although message authentication permits the receiver to verify that messages are genuine, it does not always permit these properties (origin, contents, timeliness, and intended destination) to be proven to or verified by a third party. (If both the sender and receiver share the same secret information used in the authentication procedure, the sender could later claim that a message was manufactured by the receiver.)

However, an approach using digital signatures does permit the receiver to prove to a third party that messages are genuine (see Digital Signatures, Chapter 9). It provides that the sender cannot later disavow messages as his own, the receiver is unable to forge messages or signatures, and the receiver can prove to an impartial third party (referee) that the content of a message is genuine and that it originated with the sender.

Authentication of a Message's Origin

Two methods for a receiver to verify a message's origin (sender) are discussed. In the first method, the sender's identity is verified through the use of a

[2] Proposed Federal Standard 1026 [4] defines authentication as "the process of assuring that only intended parties or locations exchange and accept a given unit of information as being valid." The standard describes cryptographic methods to achieve the following security objectives: detection of fraudulent insertion of messages; detection of fraudulent deletion of messages; detection of fraudulent modification of messages; and detection of replay of previously valid messages. These security objectives, therefore, represent a subset of those specified herein.

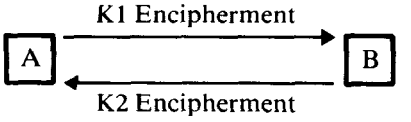


Figure 8-2. Message Encipherment Between A and B

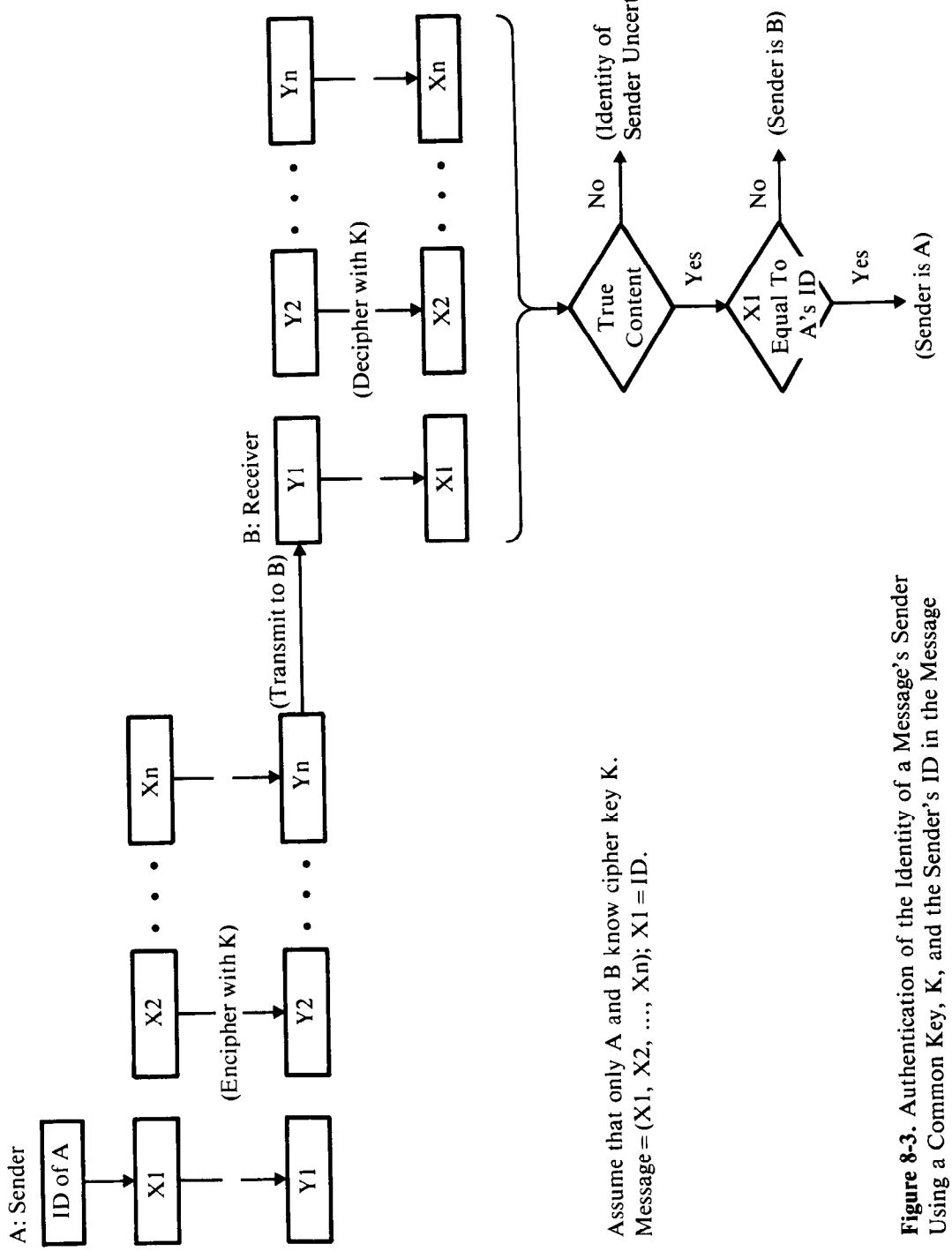
secret data-encrypting key shared by the sender and receiver. The receiver verifies the sender's identity by deciphering the received ciphertext using the agreed upon key (determined by the sender's ID), and checking that the text has been properly recovered. In the second method, the sender's identity is verified through the use of a secret password (known to both the sender and receiver) that is included within the transmitted message.

Suppose that communicants A and B use cryptographic keys K1 and K2, where K1 is used only for transmission from A to B and K2 is used only for transmission from B to A (Figure 8-2). To prove that a message originated with A, B need only demonstrate that the message is properly recovered using K1. Similarly, to prove that a message originated with B, A need only demonstrate that the message is properly recovered using K2.

If the communicants A and B use a common key K there will always be some uncertainty as to which of the two (A or B) was the true originator of a message. For example, it may be possible for a stale message sent from A to B to be injected back into the communication path and redirected to A. Unless steps are taken to prevent such a condition, A could never be certain that a received message originated with B. This problem can be avoided if the sender includes an identifier (ID) in all transmitted messages and the receiver authenticates the contents of the message (Figure 8-3).

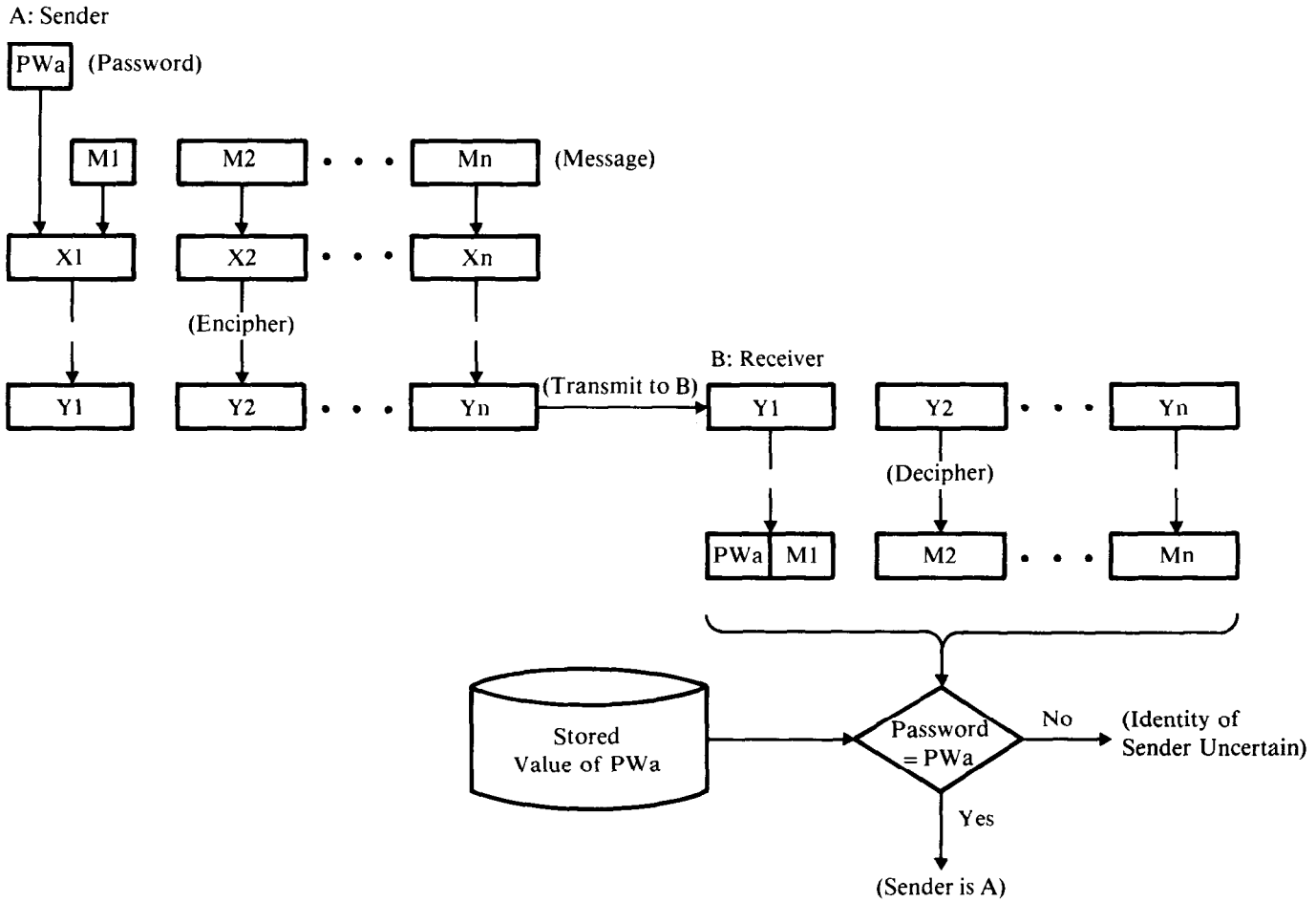
When it is either impractical or undesirable to verify a message's sender on the basis of a cryptographic key, a password can be used instead. In that case, A and B must agree ahead of time on the passwords to be used. Let PWa and PWb be the respective passwords assigned to A and B. A includes PWa in all messages sent to B, and B includes PWb in all messages sent to A. To prevent an opponent from finding out the value of one of these passwords, which could then be used to impersonate the respective communicant, passwords are encrypted. But encryption alone is not enough, since an opponent may be able to impersonate a communicant by substituting the communicant's encrypted password in a bogus message. The cryptographic procedure must be such that if an encrypted password is used in a bogus message, it causes either the recovered password to be incorrect or some other part of the message authentication procedure to fail. Various techniques can be used. For example, the password could be encrypted with a dynamically changing data-encrypting key, or block chain encryption could be used with a variable initializing vector (see Block Ciphers with Chaining, Chapter 2). The use of constant passwords is illustrated in Figure 8-4. (The assumption is made here that the number of password combinations is large enough so that it is impractical or impossible for an opponent to cause a correct password to be produced via some process of trial and error.)

In a public-key cryptographic system (see Chapter 2), the sender (A) of an



Assume that only A and B know cipher key K.
Message = (X1, X2, ..., Xn); X1 = ID.

Figure 8-3. Authentication of the Identity of a Message's Sender Using a Common Key, K, and the Sender's ID in the Message



PWa and PWb are secret passwords known only to A and B.
PWa is included in each message transmitted from A to B, and
PWb is included in each message transmitted from B to A.

Figure 8-4. Authentication of the Identity of a Message's Sender Using Constant Passwords

enciphered message cannot be identified with the receiver's (B) secret key (SKb). This is because the receiver's enciphering key (PKb) is not secret (it is in the public domain) and, therefore, anyone can encipher a message with that key. However, the identity of a message's sender can be established if the sender (A) operates on the message with a secret deciphering key (SKa) (e.g., by deciphering the message). In that case, by enciphering the message with the sender's public key (PKa) and checking that it has been properly recovered, anyone can verify that the message originated with A. However, this form of authentication is achieved at the price of having no secrecy. If secrecy is desired (or required), the deciphered message, $D_{SKa}(M)$, must be enciphered with the receiver's public key (PKb). (For more details, see Digital Signatures, Chapter 9.)

