Security Mechanisms in High-Level Network Protocols

VICTOR L. VOYDOCK AND STEPHEN T. KENT

Bolt, Beranek and Newman, Inc , Cambridge, Massachusetts 02238

The implications of adding security mechanisms to high-level network protocols operating in an open-system environment are analyzed. First the threats to security that may arise in such an environment are described, and then a set of goals for communications security measures is established. This is followed by a brief description of the two basic approaches to communications security, link-oriented measures and end-to-end measures, which concludes that end-to-end measures are more appropriate in an open-system environment. Next, relevant properties of data encryption—the fundamental technique on which all communications security mechanisms are based—are discussed. The remainder of the paper describes how end-to-end measures can be used to achieve each of the security goals previously established.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*security and protection*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*protocol architecture*; E.3 [**Data**]: Data Encryption—*Data Encryption Standard (DES)*; *public-key cryptosystems*

General Terms: Design, Security, Standardization

Additional Key Words and Phrases: Computer networks, wiretapping, authentication

INTRODUCTION

This paper analyzes the implications of adding security mechanisms to high-level[1] network protocols. For generality, we assume that these protocols operate in an environment consisting of an arbitrary collection of independently managed interconnected networks, each network consisting of an arbitrary number of heterogeneous hosts—an "open-system environment" in ISO terminology [International Organization for Standardization 1980].

There are several trends, which the widespread adoption of standard protocols will intensify, that emphasize the need to develop such security mechanisms. First, the increased use of networks to provide remote access to computer facilities, coupled with improved physical security measures at computer sites, makes attacking networks more attractive to an intruder. Second, the growing quantity and value of information made vulnerable by the breaching of network security make networks tempting targets. Third, computer systems connected by networks are likely to cooperate in various ways to provide resource sharing for a user community. As a result of this sharing, the security of information on a given host may become dependent on the security measures employed by the network and by

[1] The term *high-level protocol* is used in this paper to denote a protocol that exists at one of layers 4–7 of the ISO Reference Model of Open-System Interconnection [International Organization for Standardization 1980]. See Section 1.1.

V. L. Voydock's present address is Microcom, Inc., 1400A Providence Highway, Norwood, MA 02062.

CONTENTS

other hosts. Finally, the development of new network technologies facilitates certain kinds of attacks on communication systems; for example, it is easy for an intruder to monitor the transmissions of satellite and radio networks.

As discussed in Section 1.3, potential security violations can be divided into three distinct categories:

- unauthorized release of information,
- unauthorized modification of information,
- unauthorized denial of resource use.

The term *unauthorized*, used to describe the three categories of attacks, implies that the release, modification, or denial takes place contrary to some security policy. The *intruder* may be either a wiretapper[2] outside of the user community or an otherwise legitimate user of the network. Communication security techniques have traditionally been employed to counter attacks by the former type of intruder, while authentication and access-control techniques provide the finer granularity of protection required in the latter case. Both techniques must be used in conjunction with traditional physical, electromagnetic-emanation, procedural, and personnel security controls. This paper concentrates on the communication security aspects of network security.[3]

The paper begins by describing the threats to security that arise in an open-system environment and goes on to establish a set of goals for communication security measures. This is followed by a brief description of the two basic approaches to communication security, link-oriented measures and end-to-end measures, with the conclusion that end-to-end measures are more appropriate in an open-system environment. It goes on to discuss relevant properties of data encryption, the fundamental technique on which all communication security measures are based. The remainder of the paper describes how end-to-end measures can be used to achieve each of the security goals previously established.

1. THREATS TO NETWORK SECURITY

This section begins with a description of the architectural model within which our analysis of security measures takes place. This is followed by a description of the open-system environment in which the protocols are assumed to be operating. The

[2] The term *wiretapper* is used here in a broad sense to denote an intruder whose attacks do not involve subversion of the host computers of the network. Section 1.3 describes the wide range of ways in which such an intruder can attack network security.

[3] In particular, the paper concentrates on providing security for real-time communication. Applications involving one-way communication, such as secure file storage and self-authenticating messages, involve protocols that are outside the scope of this paper [Needham and Schroeder 1978].

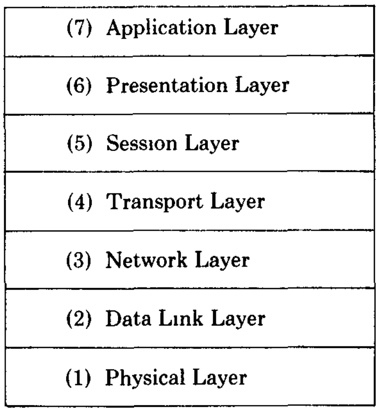

Figure 1. The ISO reference model.

section concludes with an analysis of the kind of security threats that can arise in such an environment. Subsequent sections discuss measures to counteract these threats.

1.1 The ISO Reference Model

Our analysis of security takes place within the context of the ISO Reference Model of Open-System Interconnection [International Organization for Standardization 1980]. In this model, the data communication path is logically composed of an ordered set of subsystems, called *layers*, through which application programs (entities at the highest layer) communicate. Figure 1 depicts the seven layers of the model, and the following paragraphs introduce some terminology from the model that is used throughout this paper.

Each of the seven layers of the model is composed of *protocol entities*. Entities that exist at the same layer are termed *peer entities*. Peer entities communicate with each other using a peer-to-peer protocol. They receive data from the next higher layer, attach the appropriate protocol control information to those data, and then pass the result to the next lower layer. For example, to communicate with a peer entity, a session entity (layer 5) passes its control information and data to a transport entity (layer 4). The transport entity adds its own control information to the session data and passes this new construct as data to a network entity (layer 3). At the destination system, a transport entity will receive these data from its network entity, remove its own control information, and forward the remaining data to the receiving session peer entity.

To be more formal, protocol entities at layer $N + 1$ are referred to as $(N + 1)$-entities. An $(N + 1)$-entity communicates with one of its peers using a peer-to-peer protocol; the units of data with which they communicate are called $(N + 1)$-protocol data units (PDUs). That is, an $(N + 1)$-layer peer-to-peer protocol is implemented through the exchange of $(N + 1)$-PDUs between $(N + 1)$-entities.