Authentication of a Message's Timeliness

A procedure for verifying that messages are received in the same order they were transmitted by the sender is as follows:

1. If a time-variant quantity Z is known in advance to both the sender and receiver, then the order of transmitted messages can be established by requiring that each message be enciphered using Z as an initializing vector.[3]
2. If a time-variant quantity T is known in advance to both the sender and receiver, then the order of transmitted messages can be established by requiring that T be included in the text of each message.

Let Z1, Z2, . . . ,Zn denote time-variant quantities used in transmitting a set of messages, M1, M2, . . . , Mn, respectively. Assume that block chaining with error propagation is used for message encryption (see Figure 2-16). The Z-values establish the sequence of transmitted messages because the ith initializing vector (Zi) allows only the ith enciphered message (Mi) to be recovered. Zi will not permit a different message (Mj) to be recovered. Therefore, to verify that a message is received in its proper sequence, it is necessary only to decipher the message using the appropriate initializing vector, and to verify that it is properly recovered.

Messages can also be sequenced via a time-variant quantity T which is included in the text of each transmitted message. Let T1, T2, . . . , Tn denote the time-variant quantities used in transmitting messages M1, M2, . . . , Mn, respectively. For example, T1, T2, . . . , Tn could be the nonsecret sequence numbers 1, 2, . . . , n. In that case, the recipient could verify that messages are received in their proper sequence if the procedure in Figure 8-3 were changed to include T as well as ID.

[3] There are many ways in which initializing vectors can be established between two communicants. In general, selecting a satisfactory and secure method will depend on the intended application and environment.

Another approach is to use a list of one-time passwords, T1, T2, . . . , Tn, which are agreed upon in advance. These passwords, if used with the procedure in Figure 8-4, allow the receiver to establish the identity of the sender and verify that messages are received in their proper sequence.

The disadvantage of this procedure is that each communicant must keep a record of the next value of T to be used. In addition, the procedure must handle cases where synchronization has been lost. This can be partially overcome if T is a value taken from a TOD clock. In that case, the receiver can authenticate T by using a simple test of reasonableness.

In a different approach, whenever A informs B that it wants to send B a message, B sends A a random number T which A includes in the text of the prepared message. B ensures that the message has been received in its proper sequence by verifying that the correct value of T has been returned in the text of the message. In this procedure, T is a variable that is produced when needed. However, since T is included in the text of the message, the procedure works only if the receiver also verifies the contents of the message.

Authentication of a Message's Contents

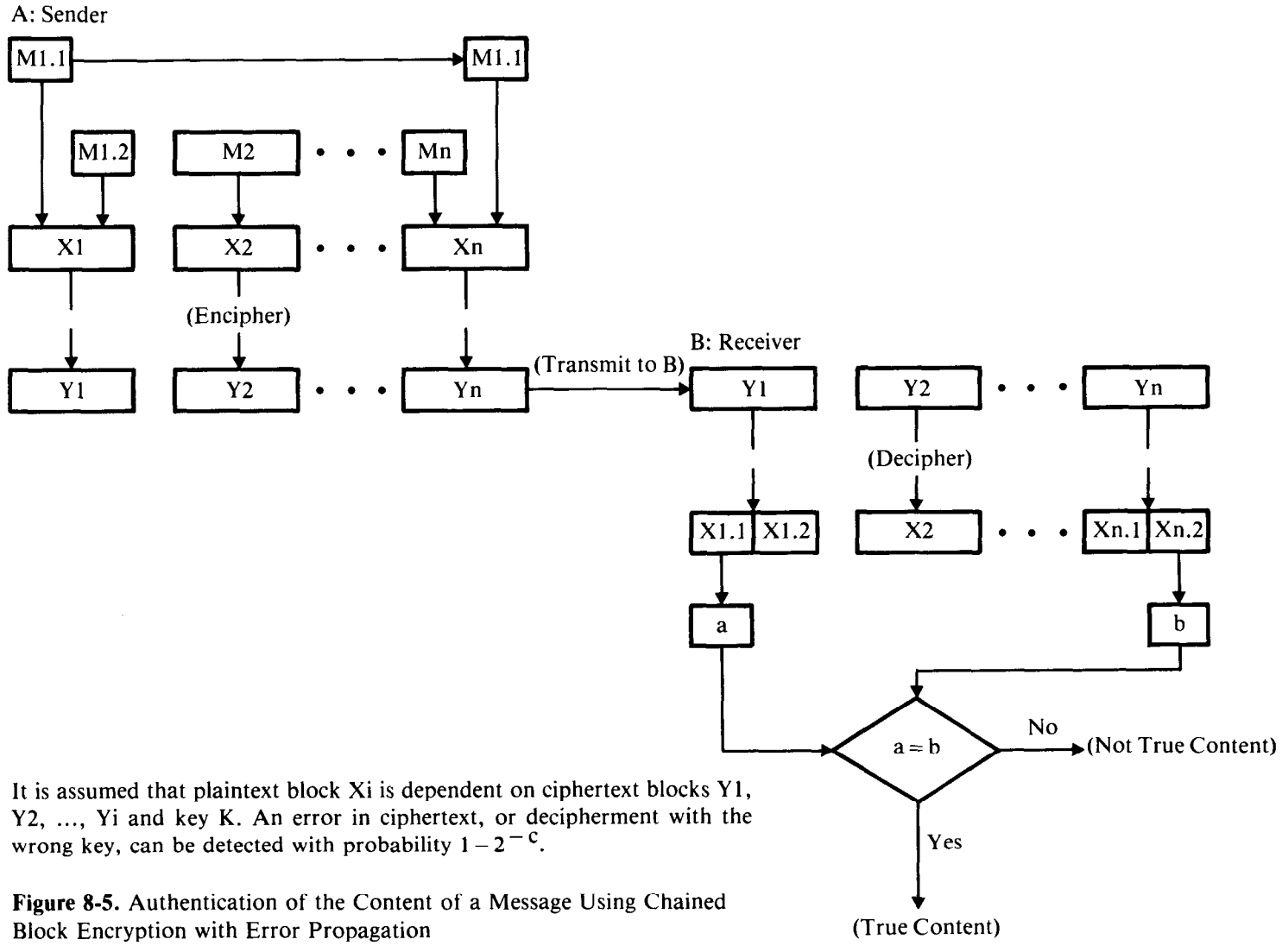
The contents of a message are authenticated by verifying the correctness of an *authentication code* (AC), also called a *message authentication code* (MAC), that is computed by the sender and appended to the message (plaintext or ciphertext) before it is sent to the receiver.[4]

Authentication by an Encryption Method with the Property of Error Propagation

Authentication of the contents of a message is easily accomplished if the message is enciphered with an encryption method that has the property of error propagation. A block chaining scheme such as plaintext-ciphertext feedback (see Figure 2-16) could be used. The procedure requires that a quantity known to the receiver (and sender) be appended to the end of the message prior to encipherment (e.g., the initializing vector Z, or the first block of plaintext X(1)). After decipherment, the contents of the message can be authenticated by verifying that the correct quantity appears at the end of the message. If the quantity appended to the end of the message contains c bits, then any change in the ciphertext will be detected with a probability approximately equal to $(2^c - 1)/2^c = 1 - 1/2^c$.

Figure 8-5 illustrates a procedure in which the first c bits of the message are repeated at the end of the message. Upon encryption, these c bits represent the AC. If an error occurs in the ciphertext, the recovered plaintext (after the point of error) will also be in error. In that case, the first c bits of the recovered plaintext will differ from the last c bits with a probability equal to $1 - 1/2^c$.

[4] Procedures to authenticate a message's contents are discussed in Cryptographic Message Authentication Using Chaining Techniques, Chapter 2.



It is assumed that plaintext block Xi is dependent on ciphertext blocks Y1, Y2, ..., Yi and key K. An error in ciphertext, or decipherment with the wrong key, can be detected with probability $1 - 2^{-c}$.

Figure 8-5. Authentication of the Content of a Message Using Chained Block Encryption with Error Propagation

Authentication by an Encryption Method Without the Property of Error Propagation

If the encryption method does not have the property of error propagation (e.g., chained block encryption using ciphertext feedback, Figure 2-17), then a change to the ciphertext may or may not cause the last block of recovered plaintext to be in error. In this case, an approach slightly different from that just discussed can be used to authenticate the contents of a message.

Since an error in the ciphertext is not propagated in the recovered plaintext but instead is localized, the pattern of bits appended to the end of the message must be a function Δ of the entire message (not merely part of the message). The function Δ transforms a message M of arbitrary length into a relatively short, fixed length, pattern of bits $\Delta(M)$, such that with high probability $\Delta(M)$ will differ from $\Delta(M')$ whenever the recovered message (M′) differs from the original message (M). The enciphered value of $\Delta(M)$ represents the AC. (Δ may be a nonsecret function or a secret function known only by the appropriate communicants.) A procedure in which messages are enciphered using block chaining with ciphertext feedback is illustrated in Figure 8-6. (Ciphertext feedback is illustrated in Figure 2-17.)

Even if Δ were such that one could easily find an M and M′ where $\Delta(M) = \Delta(M')$, this alone would be insufficient to allow the authentication procedure to be defeated. This is because M and $\Delta(M)$ are encrypted. Even if a forged message (M′) could be derived, the opponent could not encrypt it with the proper key. On the other hand, in order to calculate M′ such that $\Delta(M') = \Delta(M)$, the value of M or $\Delta(M)$ must first be known. But since M and $\Delta(M)$ are encrypted, it is assumed that this information is denied to the opponent. Consequently, the opponent can make changes only to the ciphertext, in which case he ordinarily has no control over the recovered plaintext (M′) when the message is deciphered. If the length of $\Delta(M)$ is c bits and the bit patterns $\Delta(M1), \Delta(M2), \ldots, \Delta(Mn)$, are (nearly) randomly distributed over the set of 2^c possible values for n arbitrarily selected messages, then $\Delta(M')$ will differ from $\Delta(M)$ with a probability equal to $1 - 1/2^c$.

Let $\Delta(M) = X1 \boxplus X2 \boxplus \ldots \boxplus Xn$, where $\boxplus$ denotes a hashing operator, and let the augmented message $M^* = X1, X2, \ldots, Xn, \Delta(M)$ be encrypted, e.g., using block chaining with ciphertext feedback (Figure 2-17, i.e., CBC mode). The last block of ciphertext is defined as the AC.

Jueneman has shown that defining $\boxplus$ as modulo 2 addition is not secure if encryption of M* is performed using CBC mode [5]. (A permutation of the ciphertext blocks and/or insertion of an even number of spurious ciphertext blocks will produce a recovered message $M' = X1', X2', \ldots,$ defining $\boxplus$ as modulo 2 addition is not secure if encryption of M* is performed using cipher feedback (Figure 2-30, i.e., CFB mode) or with a key auto-key cipher (Figure 2-12) [6].

A simple method of computing $\Delta(M)$ (still to be more thoroughly analyzed, see Reference 6) is to define $\boxplus$ as modulo 2^{64} addition.

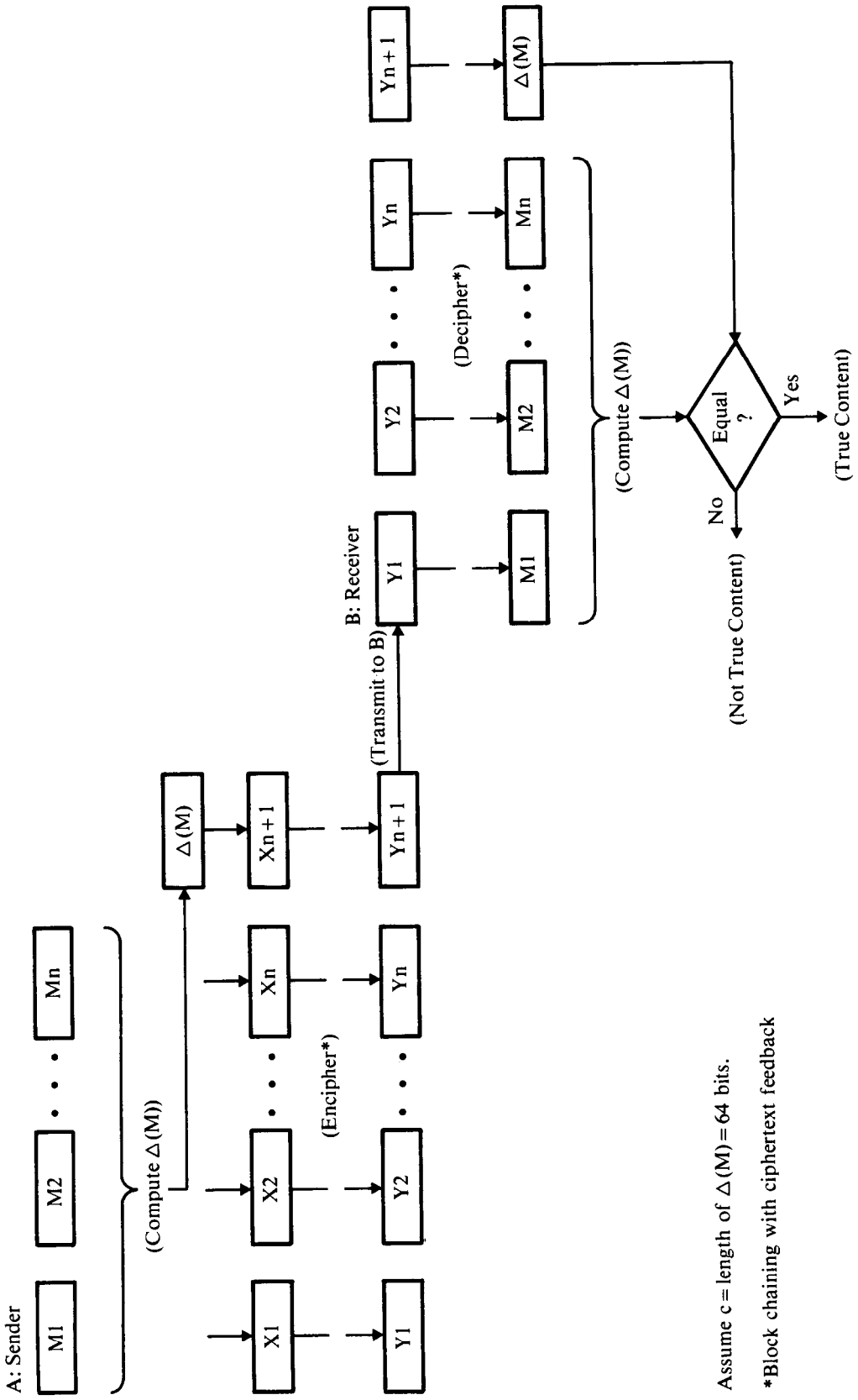


Figure 8-6. Authentication of the Content of a Message where the Authentication Code is Δ(M) Enciphered Under a Secret Key Known Only to the Sender and Receiver

Another, even more secure, method of computing Δ(M) would be to encrypt M under a first secret key (using CBC mode) and define Δ(M) as the last block of ciphertext. The augmented message M* is then encrypted under a second or variant secret key (again using CBC mode).

Authentication Without Message Encryption

It may happen that message authentication is required in cases where messages are sent in unencrypted form, either because there is no need for privacy or because an intermediate network node without an encryption capability must be able to read messages. Thus it is worthwhile to investigate schemes that authenticate unencrypted text (plaintext).

In the discussion that follows, assume that M = (X1, X2, . . . , Xn) is a message to be transmitted in unencrypted form, and that M* = (M, AC) is the augmented message comprised of M and an appended AC. Since M is nonsecret, the authentication procedure must be such that only authorized individuals can compute AC. In effect, this means that the procedure itself, or a parameter used by the procedure, must remain secret. Otherwise, an opponent could compute AC′ for any M′ and substitute (M′, AC′) for (M, AC).

Consider a solution in which the communicants, denoted by the end points in the network, have an encryption capability, whereas the intermediate points in the network have no encryption capability. The cryptographic algorithm and a shared key are used to compute AC. A strong procedure results if the message M is enciphered using block chaining with ciphertext feedback and the AC is defined as the last block of ciphertext (Yn).

However, a weak procedure results if AC is defined as the encipherment of Δ(M) under K and Δ(M) is produced by Exclusive-ORing the blocks of plaintext:

$$\Delta(M) = X1 \oplus X2 \oplus \ldots \oplus Xn.$$

Such a scheme is easily defeated by replacing X1 through Xn − 1 with any quantities X′1 through X′n − 1 and replacing Xn with X′n, where X′n is defined as

$$X'n = X'1 \oplus X'2 \oplus \ldots \oplus X'n - 1 \oplus \Delta(M)$$

In this case, knowledge of M provides an opponent with enough extra information to defeat the procedure successfully.

The procedure for authentication without message encryption could also be defeated if Δ(M) were computed with a polynomial code. Let q be the modulus, and let r be the remainder when M is divided by q. M can be expressed by the equation M = kq + r, where k is an integer that depends on

the value of M. However, for a fixed q and r, and any i not equal to k, each message in the set $\{Mi : Mi = iq + r, i \neq k\}$ would be a candidate that could be substituted for M.

If quantity $E_K(\Delta(M))$ is defined as the AC, then it is evident that an opponent cannot merely substitute a bogus message (M′) for M, since without K the new authentication code, $AC' = E_K(\Delta(M'))$, cannot be computed. However, if Δ has the property that an M′ can be easily found such that $\Delta(M') = \Delta(M)$, then the procedure is exposed to an attack wherein M′ is substituted for M.

Authentication of a Message's Receiver

The methods already discussed for authenticating a message's sender are easily adapted so that the message's receiver can authenticate that it is the intended receiver. Let A and B denote the sender and receiver, respectively. The receiver has enough information to verify that the message has been delivered to the correct destination (receiver) whenever any of the following conditions are satisfied:

1. A and B share a pair of secret keys, where one key is used to encipher messages sent from A to B and the other key is used to encipher messages sent from B to A.
2. A and B share one secret key used to encipher messages sent to each other, and the ID of the receiver is included in the text of each message.[5]
3. A and B share a pair of secret passwords, where A's password is included in the text of each message sent to B, and vice versa.
4. A and B share one secret password, but they include ID information in each message, as discussed in condition 2 above.

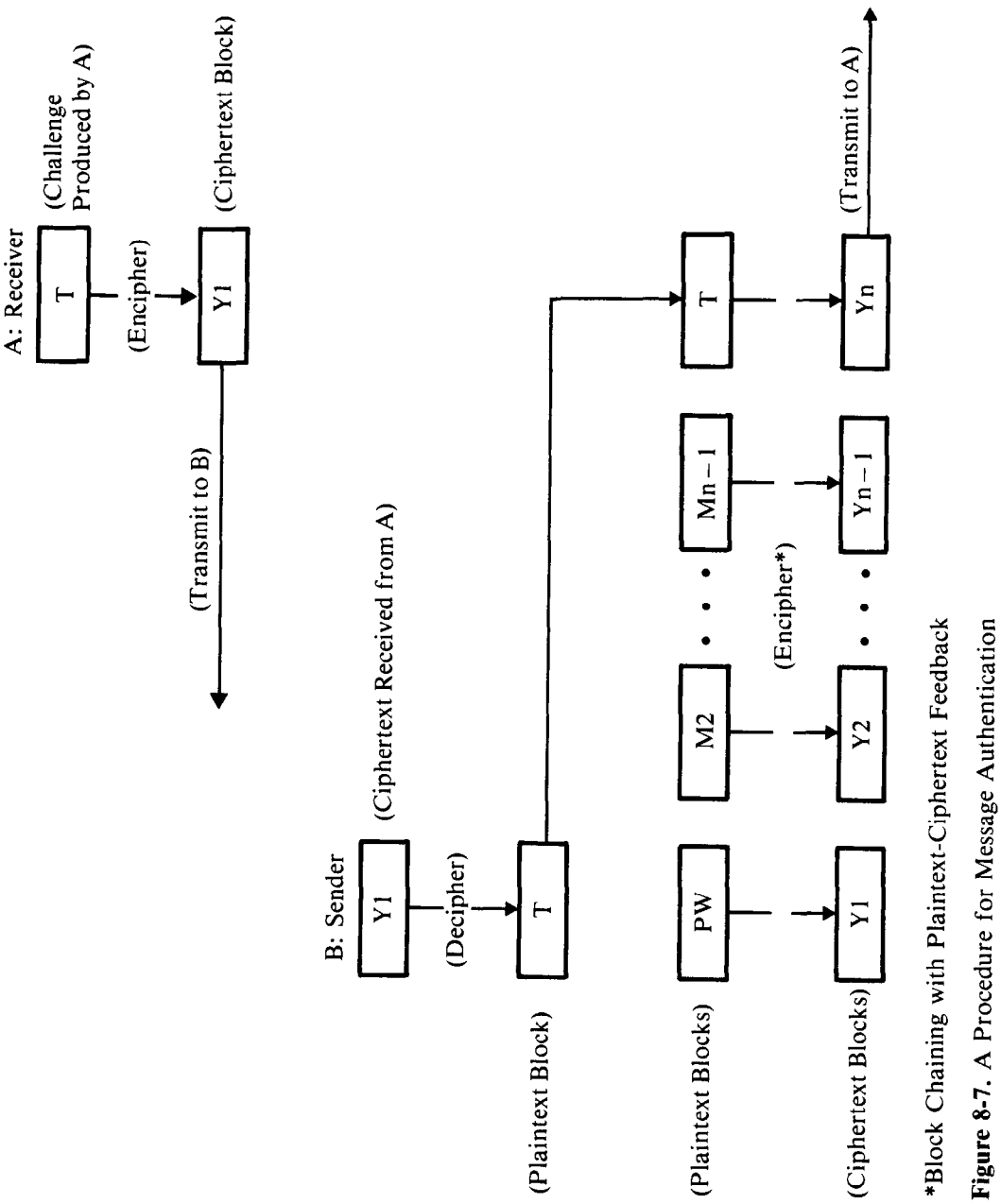
In a public-key cryptographic system (see Public-Key Algorithms, Chapter 2), the receiver uses its secret deciphering key to verify that it is the message's intended receiver. The receiver knows that a message was directed to it if the correct text is recovered after the message has been deciphered with its secret key.