To transmit an $(N + 1)$-PDU to one of its peers, an $(N + 1)$-entity must use the services of the N-layer. The units of data that the N-layer will transmit for the $(N + 1)$-entities are known as N-service data units (SDUs). An $(N + 1)$-PDU presented to the N layer becomes an N-SDU.

There are two basic types of services provided by the N layer: *connection service* and *connectionless service*. If the connection service is used, the N-layer establishes a conduit (called an *N-connection*) between the two $(N + 1)$-entities, through which N-SDUs (i.e., $(N + 1)$-PDUs) are exchanged. Connectionless service allows an $(N + 1)$-entity to send an N-SDU to another $(N + 1)$-entity without prior establishment of an N-connection. The N-SDU presented to the N-layer will be converted into one or more N-PDUs. Each N-PDU consists of $(N + 1)$-layer data, together with N-layer protocol control information.

1.2 Environment Models

This section presents a model of the open-system environment in which the protocols are assumed to operate, and a model of an end-to-end data path through that environment. This model provides a context for the subsequent discussion and categorization of security threats. In this paper we use the terms *network* and *open-system environment* synonymously.

1.2 1 The Network Model

The basic component of the network is the communication subnet which functions as

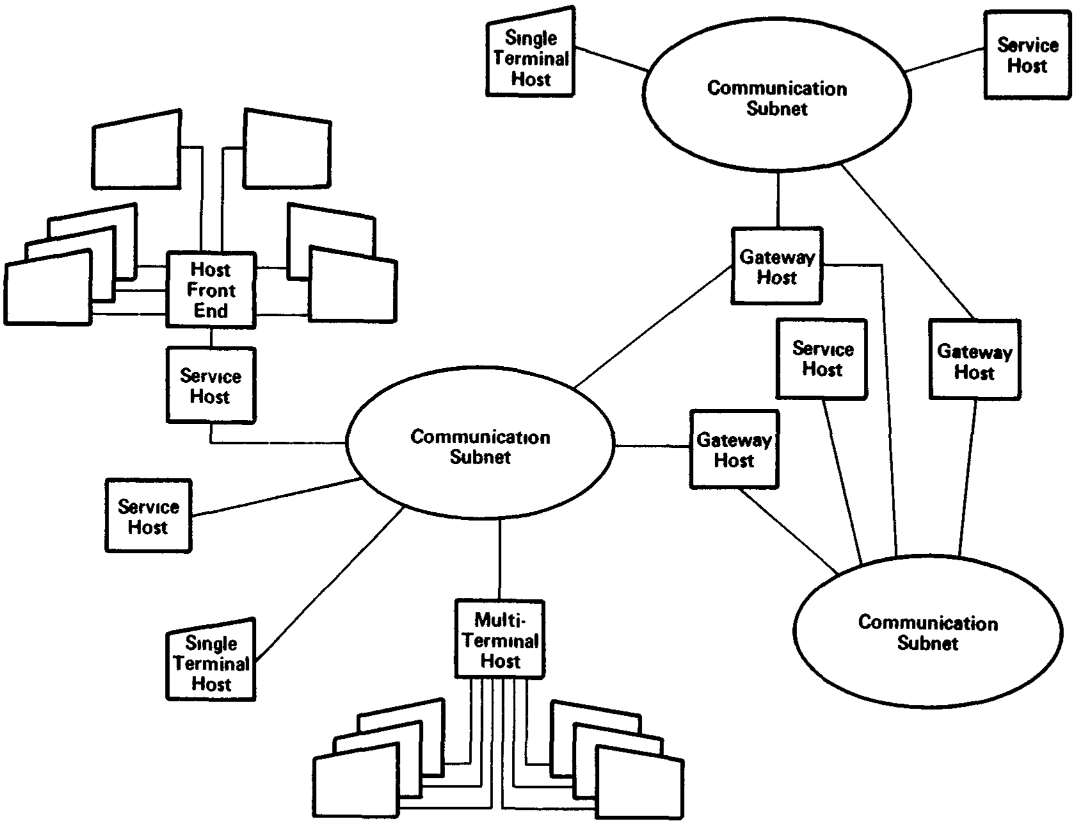

Figure 2. The network model.

a transmission medium. For example, in an Ethernet [Metcalfe and Boggs 1976] this transmission medium is a coaxial cable, while in the Distributed Computer System (DCS) ring network [Farber and Larson 1972] it consists of a twisted pair with network interface repeaters at intervals along the ring. In the ARPANET, the transmission medium consists of packet switches (TIPs or IMPs) connected by leased phone lines or satellite channels. Whatever its implementation, the communication subnet serves to connect the hosts on a network. This model distinguishes four types of hosts: single-terminal, multiterminal, service, and gateway hosts. Figure 2 illustrates the network model discussed in this section.

A single-terminal host connects an individual user terminal to the network and might be implemented by a microprocessor contained within the terminal. A multiterminal host (also known as a PAD, in CCITT terminology) connects several terminals to the network and can be implemented by a microprocessor or by a minicomputer, depending on the number of terminals being served and the level of service provided. The terminals may be connected to a multiterminal host by direct wire or by dial-up through the telephone network. The distinction between a single-terminal and multiterminal host is important because, for the latter, one must consider the issue of data-stream mixing from the various terminals served.

The service hosts on the net may consist of both conventional computers, providing general information-processing functions, and special-purpose computers, providing specialized functions. These service hosts may have terminals connected to them by direct wire or by telephone dial-up, as well as by the communication network.

Gateway hosts [Cerf and Kahn 1974; Cerf and Kirstein 1978] interconnect two or more networks. The services provided by a gateway include routing PDUs from one network to another, fragmenting PDUs that are too large for the destination network, and embedding internetwork PDUs in local network PDU formats. There may be more than one gateway path between two networks.

Terminals form the final component of the network model. Terminals are important as distinct entities in the network model because they provide the most direct interface to the human users of the network. Because a user is frequently the ultimate source or destination of the information to be protected, the terminal provides the earliest point in the system at which one can begin this job of protection.