A Procedure for Message Authentication

Several of the previously discussed ideas are now combined into a single procedure for authenticating messages (Figure 8-7). The procedure has the following characteristics:

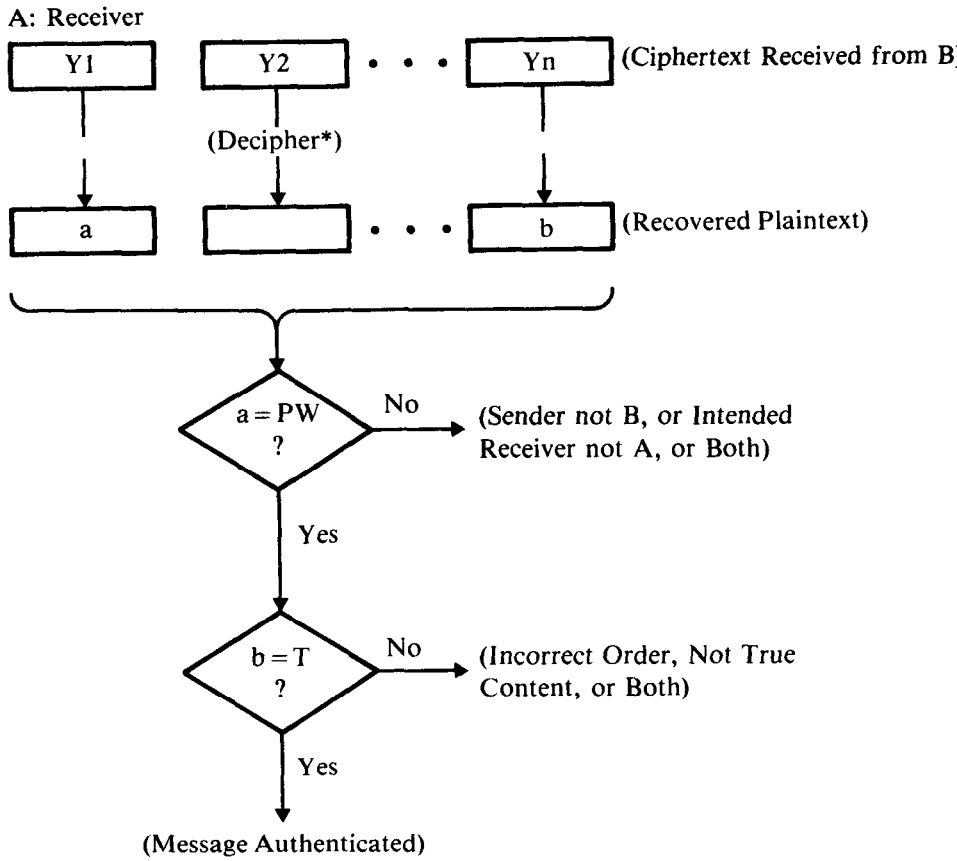
1. An encryption method with the property of error propagation is used to authenticate the contents of a message.
2. Passwords are used to authenticate the sender and receiver (i.e., the sender and receiver share secret passwords).

[5] Since there are only two communicants, an acceptable protocol could be established in which only the ID of the sender is included in the text of the message.



*Block Chaining with Plaintext-Ciphertext Feedback

Figure 8-7. A Procedure for Message Authentication



*Block chaining with plaintext-ciphertext feedback

Figure 8-7 (cont'd).

3. A time-variant quantity T is used to verify that messages are received in their proper sequence.

If the cryptographic key used for encipherment is a personal key, KP, then the password, PW, may be omitted from the procedure. On the other hand, if the key used for enciphering is a session key, KS, then the following can be said:

1. The sender need only include PW in the first transmitted message. This is because at session initiation a correspondence between PW and KS is established that lasts for the duration of the session. Once PW has been verified, KS and T can be used to authenticate the message's sender and intended receiver.
2. If T is omitted from the procedure, it is still not possible for messages intercepted during one session to be successfully redirected to one of

the communicants during another session. This is because KS is also a time-variant quantity—a message enciphered with one session key is not properly deciphered with a different session key. In that case, the message's content can be authenticated using the method described in Figure 8-5 (i.e., by appending a known value to the end of the message and enciphering the result with an encryption method that has the property of error propagation).

Generally, the quantity T should contain at least 16 bits (c = 16). If greater integrity is desired, up to 64 bits could be used conveniently with DES.

If passwords are entered into the system through a single standard interface, such as at a remote terminal during sign-on, additional measures can be taken to enhance the security of the personal identification procedure (see Chapters 10 and 11). For example, passwords could be enciphered under a special cryptographic key resident within the terminal, or each password could be treated as a cryptographic key (provided it has enough combinations) and used to encipher a nonsecret value that has been agreed upon in advance. An opponent who intercepts a transformed password cannot enter this value through the standard interface and gain entry to the system.

In the final analysis, there is no set of authentication procedures that can satisfy the processing and security requirements of all conceivable applications implemented under all operating conditions. Thus each method of authentication (origin, contents, timeliness, and intended recipient), or combination thereof, must be evaluated according to the particular application for which it is intended. In so doing, a set of assumptions about the type of information an opponent would have available to defeat the method must also be established. Selection of a method is accomplished by ensuring that it defends against the opponent's anticipated attacks.

AUTHENTICATION OF TIME-INVARIANT DATA

Computer system integrity is the state that exists when there is complete assurance that the system will perform as intended by its designers. Historically, computer system integrity has been based on the concept of accidental error [7]. That is, ways were sought to provide protection only from the accidental loss or destruction of data. No attempt was made to prevent deliberate tampering with the operating system or stored data.

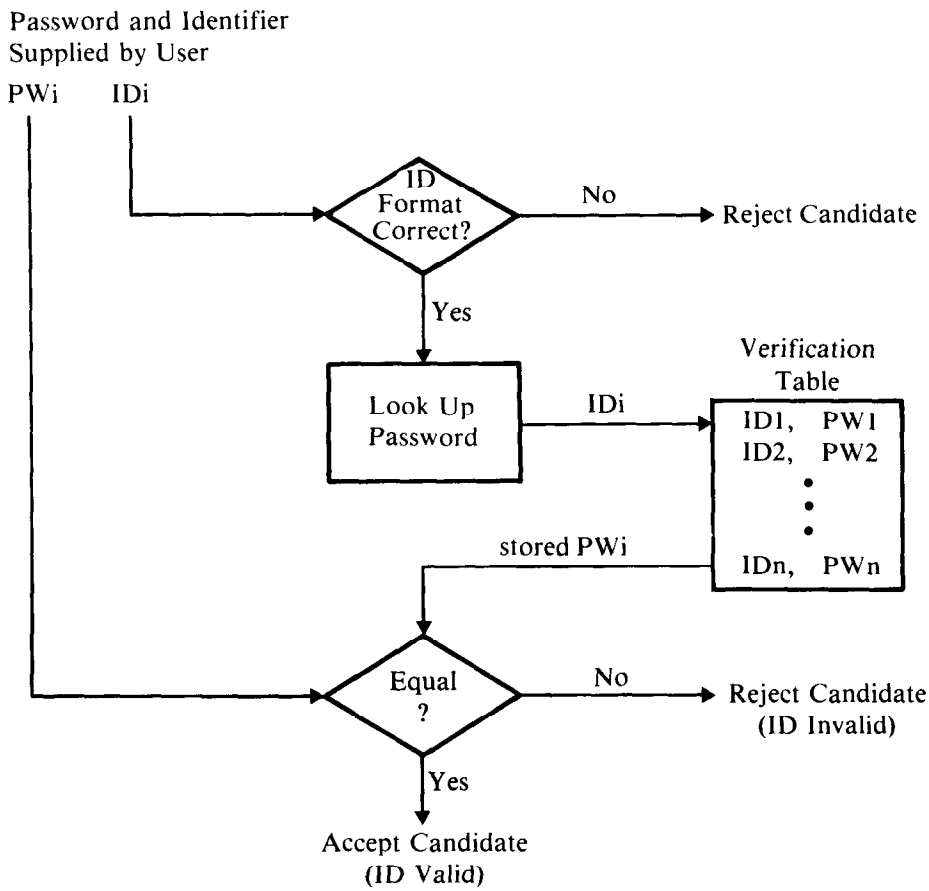
Today it is recognized that data security[6] must include protection from intruders who deliberately attempt to gain unauthorized access to protected resources. Authentication methods that permit only authorized system users to access and manipulate system resources are essential to achieve system integrity. Authentication is the process which proves that someone or something should be accepted as being valid or genuine.

[6] *Data security* is defined as the protection of data from unauthorized disclosure, destruction, and modification, either by accident or intent.

Authentication of Passwords

A common method of authenticating time-invariant data is by comparing the data with a copy of those data which has been stored elsewhere in the system. For example, a user's ID (i.e., the users name or account number) is verified by comparing a supplied password (PW) with a corresponding password stored within the system (Figure 8-8).

This method of checking can be improved whenever passwords are entered into the system from a remote entry device (terminal or hand-held unit attached to a terminal) capable of encryption [2, 3]. A cryptographic operation, Δ, is used to protect the secret PW by transforming it into a nonsecret quantity, Δ(PW). The quantity Δ(PW) is then transmitted to a central facility to verify the user's identity by comparing it with a similarly precomputed value stored in the system.



ID is a user identifier (name, account number).

Figure 8-8. Authentication Using a Stored Table of Passwords

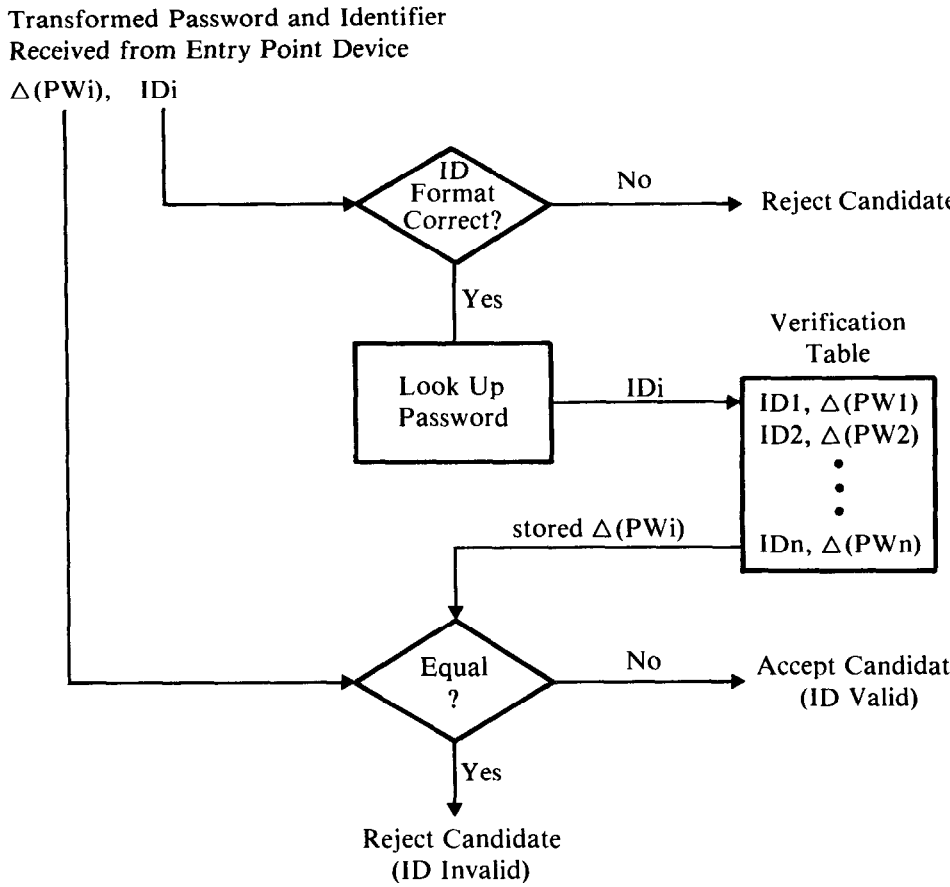
When the cryptographic operation (Δ) is based on a strong algorithm such as DES, the value of PW cannot be deduced, even using involved cryptanalysis. In some cases, this degree of protection is absolutely essential (e.g., when PW is the basis for commercial wire transfer of funds as in electronic funds transfer, EFT). In general, the services provided by such a system (debiting and/or crediting customer accounts) are predicated on knowing the customer's identity.

Modern banking methods incorporate self-service terminal devices at which customers conduct their banking business. Such methods do not require customer recognition by an institution employee. Instead, the device and supporting system must be responsible for identifying the customer. Typically, when a transaction is initiated, the customer supplies a secret parameter (a password or more precisely a personal identification number, PIN) together with a claimed ID. The PIN must be managed securely to preserve the integrity of the transaction.

One way to protect PW (or, equivalently, PIN for EFT systems) would be to encipher PW under a secret key available at an entry point to the system (i.e., $\Delta(PW) = E_{Key}(PW)$). Alternatively, when PW has enough combinations, it may be treated as a cryptographic key and used to encrypt (at the entry point) a test phrase such as the user's ID (i.e., $\Delta(PW) = E_{PW}(ID)$). The latter approach has the advantage that only user-supplied quantities are needed to effect the transform Δ, whereas in the former case a system-managed key is required.

However, with most customer-oriented EFT systems, the PIN (PW) is usually comprised of only four to six characters, so that it is easy to remember and convenient to use. The short length also makes the PIN susceptible to exhaustive attacks (methods to determine the PIN using direct searches or trial and error techniques). The PIN's short length thus makes it unsuitable for use as a cryptographic key. However, an approach which overcomes the objection of a short PIN and at the same time enhances key management would be for each customer to supply also a secret *personal key* (KP). With such an approach, exhaustive attacks against PIN and KP could be blocked by defining $\Delta(PIN, KP) = E_{KP \oplus PIN}(ID)$. (See Applying Cryptography to Electronic Funds Transfer Systems–Personal Identification Numbers and Personal Keys, Chapter 11.) In any case, the purpose of Δ is to transform PW into a nonsecret form so that it can be safely transmitted to a central facility (host processor) for authentication.

Upon receipt at the host processor, $\Delta(PW)$ is used to verify the user's identity by comparing it with a similarly precomputed value stored in the system (Figure 8-9). Storing $\Delta(PW)$ in a verification table at the central facility will protect the secrecy of passwords (Figure 8-9), but it will not protect against an intruder who could alter data stored in the verification table. If an opponent were able to create $\Delta(X)$ for an arbitrary value of X, and replace $\Delta(PWi)$ with $\Delta(X)$ for some user identifier (IDi) in the verification table, then entry to the system could be gained by inputting the known value of X at any entry point. Moreover, a legitimate user of the system with IDj could gain entry under a different identifier, say IDi, by replacing $\Delta(PWi)$ with $\Delta(PWj)$. In either case, once entry had been achieved, $\Delta(PWi)$ could be put back into the verification table to prevent subsequent detection.



Δ is a cryptographic operation used to transform PW at its entry point into the system. For example, Δ(PW) could be represented by $E_{Key}(PW)$ or $E_{PW}(ID)$, where ID is a user identifier (name, account number).

Figure 8-9. Authentication of Passwords that have been Transformed Under a Cryptographic Operation Δ

An active wiretap threat in which Δ(PWi) and IDi are intercepted and later retransmitted to the host could be blocked by cryptography (e.g., by implementing a protocol for message authentication between the entry point (terminal) and host). However, defenses against the threat of wiretaps in communications networks can be achieved through communication security techniques described in Chapters 4–7 and are not treated in the present discussion.

Gaining unauthorized entry to the system by altering or manipulating the verification table can be prevented by using the technique described for password verification. This technique protects against misuse of information stored in a verification table, even by those who manage and maintain the table and

who have legitimate access to it. This authentication technique makes use of a special test pattern (TP) not equal to Δ(PW), which can be used at any later time together with a special cryptographic operation to verify Δ(PW).

In the discussion that follows, ID and Δ(PW) are assumed to be 64-bit quantities. (Parameters less than 64 bits can be padded to 64 bits using some agreed upon protocol; e.g., padding with zero bits.) Moreover, it is assumed that an opponent has read and write access to the verification table.

Authentication Using Test Patterns Generated from the Host Master Key

This procedure, which assumes the DES algorithm, is a general technique to authenticate the contents of a 64-bit block of data (X). For example, password authentication is performed by setting X equal to Δ(PW). However, unlike the method of password authentication described above, this method is not exposed to attacks based on an alteration or manipulation of the verification table.

The basic idea can be implemented within a host processor that has an encryption capability by providing two cryptographic authentication operations: *authentication forward* (**AF**) and *authentication reverse* (**AR**). These two operations are defined

$$\mathbf{AF}: \{\mathbf{X}, \mathbf{TP}\} \longrightarrow \mathbf{VP}$$
$$\mathbf{AR}: \{\mathbf{X}, \mathbf{VP}\} \longrightarrow \mathbf{TP}$$

where the contents in the braces indicate the input to the host's cryptographic facility, the arrow points to the result, and

X = the data block to be authenticated
TP = test pattern associated with X
VP = verification pattern assigned to X

TP and VP are nonsecret parameters, each containing 64 bits.

Basically, AR computes the test patterns needed to initialize the system and AF validates data parameters. Hence AR must be restricted to certain special runs with particular users—those responsible for initializing and updating the system. One way to ensure that AR is executed only under secure conditions is to require that it be activated by physically turning a key-operated security lock.[7] AF is available to any program or system user needing to validate data parameters. Therefore, AF must have the property that TP cannot be computed or deduced from X and VP.

For each Xi requiring authentication, there is a corresponding test pattern (TPi) that must be computed in advance. For this reason, the quantity to be authenticated must be known beforehand in clear form. The technique cannot be used when the data to be authenticated are time-variant, such as messages transmitted in a network. Otherwise, this method is completely general; it can be applied to any quantity (Xi).

[7] In general, what is needed to perform the mappings AF and AR is a trapdoor one-way function [8].

Authentication of Xi takes place by exercising the AF operation

$$AF: \{Xi, TPi\} \longrightarrow VPi$$

and comparing the result (VPi) for equality with the corresponding predefined verification pattern to determine whether Xi should be accepted. If the two values are equal, Xi is accepted. Otherwise, Xi is rejected. Only the correct Xi and TPi will produce VPi. A different result ($\neq$ VPi) is produced if one or both of the input parameters are changed.

VPi must be a value related to the identity of Xi. For example, VPi could reflect the name of Xi or the address in main storage where Xi is stored (assuming Xi's storage location does not change). If VP1 = VP2 = . . . = VPn = constant, then by replacing (Xi, TPi) with (Xj, TPj) the system could be deceived into using Xj in place of Xi.

Let the relationship between VPi and IDi (the identifier of Xi) be expressed as

$$VPi = \nabla(IDi), \quad i = 1, 2, \ldots, n$$

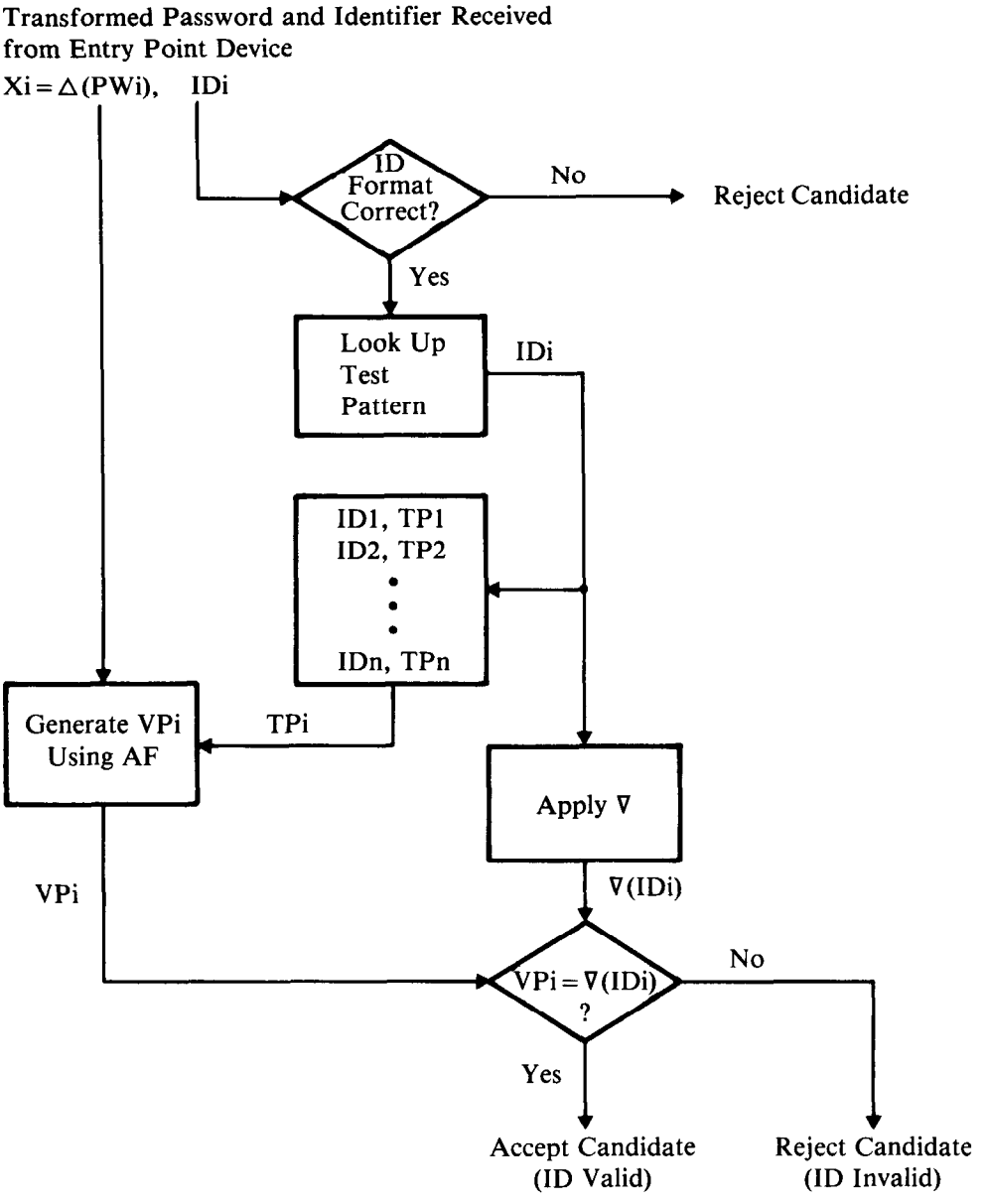
where ∇ is a function which prevents ID from being computed from VP. For example, ∇(ID) could be defined as $E_{ID}(C)$, where C is a nonsecret constant value.[8] Figure 8-10 illustrates how the test pattern scheme could be applied to password authentication, where the quantity to be authenticated is Xi = Δ(PWi).

The reason for employing a ∇ function with these properties can be explained by referring to Figure 8-10. Suppose an opponent chooses a password (PW) and obtains Δ(PW) as the result of a sign-on attempt (e.g., via a wiretap). Assume also that VP = ID. The opponent now selects an arbitrary test pattern (TP) and exercises AF using Δ(PW) to obtain VP, which subsequently is specified as ID. By creating an entry in the verification table for the forged values of ID and TP, the opponent could then sign-on the system using the selected password and the corresponding derived identifier.

This attack may already be partially blocked if, for example, the ID consists only of alphameric characters (A, B, . . . Z, 0, 1, . . . , 9) and an initial consistency check is performed to test its validity. This type of consistency check is assumed in Figures 8-8, 8-9, and 8-10 ("ID format correct?" decision block).

VPi does not need to be kept secret because a potential intruder cannot deduce the correct TPi for a given Xi and VPi. (Note that TPi can be created only by using the AR operation and the AR operation is available only to authorized persons.) It should also be realized that VPi must be checked in a dynamic sense (i.e., it must not be compared to a stored system value). Otherwise, an opponent using AF could compute VP for a valid X and an arbitrary TP and replace the correct table value with the value so obtained

[8] Greater security can be achieved if the method is extended to two test patterns. For example, one could define VPi1 = E_{ID}(C1) and VPi2 = E_{ID}(C2), where C1 and C2 are nonsecret constant values. In that case, Xi and VPi1 would be used to generate TPi1, and Xi and VPi2 would be used to generate TPi2.



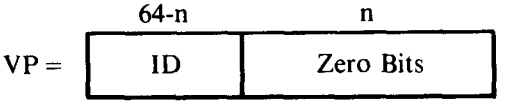
Δ is a cryptographic operation used to transform PW, at its entry point, into a nonsecret quantity. Δ(PW) could be defined as $E_{Key}(PW)$ or $E_{PW}(ID)$, where ID is used as an identifier. ∇ is a cryptographic operation used at the host to transform ID into VP.

Figure 8-10. Authentication Based Upon a Table of Test Patterns

A Short Analysis

It would be unwise to define VP equal to ID. However, to gain further insight into the matter, assume the following for the sake of discussion:

1. The length of ID is less than or equal to 64 bits.
2. VP is defined as the ID padded with enough zero bits to create a 64-bit quantity.