Having introduced the network model, we now examine several types of existing or planned networks to see how well they are represented by the model. Many packet-switched networks follow the model quite closely. For example, in the ARPANET the communication subnet consists of packet switches (IMPs) connected by leased telephone lines or satellite channels. Terminal access to the network is provided both through multiterminal hosts (TACs), to which terminals may be connected by direct or dialed lines, and through service hosts. The service hosts themselves may be accessed from outside the network through terminals attached to direct or dialed lines. Several other networks are connected to the ARPANET through gateways of varying sophistication, for example, the ALOHA system [Abramson 1970] and the Xerox PARC Ethernet. Other networks that correspond with this model include Cyclades [Pouzin 1973], Telenet [Telenet Communications Corporation 1975], and several military networks now under development, such as the Defense Data Network [Defense Communications Agency 1982].

The salient features (with respect to security) of several packet-broadcast networks can also be represented by the simple network model presented in Section 1.2. Some examples of these networks include the Ethernet, the ARPA Packet Radio Network [Kahn 1975], and the DCS Ring Network. The Packet Radio Network uses both service hosts and multiple-terminal hosts, and features an elaborate communication subnet that tracks the mobile packet radio terminals and performs routing functions. Hosts on the Ethernet are primarily personal computers, which can be viewed as extremely sophisticated single-terminal hosts, although some general-service hosts are also attached. The DCS Ring Network connects minicomputer hosts, some acting purely as network-accessible service hosts and others acting as conventional hosts with terminals directly attached. Thus the network model described above is applicable to a wide range of packet-switched and packet-broadcast networks.

1 2.2 The Association Model

The term *association* is used to refer to an (potentially bidirectional) end-to-end data path through the network modeled in Section 1.2. In the terminology introduced in Section 1.1, an association refers either to a connection at any of layers 4–7 of the ISO Reference Model, or to the exchange of messages via the connectionless data transmission service at any of those layers. Thus, an *N-association* is the data path over which $(N + 1)$-entities communicate. The information transmitted on an N-association consists of a series of N-PDUs, each comprised of $(N + 1)$-layer data packaged with N-layer protocol control information.

The model assumes that both ends of the association terminate in secure areas, but that the remainder of the association may be subject to physical attack. A terminal that forms one end of the association may, at different times, be used by various individuals at different authorization levels. The hosts on which the communicating protocol entities reside may provide services to a diverse user community, not all of whose members employ communication security measures. An intruder is represented by a computer under hostile control, situated in the communication path between the ends of the association. Thus all PDUs transmitted on the association must pass through the intruder. The association model is depicted in Figure 3.

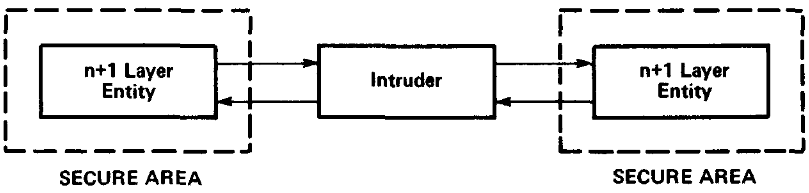

Figure 3. The association model.

For the most part, one can discuss security threats and countermeasures in the context of the association model without regard to the differences between connection and connectionless service. Consequently, we discuss their differing characteristics only when they have impact on network security measures.

1.3 Classification of Attacks

This section discusses and classifies the kinds of security attacks that can be mounted by an intruder in the context of the models presented in Section 1.2. As mentioned before, potential security violations can be divided into three distinct categories:

- unauthorized release of information,
- unauthorized modification of information,
- unauthorized denial of use of resources.

Attacks that cause information release are known as *passive attacks*, while those that cause information modification, or denial of resource use are known as *active attacks*. We assume that the intruder can position himself at a point in the network through which all information of interest to him must pass. We also assume that he can mount both active and passive attacks.

1.3.1 Potential Loci for Intruder Attacks

To illustrate the variety of places at which an intruder attack can take place, Figure 4 depicts the physical path that information might traverse, where one end of the association is a user at a terminal and the other is an application program in a service host. Here, information entered on the terminal by the user must first pass along a communication link to the multiterminal host. From there it passes along another link connecting the host to a packet switch in the communications subnet. It continues to pass from link to link in the subnet until it arrives at the packet switch to which the destination service host is connected. In this case, the service host is not directly connected to the subnet. Instead, information first passes to a host front-end computer that implements the lower level network protocols, and then to the service host itself.

An intruder attack can take place at any of the links described above. In addition, the intruder can attempt to subvert any of the computers in the path, either by modifying their hardware or software, or by monitoring their electromagnetic emanations. The links themselves can be ordinary telephone links, microwave links, or satellite channels. Telephone wires can be attacked using either invasive taps or inductive devices that monitor electromagnetic emanations. Invasive taps allow both active and passive attacks to be easily performed, whereas inductive taps are most suitable for passive attacks. Microwave and satellite transmissions can be intercepted with little risk to the intruder. This is especially true of satellite links, since their signals are accessible over a wide geographic area, and since satellite ground station receivers are becoming increasingly inexpensive. Active attacks on microwave and satellite links are also possible, although more difficult since they require expensive equipment and a high degree of technical sophistication.

As another example, consider Figure 5 which depicts an attack that can occur in an internetworking environment. In this case, the intruder has subverted a gateway in some intermediate network that provides the only communication path between the two end protocol entities. Here, even though the source (A) and destination (D)

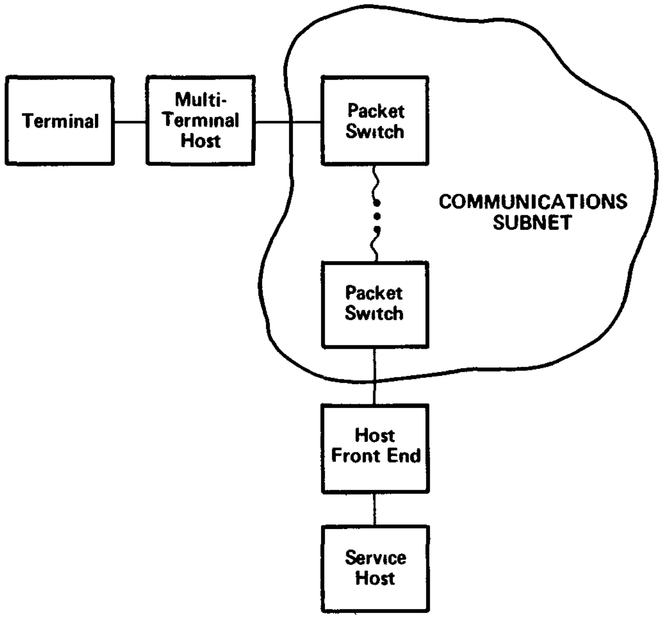

Figure 4. Typical physical path.