3. Each of the m = 2^{64-n} combinations for ID are valid, and hence there are m possible values for VP.

In theory, the authentication procedure could be attacked in the following way:

1. For each of the m values of ID, the pairs (ID1, ∇(ID1)), (ID2, ∇(ID2)), . . . , (IDm, ∇(IDm)) are computed.
2. For an arbitrary Xi, the AF operation is exercised using different values of TP. This is continued until a VP is found which matches one of the m values in the list: ∇(ID1), ∇(ID2), . . . , ∇(IDm).

Since there are 2^{64} possible verification patterns, the probability that an arbitrary TP will produce a valid verification pattern is $2^{64-n}/2^{64} = 1/2^n$. Therefore, about 2^{n-1} trials (exercising AF) are needed, for a given X, to find a TP that produces a VP in the list:

$$\nabla(ID1), \nabla(ID2), \ldots, \nabla(IDm)$$

As a consequence, about $2^{64-n} + 2^{n-1}$ trials are needed to carry out the attack. The ∇ function must be exercised 2^{64-n} times to obtain the relationship between ID and ∇(ID) for each ID, and the AF operation must be exercised 2^{n-1} times to find an appropriate TP. If n is small (the number of IDs is large), then the ∇ function must be exercised more frequently than the AF operation. If n is large (the number of IDs is small), then the AF operation must be exercised more frequently then the ∇ function. The value of n that is selected thus determines the opponent's work factor.

Table 8-1 provides a summary of the effective security achieved with each of the three authentication procedures: unenciphered passwords, transformed passwords, and test patterns.

Implementing AF and AR

One way to implement AF and AR using a conventional cryptographic algorithm like DES would be to define special encipher and decipher opera-

Protection Method	Opponent can read from, but not write into verification table.	Opponent can read from and write into verification table.
Password stored in verification table (Figure 8-8).	Actual Verification information can be obtained. Opponent can LOGON pretending to be someone else.	Actual Verification information can be obtained. New verification information can be created, i.e., a new user can be introduced into the system.
Password stored in verification table (Figure 8-9).	No subversion possible.	Actual verification information cannot be obtained. New verification information can be created, i.e., a new user can be introduced into the system.
Test patterns stored in verification table (Figure 8-10).	No subversion possible.	No subversion possible.

Table 8-1. Comparison of Different Verification Procedures

tions using a new variant[9] of the host master key—a variant defined solely for the purpose of authentication [9]. With a public-key algorithm such as the trapdoor knapsack or the RSA algorithm (see Block Ciphers, Chapter 2), AF and AR would be implemented via the public and private keys, respectively.

Let KM5 represent the fifth variant of the host master key and define

$$\text{AF: } \{Xi, TPi\} \longrightarrow E_{D_{KM5}(Xi)}(TPi) = VPi$$

$$\text{AR: } \{Xi, VPi\} \longrightarrow D_{D_{KM5}(Xi)}(VPi) = TPi$$

Figure 8-11 describes the steps taken by the cryptographic facility to perform the AF and AR operations.

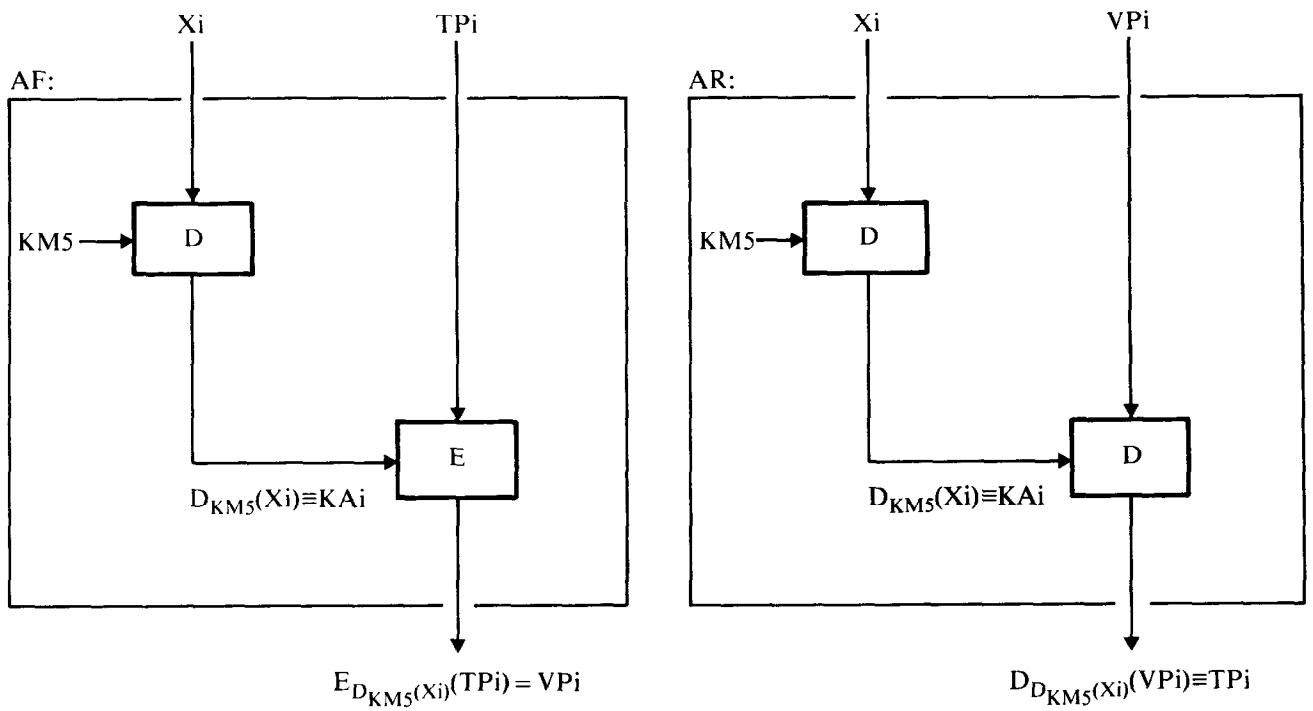
Let KAi be a cryptographic key defined as follows:

$$KAi = D_{KM5}(Xi), \quad i = 1, 2, \ldots, n$$

The integrity of the authentication procedure is assured because of the following:

1. By using a special variant of the host master key (KM5), it is not possible to use other cryptographic operations, singularly or in combina-

[9] A *variant* of the host master key is derived by inverting selected bits in the master key (see Protection of Host Keys, Chapter 4). In effect, a cryptographic system with multiple master keys is achieved when in fact only a single master key is stored in the cryptographic facility. By defining a cryptographic operation to be dependent upon a specific host master key variant, one effectively isolates that operation from all others without loss of cryptographic strength.



AF is used to authenticate Xi with TPi; AR is used to generate test patterns. Xi is the value to be authenticated; TPi and VPi are the test pattern and verification pattern corresponding to Xi. KM5 is the fifth variant of the host master key.

Figure 8-11. The AF and AR Operations

tion, to subvert or reverse the effect of AF and AR. Likewise, no other operation is provided that will allow encipherment or decipherment under KM5. Furthermore, KAi never appears in the clear outside of the cryptographic facility. Hence for an arbitrary Xi there is no way to determine the corresponding KAi, and vice versa.

2. AF allows arbitrary encipherment under KAi, but no (inverse) operation is provided to allow decipherment under KAi.

An attack in which the AF operation and the ∇ function are repeatedly exercised in order to find an X, TP, and ID such that

$$\text{AF: } \{X, TP\} \longrightarrow VP = \nabla(ID)$$

can be thwarted by making the number of trials sufficiently large (see also footnote 8).

The method for authentication discussed herein is a general scheme for validating the contents of a time-invariant data variable of arbitrary length. The test of legitimacy is based on a previously computed, nonsecret, nonforgeable test pattern (TP), whose functional relationship to the data is verified (at any later time) using a dynamically generated, nonsecret verification pattern (VP). The method does not rely on either the security or integrity features of a host processor and operating system to protect a verification table.

The cryptographic principles upon which this method is founded include:

1. Ability to create a test pattern using a cryptographically secure operation involving the quantity to be authenticated and a secret key resident in a host.
2. Ability to deny unrestricted usage of AR, the cryptographic operation used to create test patterns. (For all practical purposes, the user sees only a one-way function which allows a verification pattern to be generated from the quantity to be authenticated and the test pattern.)

An Implementation Using the Cryptographic Operations Proposed for Communication and File Security[10]

Except for *set master key*, each of the host's cryptographic operations produce outputs that depend on either the host master key or one of its derived variants (see Chapters 4 and 5). There are many mathematical identities involving the cryptographic operations which might be the basis for an authentication procedure. One such technique is described here.

At the time the host's master key is entered into the cryptographic facility (read into main storage and used as the object of a set master key operation), the following steps are performed:

1. KM1 and KM2 are obtained directly from KM0

[10] Only the cryptographic operations defined in Chapter 4 are used.

2. EMK: $\{KM1\} \longrightarrow E_{KM0}(KM1)$
3. DCPH: $\{E_{KM0}(KM1), C\} \longrightarrow D_{KM1}(C)$
4. EMK: $\{KM2\} \longrightarrow E_{KM0}(KM2)$
5. ECPH: $\{E_{KM0}(KM2), D_{KM1}(C)\} \longrightarrow E_{KM2}(D_{KM1}(C))$

C is an arbitrary nonsecret value which remains constant for all generated test patterns. The quantity

$$KA \equiv E_{KM2}(D_{KM1}(C))$$

is called the *system authentication key*. The system authentication key is a secret parameter used to create test patterns. It is created during a secure computer run, before or during the time when test patterns are created. A copy of KA should be sent to a suitable output device and stored in a secure place for future use. KA and all quantities involved in the computation are erased from main storage when processing is complete.

Test patterns are also created during a secure computer run. KA is made available either by reading it into main storage or by initially creating it. It is assumed that Xi and VPi are available for each required TPi. The procedure is as follows:

1. DCPH: $\{Xi, VPi\} \longrightarrow D_{D_{KM0}(Xi)}(VPi) = Qi$
2. RTMK: $\{KA, Qi\} \longrightarrow E_{KM0}(D_{D_{KM1}(C)}(Qi)) \equiv TPi$

The output of step 2 is defined as the generated test pattern. KA and all quantities involved in the computation are erased from main storage upon completion of this sequence.

Authentication of Xi is performed in the following manner:

1. RFMK: $\{C, TPi\} \longrightarrow D_{D_{KM0}(Xi)}(VPi)$
2. ECPH: $\{Xi, D_{D_{KM0}(Xi)}(VPi)\} \longrightarrow VPi$

The value C used here must be the same as that used to create TPi. If the result at step 2 agrees with the known value of VPi, then Xi is accepted; otherwise, Xi is rejected.