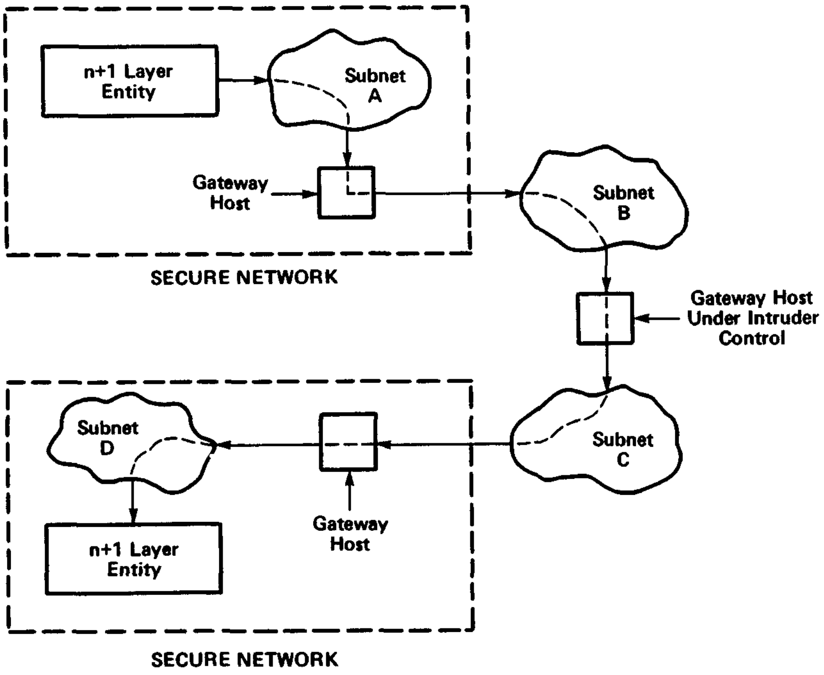

Figure 5. Intruder in internetwork environment.

networks are secure, the intruder can attack the association as it passes through the gateway connecting networks B and C.

From the above discussion it should be clear that the number of places in which an intruder can attack network security is very large. The following sections discuss passive and active attacks in turn.

1.3.2 Passive Attacks

In a passive attack, the intruder merely observes PDUs passing on an association without interfering with their flow. Such intruder observation of the $(N + 1)$-layer data in an N-layer PDU is termed *release of message contents* and constitutes the most fundamental type of passive attack. Even if the data are not intelligible to him, the intruder can observe the protocol control information portion of the PDU and thus learn the location and identities of the communicating protocol entities. Finally, the intruder can examine the lengths of PDUs and their frequency of transmission to learn the nature of the data being exchanged. These latter two types of passive attacks are usually referred to as *traffic analysis* or *violations of transmission security*.[4]

1.3.3 Active Attacks

The intruder can also mount active attacks, performing a variety of processing on PDUs passing on the association. These PDUs can be selectively modified, deleted, delayed, reordered, duplicated, and inserted into the association at a later point in time, or be allowed to pass through unaffected. Bogus PDUs can be synthesized and inserted into the association.

While all active attacks involve some combination of the methods listed in the previous paragraph, the countermeasures employed against them vary with the form of the attack. For this reason, it is useful to subdivide active attacks into the following three categories [Kent 1976]:

- message-stream modification,
- denial of message service,
- spurious association initiation.

These categories are separately discussed in the paragraphs below.

Message-stream modification includes attacks on the *authenticity, integrity,* and *ordering* of the PDUs passing on the association. In the context of the model, authenticity means that the source of a PDU can be reliably determined (i.e., that a received PDU was transmitted by the protocol entity at the other end of the association). Integrity means that a PDU has not been modified en route, and ordering means that a PDU can be properly located in the stream of information being transmitted.

Attacks on authenticity can be made by modifying the protocol control information in PDUs so that they are sent to the wrong destination, or by inserting bogus PDUs (either synthesized or saved from a previous association) into an association. Attacks on integrity, in turn, can be effected by modifying the data portion of PDUs, whereas attacks on ordering can be effected by deleting PDUs or modifying sequencing information in the protocol control portion of PDUs. Although protection against message-stream-modification attacks is often provided by communication protocols for reliability purposes, in this context it must be provided to thwart malicious attacks rather than simply to protect against benign component failures.

Denial of message service, the second category of active attacks, comprises attacks in which the intruder either discards all PDUs passing on an association or, in a less drastic action, delays all PDUs going in one or both directions.[5] The subtle dif-

[4] Although traffic analysis is usually seen as a means of inferring information by observing legitimate message traffic, this form of attack can also be used in conjunction with a *Trojan Horse* [Anderson 1972] program operating in a user process at a service host. The Trojan Horse program could perform some legitimate function within the process while clandestinely modulating message destination, length, or frequency of transmission, thus using the association as a covert channel to transmit to the intruder data legitimately accessible by the process [Karger 1977; Kent 1976]. In an interactive communication environment, if the bandwidth of the communication network is high relative to the speed of user terminal equipment, a Trojan Horse program could modulate message length and frequency completely unbeknownst to the unwilling, legitimate user.

[5] Depending on the communication medium employed, measures can be taken to make it extremely difficult

ference between message-stream-modification attacks and denial-of-message-service attacks is a function both of the degree of the attack and of the state of the association. For example, message-stream-modification countermeasures can detect transient denial-of-service attacks. But, if the association is quiescent (i.e., no PDUs are outstanding in either direction), a protocol entity at one end of the association may have no way of determining when the next PDU should arrive from its correspondent peer entity. In this state it would thus be unable to detect a denial-of-service attack that completely cut off the flow of incoming PDUs. To handle attacks of this sort, additional countermeasures are necessary.

Spurious association initiation, the third category of active attacks, comprises attacks in which the intruder either "plays back" a recording of a previous legitimate association initiation sequence or attempts to establish an association under a false identity. To counteract the "play-back" kind of attack, association initiation must include a mechanism that verifies the time integrity of the association (i.e., determines that the association initiation attempt is being made in real time). To counteract the "false identity" kind of attack, an association must be initiated in a fashion that supports secure identification of the protocol entities at each end. Although verification of identity is a complex issue that interacts with authentication and security controls outside of the association model, a portion of the identification problem must be dealt with within the model. Although this category of attack is similar in nature to message-stream-modification attacks, the context of association initiation requires the use of additional countermeasures.

for an intruder to severely disrupt communications, for example, spread spectrum techniques in radio networks. Such "tamper-free" techniques are of interest in environments like military communication networks [Baran 1964], where disruption of communication cannot be tolerated. In many ways, such techniques are more akin to reliability measures than to the communication security measures discussed here. In addition, economic or technical considerations prohibit the use of such techniques in many communication environments. Thus, in this paper, we make the generally applicable assumption that an intruder cannot be prevented from disrupting communication services in the network.

1.4 Communication Security Goals

As seen below, active and passive attacks are in some sense duals. That is, although message-stream modification, denial of message service, and spurious association initiation attacks cannot be prevented, they can be reliably detected. Conversely, release of message contents and traffic analysis attacks usually cannot be detected,[6] but they can be effectively prevented.

Mindful of these limitations, we have established the following five goals for designing mechanisms to provide communications security:

- prevention of release of message contents,
- prevention of traffic analysis,
- detection of message-stream modification,
- detection of denial of message service,
- detection of spurious association initiation.

In the remaining sections of this paper, we discuss measures for achieving these communication security goals. We present the measures within the context of the association model, without regard for the detailed properties of any particular protocol layer.

2. APPROACHES TO COMMUNICATION SECURITY

There are two basic approaches to communication security: *link-oriented* security measures and *end-to-end* security measures. The former provide security by protecting message traffic independently on each communication link, while the latter provide uniform protection for each mes-

[6] In some communication media, techniques such as the careful measurement of inductance changes over circuits can be employed to detect passive attacks. However, recent trends in network technology involve media in which such detection cannot be carried out. Moreover, in most environments, preventing the release of information in the face of a passive attack is more important than detecting such an attack. Thus no discussion of detection of passive attacks is included in this paper.

sage all the way from its source to its destination. These two approaches differ not only in their internal implementation characteristics, but also in the nature of the security they provide. In the following sections we discuss the characteristics of each of these approaches and the implications of using them.

2.1 Link-Oriented Measures

Link-oriented protection measures provide security for information passing over an individual communication link between two nodes, regardless of the ultimate source and destination of that information. As discussed in Section 1.3, information passing on an association may traverse a considerable number of communication links. Each of these links corresponds to a Data Link Layer association in the ISO Reference Model. These links may be telephone lines, microwave links, or satellite channels. In many cases, the links will be physically unprotected and thus subject to attack.

In a network employing link-oriented measures, encryption is performed independently on each communication link. A different encryption key is often used for each link, so that subversion of one link need not necessarily result in release of information transmitted on other links. To encrypt the information, stream ciphers are generally employed. Since information is not processed as it passes on a link, both the protocol control information and the data in PDUs can be enciphered.[7] This masks origin–destination patterns. If a continuous stream of ciphertext bits is maintained between nodes, PDU frequency and length patterns can be masked as well. In this case, all forms of traffic analysis are completely prevented. Using this technique does not degrade the effective bandwidth of the network because it does not usually require transmission of any additional data; it does, however, entail continuous key-stream generation at each node.

Since information is enciphered only on the links and not within the nodes they connect, the nodes themselves must be secure. Although the origin and destination nodes of the network (i.e., the hosts) are assumed to be physically secure (see Section 1.3.2), link encryption requires that all intermediate nodes (packet switches and gateways) be physically secure as well. Not only must they be physically secure, but their hardware and software components must be certified to isolate the information on each of the associations passing through them.

Subverting of one of the intermediate nodes exposes all of the message traffic passing through that node, despite any physical security precautions still in effect at the source and destination nodes. In a network such as the ARPANET, where adaptive routing strategies are employed, a subverted intermediate node could cause PDUs to be routed through it, regardless (with some exceptions) of their source and destination hosts. Thus, one problem with link-oriented protection is that subversion of a single intermediate node can expose substantial amounts of message traffic.

Another serious problem is the cost of maintaining the security of the nodes. In addition to the one-time expense of providing encryption hardware and a secure physical environment for each node, there are a number of ongoing expenses whose total cost may well exceed the one-time outlays. These include the cost of the employees to protect the physical security of the nodes and the cost of the key distribution process—keys need to be changed frequently, and this can be expensive in a network with a large number of nodes. Other ongoing expenses include the cost of screening the security personnel and the cost of frequent audits to ensure that security policies and procedures are being carried out correctly. In addition, because the operators of a public network providing link encryption could well be liable for damages resulting from security violations, the cost of insuring against such losses would also need to be borne by the network users.

[7] Some link-oriented encryption devices selectively encrypt portions of PDUs and send other portions in the clear. This, however, is made necessary by the specific logical organization of those devices. It is not an intrinsic requirement of link-oriented encryption.

To make matters worse, it is difficult to apportion the costs of link-oriented protection fairly. While the cost of providing such protection on terminal-to-host and host-to-communication subnet links can be borne by the directly affected parties, the overall cost of link-oriented measures within the communication subnet itself must be amortized over all of its users, whether or not they feel the need for such protection. In addition, many network users may not want to rely, for security services, on the authorities controlling the communications subnet. This is especially true in an open-system environment where information exchanged on an association may traverse multiple networks controlled by diverse organizations. For all of these reasons, link-oriented measures do not appear to be appropriate as the basis for communication security in an open-system environment.

2.2 End-to-End Measures

Link-oriented measures model a network as a collection of nodes joined by communication links, each of which can be independently protected. End-to-end measures, on the other hand, model a network as a medium for transporting PDUs in a secure fashion from source to destination. In keeping with this perspective, end-to-end security measures protect PDUs in transit between source and destination nodes in such a way that subversion of any of their communication links does not violate security.