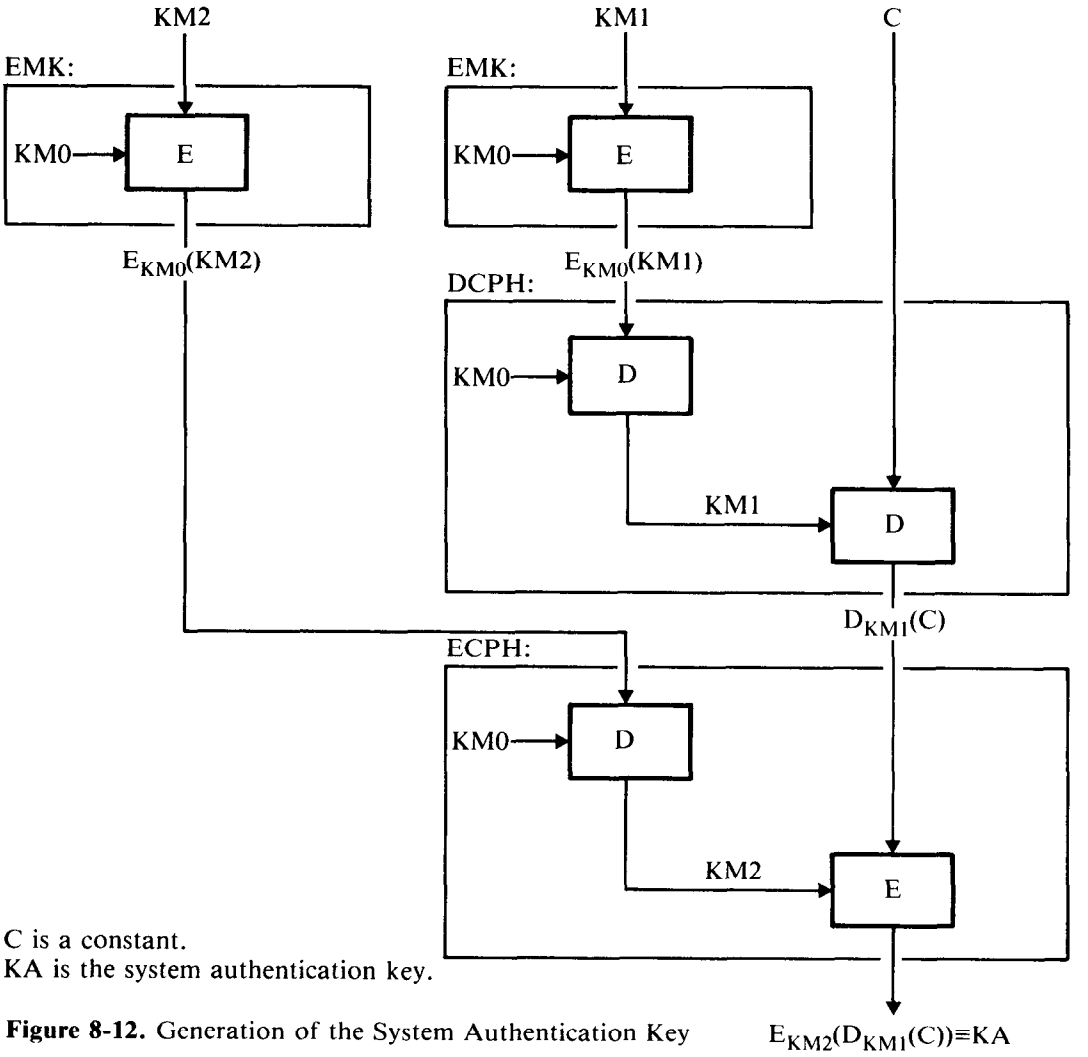
The integrity of the procedure depends on the secrecy of quantity $D_{KM1}(C)$. This quantity appears in clear form only within the cryptographic facility. Moreover, there is no way to use the cryptographic operations to decipher under KM1 (the first variant of the host's master key). Because of this, there is no way for an opponent to derive $D_{KM1}(C)$.

It is important, however, that an opponent not be able to alter or control the value of C. If an opponent could manipulate the value of C and could also obtain X and $E_{KM1}(X)$, for some arbitrary value of X, the procedure would be inadequate for security. For example, if $E_{KM1}(X)$ were substituted for C, this would yield the value X for the intermediate quantity $D_{KM1}(C)$.

Since the opponent knows the value of X, he could now forge test patterns for any desired quantities.

It is conceivable that values of X and $E_{KM1}(X)$ could be acquired, provided that the system's security is violated; for example, if KMT could be illegitimately acquired from a terminal and the corresponding value of $E_{KM1}(KMT)$ could be obtained from the host's key table. On the other hand, if the opponent could manage to capture $E_{KM1}(KM1)$, which is a system activation key (see Generation of Key-Encrypting Keys, Chapter 6), then the EMK and RFMK operations could be used to encipher arbitrary values under KM1.

Figure 8-12 illustrates the procedure for generating KA. Figure 8-13 illustrates the procedure for generating test patterns and for authenticating objects.



C is a constant.
KA is the system authentication key.

Figure 8-12. Generation of the System Authentication Key

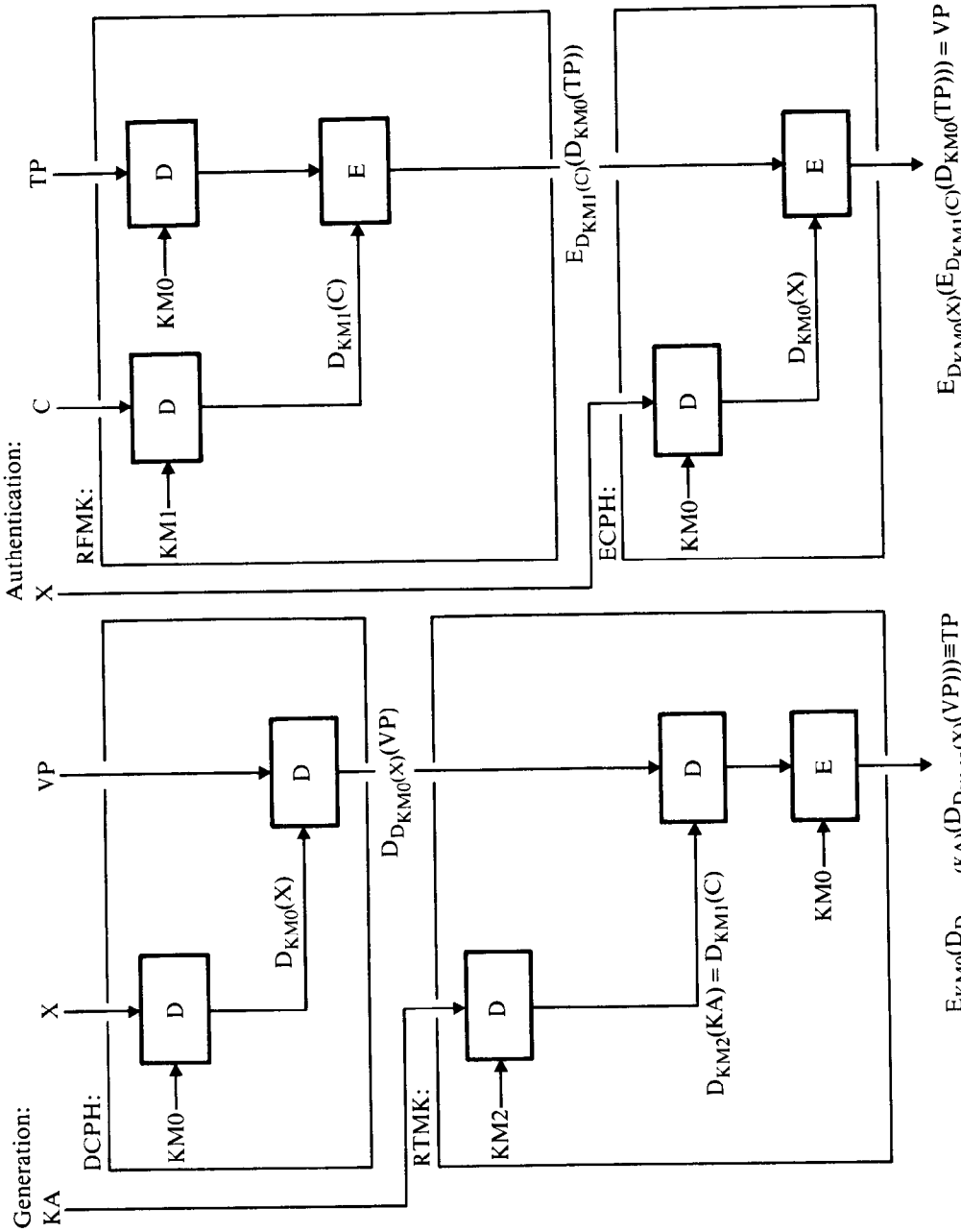


Figure 8-13. Test Pattern Generation and Authentication of Data Parameters

A Procedure for Authentication of Cryptographic Keys

Except for the host master key, which is stored in clear form within the host's cryptographic facility, all keys used by the host system are maintained in enciphered form. Data-encrypting keys, such as session and file keys, are protected by encipherment under the host master key, KM0. Secondary keys, such as terminal master keys, secondary communication keys, and secondary file keys, which are stored in the cryptographic key data set (CKDS), are protected by encipherment under either the first or second variant of the host master key, KM1 or KM2 (see Chapters 4 and 5). Several considerations bear upon any decision to provide authentication for cryptographic keys.

Secondary keys (stored enciphered under KM1 or KM2) could be parity-adjusted at key generation, and the key's parity could be checked at the time it is deciphered in the cryptographic facility. Since each byte in the external key consists of seven key bits and one parity bit, any alteration of the encrypted key due to noise would be detected with a probability of 255/256.

Session keys (KSs) and file keys (KFs) are not parity-adjusted at the time they are created, since they are defined to be enciphered already under KM0 or KNF. A data-encrypting key (KS or KF) is created by generating a random number RN, and defining RN as KS enciphered under KM0, or as KF enciphered under KNF. On the average, about 128 pseudo-random numbers would have to generated before one would be found that provided the recovered KS or KF with correct parity. Such a key generation procedure would be too inefficient for most purposes.

As to communication security, any alteration of stored secondary communication keys or terminal master keys would cause the initiated session keys to be different for each end user. Such a condition would be detected by the handshaking procedure.

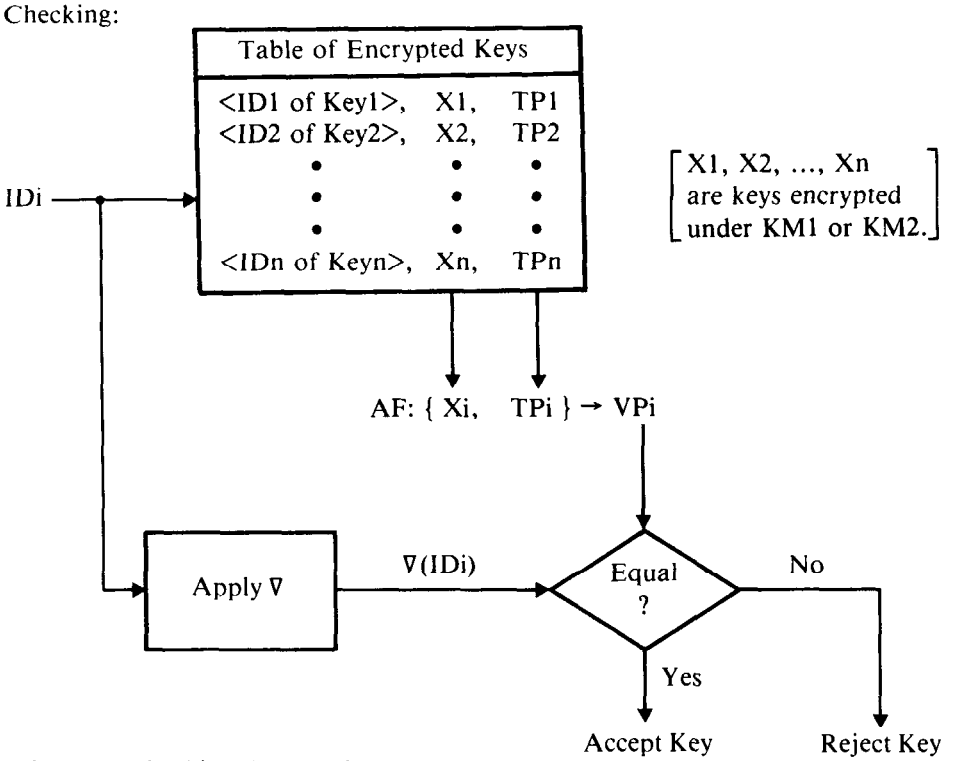
As to file security, any alteration of stored secondary file keys would cause the recovered secondary file key to have correct parity with a probability of 1/256. Under such conditions, an incorrect secondary file key would go undetected with probability 1/256, and hence a file (such as one for backup) created using an incorrect secondary file key could not be recovered unless the same altered copy of the secondary file key were used.