There is some flexibility in defining the points at which end-to-end security measures are implemented: from host to host, from terminal to service host or process, from process to process. By extending the domain of end-to-end security measures, one can protect more of the path between communicating protocol entities. However, as their domain is extended, the range of hardware and software that must interface with them increases.

Link-oriented security measures can be implemented so that they are almost completely invisible to network users. End-to-end security measures usually extend beyond the communication subnet and thus require a greater degree of standardization in the protocols employed by those users. But, since protocol standardization is already coming about for technical, economic, and political reasons, this is not a serious impediment to the adoption of end-to-end security measures in an open-system environment.

A major advantage of end-to-end security measures is that an individual user or host can elect to employ them without affecting other users and hosts; thus the cost of employing such measures can be more accurately apportioned. Moreover, these measures can be employed not only in packet-switched networks, but in packet-broadcast networks where link-oriented measures are often not applicable. Finally, end-to-end measures are more naturally suited to users' perceptions of their security requirements. This stems from the fact that they rely on the security of equipment only at the source and destination of a association, while link-oriented measures require that all nodes (packet switches and gateways) in the entire open-system environment also be secure.

2.3 Association-Oriented Measures

A communication network can also be viewed as providing a medium for establishing associations between protocol entities. This view suggests that security services be association oriented or, in other words, that each association be protected individually. Thus, association-oriented security measures constitute a refinement of end-to-end measures. As with end-to-end security measures, the implementer has considerable flexibility in choosing the endpoints of the association for security purposes.

Association-oriented measures not only protect that portion of a communication path that lies between the security-defined ends of the association, but also significantly reduce the probability of undetected "cross talk," whether induced by hardware or software, over that interval.

Having shown the superiority of end-to-end over link-oriented measures for the open-system environment, we concentrate in the remainder of this paper on end-to-end measures, especially association-oriented measures, for achieving the five security goals first presented in Section 1.4:

- prevention of release of message contents,
- *prevention of traffic analysis,*
- detection of message-stream modification,
- *detection of denial of message service,*
- detection of spurious association initiation.

We present these measures within the context of the association model without regard for the detailed properties of any particular protocol layer.[8] Our discussion centers on conventional cryptosystems as exemplified by the Data Encryption Standard (DES) of the National Bureau of Standards [National Bureau of Standards, 1977, 1980a, 1980b]. As a Federal Information Processing Standard, the DES forms the basis for cryptographic communication security measures applied to nonclassified government information; it appears that it is becoming a de facto industry standard as well. The impact of the use of public-key cryptosystems on the various security measures is also briefly sketched.

In the discussion that follows, it must be remembered that the measures described provide security only in a probabilistic sense. There are two reasons for this. First, all of the measures are based on the inability of the intruder to subvert the encryption algorithm employed. Second, even if the algorithm is not subverted, the last three types of measures are not infallible. They simply ensure that active attacks will be detected with high probability.

3. DATA ENCRYPTION

Historically, encryption has been extensively employed as a countermeasure to passive attacks [Kahn 1967]. It can also serve as a foundation on which to construct countermeasures to active attacks [Branstad 1975; Feistel et al. 1975; Kent 1976, 1977]. The design and analysis of encryption algorithms are beyond the scope of this paper, but a familiarity with some characteristics of such algorithms is essential to understanding the countermeasures discussed. Thus this section describes some general characteristics of conventional ciphers and the recently developed class of public-key ciphers.

3.1 Basic Concepts

A *cipher* is an algorithmic transformation performed on a symbol-by-symbol basis on any data. The terms *encipherment* and *encryption* refer synonymously to the application of a cipher to data. An *encryption algorithm* is any algorithm that implements a cipher. The input to an encryption algorithm is referred to as *cleartext* or *plaintext*, while the output from the algorithm is called *ciphertext*. The transformation performed on the cleartext to encipher it is controlled by a *key*. For use in the communication context, the encryption algorithm must be *invertible*; that is, there must be a matching decryption algorithm that reverses the encryption transformation when presented with the appropriate key.

In *conventional ciphers*, the key used to decipher a message is the same as that used to encipher it. Such a key must be kept secret, known only to authorized users. Authorized users can use the key both to encrypt their own messages, and to decrypt messages that others have encrypted using it. Figure 6 illustrates these aspects of a conventional cipher.

In contrast, in a *public-key cipher*, the ability to encipher messages under a given key is separated from the ability to decipher those messages. This is accomplished by using pairs of keys (E, D). These keys define a pair of transformations, each of which is the inverse of the other, and neither of which is derivable from the other. Each user possesses such a key pair. One key (E) is made public, for use in enciphering messages for that user, while the corresponding key (D) is kept secret, for use in deciphering messages sent to the user under the public key.

Since anyone can transmit a message to a user (i) under that user's public key (E_i), some additional mechanism is needed to

[8] A description of how these measures can be integrated into the proposed NBS transport layer protocol is presented in Voydock and Kent 1981.

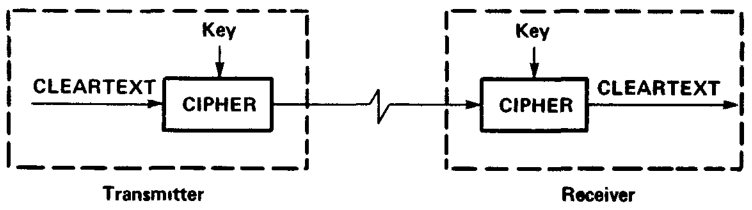

Figure 6. A conventional cipher.

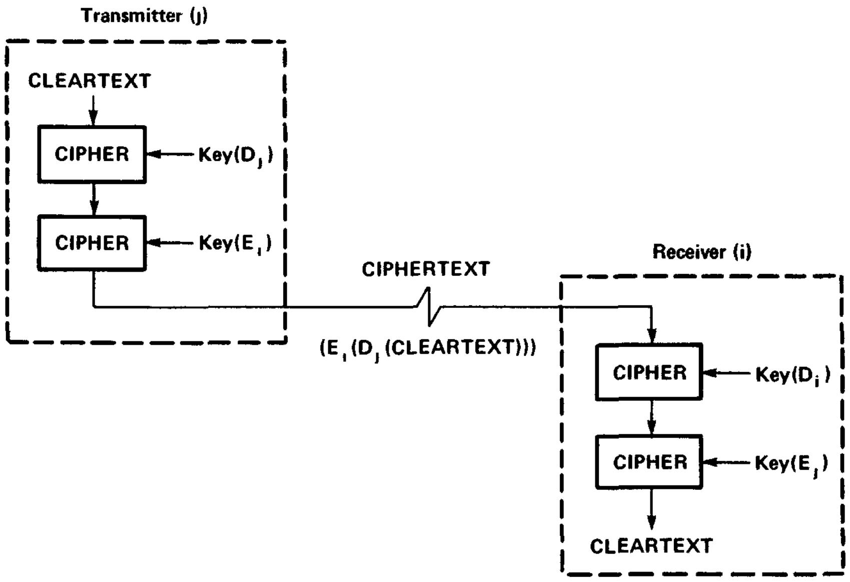

Figure 7. A public-key cipher.

securely identify the sender. Identification is accomplished by having the sender (j) encrypt the message under his secret key (D_j), then under the public key of the intended receiver (E_i). The receiver can then strip off the outer layer of encryption using his secret key (D_i), and complete the deciphering using the public key of the sender (E_j). This is illustrated in Figure 7.

One of the supposed advantages of a public-key cryptosystem is that public keys may be freely distributed without concern for secrecy. But, as seen in Section 8, the need for authentication in the distribution of public keys in an open-system environment results in there being few differences between public-key and conventional-key distribution mechanisms.

3.2 Attacks on Cryptosystems

A cipher that can resist any analytical attack, regardless of the computational power and time available to the attacker, is referred to as being *unconditionally* or *theoretically* secure. A cipher that is not unconditionally secure is considered *computationally* or *practically* secure, if the computational cost involved in breaking it exceeds the value of the information gained [Diffie and Hellman 1976]. There are three increasingly powerful classes of cryptanalytic attack: *ciphertext-only, known-plaintext,* and *chosen-plaintext.*

In a ciphertext-only attack, the cryptanalyst employs only a knowledge of the statistical properties of the language in use, such as the relative frequency of letter occurrence in a natural language and a knowledge of "probable" words, such as standard salutations in letters or common variable names in program text. More powerful than ciphertext-only, a known-plaintext attack can be mounted whenever matching plaintext and ciphertext are available. This kind of attack is easily mounted in a network environment, since software systems often generate well-known fixed-format messages; examples of these include the stylized log-in greetings and system-wide messages issued by timesharing systems. Finally, in a chosen-plaintext attack, the cryptanalyst is able to select plaintext and then obtain the matching ciphertext. This form of attack can be mounted in a variety of ways, for example, through transmission of "network mail," and thus constitutes a very real threat.[9] Hence, any cipher proposed for use in an open-system environment must be resistant to known- and chosen-plaintext attacks, as well as to ciphertext-only attacks.

3.3 Major Encryption Techniques

Two major classes of encryption techniques have been employed in modern nonvoice telecommunications and digital computer applications: *block ciphers* and *stream ciphers*. The former method enciphers fixed-sized blocks of bits under the control of a key that is often approximately the same size as the blocks being encrypted. The latter method performs bit-by-bit transformations on cleartext under the control of a stream of key bits, usually using some easily reversible operation, such as addition modulo 2. In the following sections we discuss each of these techniques in turn.

3.3.1 Block Ciphers

Block ciphers transform entire blocks of bits under the control of a key. A block cipher maps the space of cleartext blocks into the space of ciphertext blocks. If the block size is N bits, then the size of the cleartext space (the range of cleartext block values) and the size of the ciphertext space (the range of ciphertext block values) are both 2^N. Since each ciphertext block must be unambiguously decipherable, the mapping must be one to one; since the sizes of the spaces are equal, this means it will also be onto. Thus, a block cipher defines a collection of permutations on the set of N-bit blocks; the key chosen determines which permutation is used. In practice, this collection is only a subset of the $(2^N)!$ possible permutations.

Block ciphers are equivalent to classical simple substitution ciphers. This equivalence is apparent for small block sizes (e.g., 8 bits). In this case, each block corresponds to an individual character from a small alphabet. Such a cipher is known to be extremely weak, not because of its structure, but because of the size of the blocks used. The cipher can be analyzed by comparing the frequency distribution of individual blocks with the known frequency distribution of characters in large samples of cleartext. To overcome this drawback, the designer can increase the block size and use mixing transformations to conceal intrablock frequency characteristics. The resulting cryptographic system can be computationally secure [Feistel et al. 1975].

A problem with block ciphers is that the length of a message does not generally correspond to the block size of the cipher, and some means of resolving this mismatch must be employed. A common solution is to fragment the message into as many block-sized pieces as required, padding it to occupy an integral number of blocks. Unfortunately, this padding wastes bandwidth (half the block size on the average) on each message.

3.3.2 Stream Ciphers

Stream ciphers can operate on the stream of cleartext in real time, enciphering each quantum of cleartext as it is generated by combining it with a quantum from the key stream. The size of the quanta processed varies with the particular cipher employed; common sizes are 1 and 8 bits. Some cryptosystems allow the user to specify the quantum size (see Section 3.4). If the quantum size is chosen appropriately, stream

[9] Public-key ciphers are easily subjected to such attacks because the public keys are generally available.

ciphers can provide a key stream that is exactly matched to the length of the message, thus avoiding the problems associated with padding cleartext to match block sizes.

A stream cipher whose key stream is a random bit string, each of whose bits is used only once, is known as a *Vernam cipher*. That is, in a Vernam cipher, the key stream is as long as the combined length of all messages ever encrypted with it. Such ciphers have been proven to be unconditionally secure [Shannon 1949; Kahn, 1967]. Unfortunately, the volume of communication traffic in a computer network environment makes it infeasible to provide each user with a sufficient quantity of key streams. For this reason, most stream ciphers utilize pseudorandom key streams with very long periods. Unlike Vernam ciphers, such stream ciphers are susceptible to cryptanalysis [Meyer and Tuchman 1972].