If there were no cryptographic authentication, an opponent could transpose keys stored in the CKDS, and indexing errors could cause use of the wrong key without detection of the error. In each case, the parity of the key would still be correct, and this might allow unwanted or unanticipated cryptographic quantities to be derived that could be damaging to the security of the system. If an opponent could encipher under KM1 or KM2, without necessarily knowing the value of either KM1 or KM2, he could systematically replace existing keys with his own keys.

A technique employing test patterns can easily be adapted to validate encrypted keys stored in the CKDS (e.g., prior to their use within the cryptographic facility) (Figure 8-14). However, it should be realized that if the procedure is bypassed due to an unauthorized modification of programming (software), then the intended protection would not be achieved. Therefore, such an authentication procedure reduces the risk associated with

Generation: AR: { Xi, ∇(IDi) } → TPi

Checking:



Alternate Checking Approach:

RFMK: { C, TPi } → Ωi
ECPH: { Xi, Ωi } → VPi

Note: C is a dynamically generated constant.

Figure 8-14. Authentication of Cryptographic Keys Using Test Patterns

using an incorrect cryptographic key, but it cannot eliminate the problem altogether.

Another Authentication Method Using Test Patterns Generated from the Host Master Key

Another way to define AF and AR using a conventional cryptographic algorithm like DES is described below:[11]

$$\text{AR: } \{\text{IDi, Xi}\} \longrightarrow \text{TPi}$$

$$\text{AF: } \{\text{IDi, Xi, TPi}\} \longrightarrow \begin{bmatrix} = 1 \text{ if (Xi, IDi, TPi) is a valid triple} \\ \neq 1 \text{ otherwise} \end{bmatrix}$$

[11] The method is based on a similar method suggested by Smid [10]. See also A Key Notarization System for Computer Networks, Chapter 9.

In this case, only test patterns are used. Verification patterns are not required. This is because TPi is tested for validity inside the cryptographic facility, that is, the output of the AF operation indicates only the result of the test (= 1 or ≠ 1).

Let KM5 represent the fifth variant of the host master key and define

$$K0 = E_{KM5}(0)$$
$$K1 = E_{KM5}(1)$$

K0, K1, and KM5 are keys used solely by the AF and AR operations; they effectively isolate AF and AR from the other cryptographic operations defined to the cryptographic facility. AF and AR are defined as follows:

$$\text{AR: } \{IDi, Xi\} \longrightarrow E_{K1 \oplus IDi}(E_{K0}(Xi)) \equiv TPi$$
$$\text{AF: } \{IDi, Xi, TPi\} \longrightarrow D_{K1}(E_{TPi*}(D_{TPi}(E_{K1}(1))))$$

where

$$TPi* = E_{K1 \oplus IDi}(E_{K0}(Xi))$$

If TPi* = TPi, the output of the AR operation is equal to 1. Otherwise, the output is a complex function of K1, TPi*, and TPi and (in all probability) its value is unequal to 1. Figure 8-15 describes the steps taken by the cryptographic facility to perform the AF and AR operations.

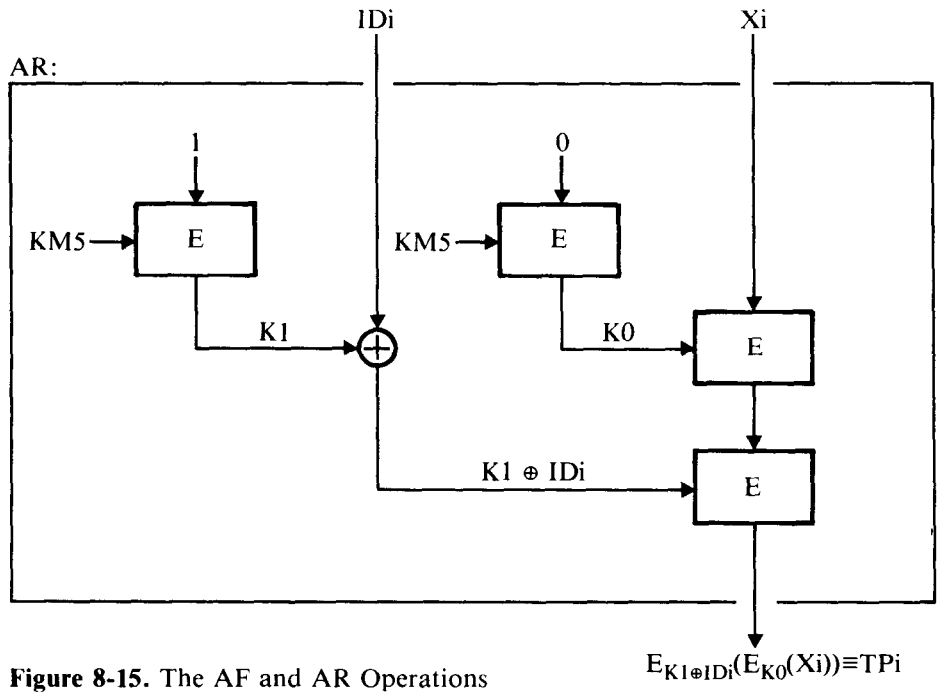


Figure 8-15. The AF and AR Operations

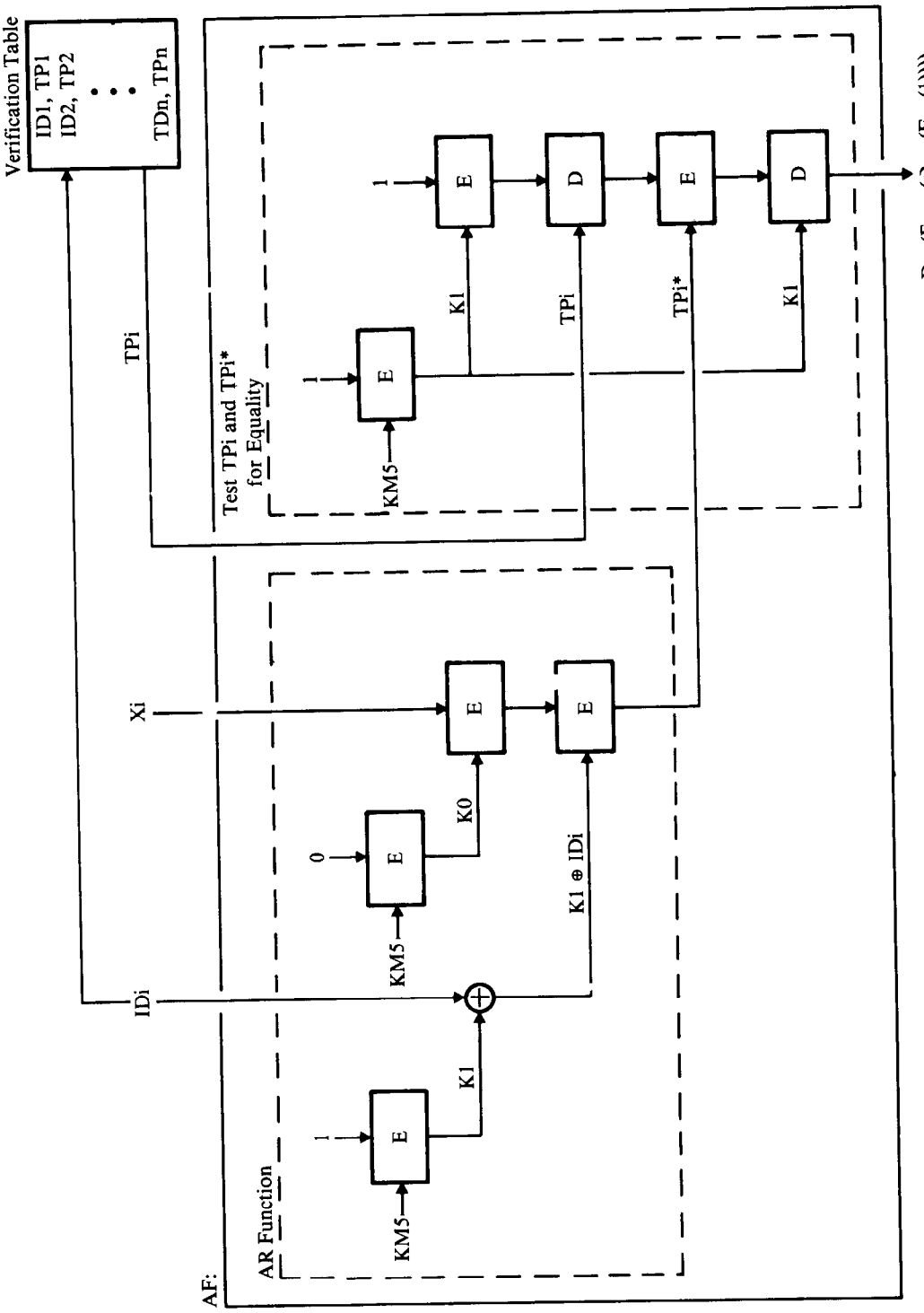


Figure 8-15 (cont'd). The AF and AR Operations

Rather than define K1 as a variant of the host master key, which would allow K1 ⊕ IDi to be manipulated to produce other variants of the host master key, K1 is produced via encipherment using a variant of the host master key. K0 (or the step of encipherment under K0) is introduced to prevent an opponent from directly controlling the input enciphered under K1 ⊕ IDi.

The reader will note that the steps involved in producing the quantity $E_{K1}(D_{TPi}(E_{TPi*}(D_{K1}(1))))$ are an alternate way to compare TPi and TPi* for equality. In effect, the comparison is implemented as a series of encipher and decipher operations.

REFERENCES

1. Wilkes, M. V., *Time-Sharing Computer Systems,* American Elsevier, New York, 1972.
2. Evans, A., Jr., Kantrowitz, W., and Weiss, E., "A User Authentication System Not Requiring Secrecy in the Computer," *Communications of the ACM,* **17**, No. 8, 437–442 (August 1974).
3. Pudy, G. B., "A High Security Log-in Procedure," *Communications of the ACM,* **17**, No. 8, 442–445 (August 1974).
4. Proposed Federal Standard 1026, *Telecommunications: Interoperability and Security Requirements for Use of the Data Encryption Standard in the Physical and Data Link Layers of Data Communications,* General Services Administration, Washington, D.C., Draft (January 21, 1982).
5. Jueneman, R. R., "Analysis of Certain Aspects of Output Feedback Mode," in *Advances in Cryptography: The Proceedings of Crypto 82,* edited by L. M. Adleman, D. L. Chaum, D. E. Denning, W. Diffie, S. T. Kent, R. L. Rivest, A. Shamir, Plenum Publishing Corp., New York (1983).
6. Jueneman, R. R., Matyas, S. M., and Meyer, C. H., "Authentication with Manipulation Detection Code," *Proceedings IEEE '83 Symposium on Security and Privacy,* Oakland, California (April 1983).
7. McPhee, W. S. "Operating System Integrity in OS/VS2," *IBM Systems Journal,* **13**, No. 3, 230–252 (1974).
8. Diffie, W. and Hellman, M., "New Directions in Cryptography," *IEEE Transactions on Information Theory,* IT-22, 644–654 (November 1976).
9. Lennon, R. E., Matyas, S. M., and Meyer, C. H., "Cryptographic Authentication of Time-Invariant Quantities," *IEEE Transactions on Communications, COM-29,* No. 6, 773–777 (June 1981).
10. Smid, M. E., *A Key Notarization System for Computer Networks,* National Bureau of Standards Special Publication 500–54, National Bureau of Standards, U.S. Department of Commerce, Washington, DC (October 1979).

Other Publications of Interest

11. Needham, R. M. and Schroder, M. D., "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM,* **21**, No. 12, 993–999 (December 1978).
12. *Guideline on User Authentication Techniques for Computer Network Access Control,* Federal Information Processing Standard (FIPS) Publication 83, National Bureau of Standards, U.S. Department of Commerce, Washington, DC (September 1980).
13. Feistel, H., "Cryptographic Coding for Data-Bank Privacy," IBM T. J. Watson Research Center, Yorktown Heights, NY, RC 2827 (March 1970).