Various techniques may be used with stream ciphers to generate the key stream. Stream ciphers in which the key stream is a function of the cleartext, of the ciphertext, or of the key stream itself are known as *autokey* ciphers. The various types of autokey ciphers have differing properties.

In *key autokey* (KAK) ciphers, the key stream is completely independent of the cleartext and ciphertext streams. Consequently, changes to individual bits in the ciphertext do not propagate to other portions of the ciphertext stream. This is in one sense an advantage, since transmission errors that alter the value of bits in the ciphertext do not affect the ability of the receiver to decipher subsequent bits correctly. However, this very lack of error propagation allows an intruder to make predictable changes to deciphered cleartext and thus hampers development of mechanisms that detect message-stream modification attacks. For this reason, KAK ciphers are not suitable as a basis for communication security for most applications in an open-system environment. This is further discussed in Section 6.2.

To achieve interbit dependence and thus ensure desirable error propagation properties, the key stream must be a function of either the cleartext or ciphertext stream. *Ciphertext autokey* (CTAK) ciphers are of particular interest. In such ciphers, as the name implies, transmitted ciphertext is used as input for key-stream generation. These ciphers are self-synchronizing; that is, they propagate errors but resume correct operation after some fixed number of unaffected ciphertext bits are received [Savage 1967].

3.4 The Data Encryption Standard (DES)

The DES has several modes of operation, allowing it to be used as either a block or a stream cipher [National Bureau of Standards 1980a]. This section briefly describes each of these modes.

3.4.1 ECB Mode

In its most fundamental mode the DES is a block cipher operating on 64-bit blocks, using a 56-bit key. This mode is known as the *electronic code book* (ECB) mode in an analogy to conventional code books. Each key parameterizes the cipher, defining a permutation on the space of 64-bit blocks. In the ECB mode, a message is fragmented into block-sized pieces and padded to occupy an integral number of blocks, if necessary. Each block is then independently enciphered.

Each bit in a ciphertext block is a function of each bit of the key and of each bit of the cleartext block from which it was generated. A change of as little as 1 bit in either the key or the cleartext results in ciphertext in which each bit is changed with approximately equal probability. Conversely, a change in 1 bit of either the key or ciphertext will produce changes in an average of 50 percent of the bits of deciphered cleartext. Although this error propagation is extensive, it is strictly limited to the block in which the error occurs—decryption of other blocks is unaffected. Thus, the cryptographic synchrony required for correct deciphering of messages is achieved when both sender and receiver employ the same key and blocks are correctly delimited. The use of the ECB mode is illustrated in Figure 8.

3.4.2 CBC Mode

In the ECB mode, each block of ciphertext is independent of all other ciphertext; that

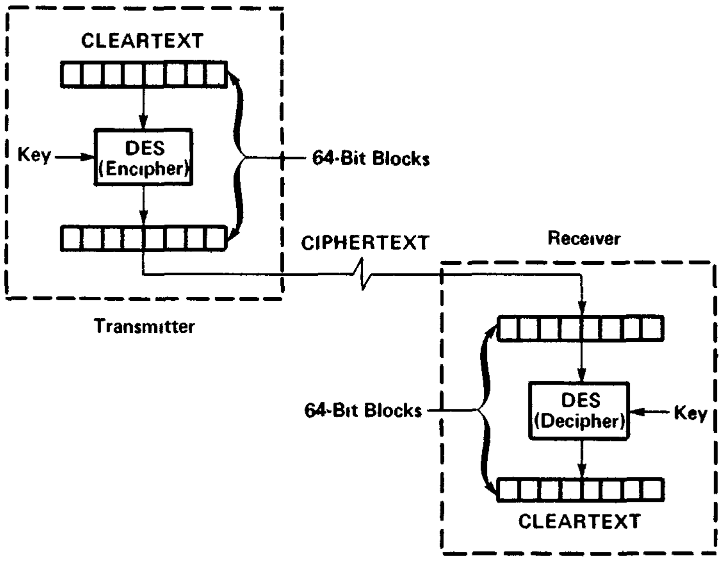

Figure 8. The ECB mode of the DES.

is, it is a function only of the key and the cleartext block that produced it. As discussed in Section 6, this independence limits the types of modification detection codes that can be employed. Also, note that identical cleartext blocks produce identical ciphertext blocks. This exposure of block-size data patterns that fall on block boundaries is often unacceptable. A second DES mode, known as the *cipher block chaining* (CBC) mode, can be used to eliminate these problems.

In the CBC mode, as in the ECB mode, a message is first fragmented into block-sized pieces and then padded, if necessary. The first block is combined, via addition modulo 2, with an *initialization vector* (IV), and enciphered as in ECB mode. This ciphertext block is then combined, via addition modulo 2, with the second message block, and the result is enciphered as it would be in the ECB mode. This process is repeated until the complete message is enciphered. An error occurring in a ciphertext block will propagate throughout that block and a portion of the following block, but will not affect subsequent blocks. The use of the CBC mode is illustrated in Figure 9.

Because the ciphertext form of each block is a function of the IV and all preceding blocks in the message, the problem of block-aligned data pattern exposure can be eliminated. This is further discussed in Section 4.

3 4.3 CFB Mode

Another mode of operation, known as the *cipher feedback* (CFB) mode, allows the DES to be used as part of a key-stream generator for a CTAK cipher.[10] The CFB mode transforms the DES into a self-synchronizing stream cipher that operates on cleartext quanta of 1–64 bits in length. The key stream is generated at both the sending and receiving ends, using the basic ECB mode of the DES.

The input to the basic DES is a 64-bit shift register. The initial contents of this shift register, known as the IV, must be the same at each end. The first quantum of key stream is generated by encrypting the IV using the basic DES; if the quantum size is less than 64 bits, the extra DES output bits are discarded. The first cleartext quantum is then combined, by addition modulo 2, with this key-stream quantum. The resulting ciphertext is fed back into the input shift register at each end, and the process repeats itself for each quantum of the message. Thus, the key stream is a function of the IV and of all transmitted ciphertext.

[10] All of the public-key encryption algorithms [Merkle and Hellman 1978, Rivest et al. 1978] that have appeared in the open literature are block ciphers There appears to be no way to transform them into stream ciphers and still preserve their public-key characteristics Like the DES, though, these ciphers can be applied in either an ECB or CBC mode

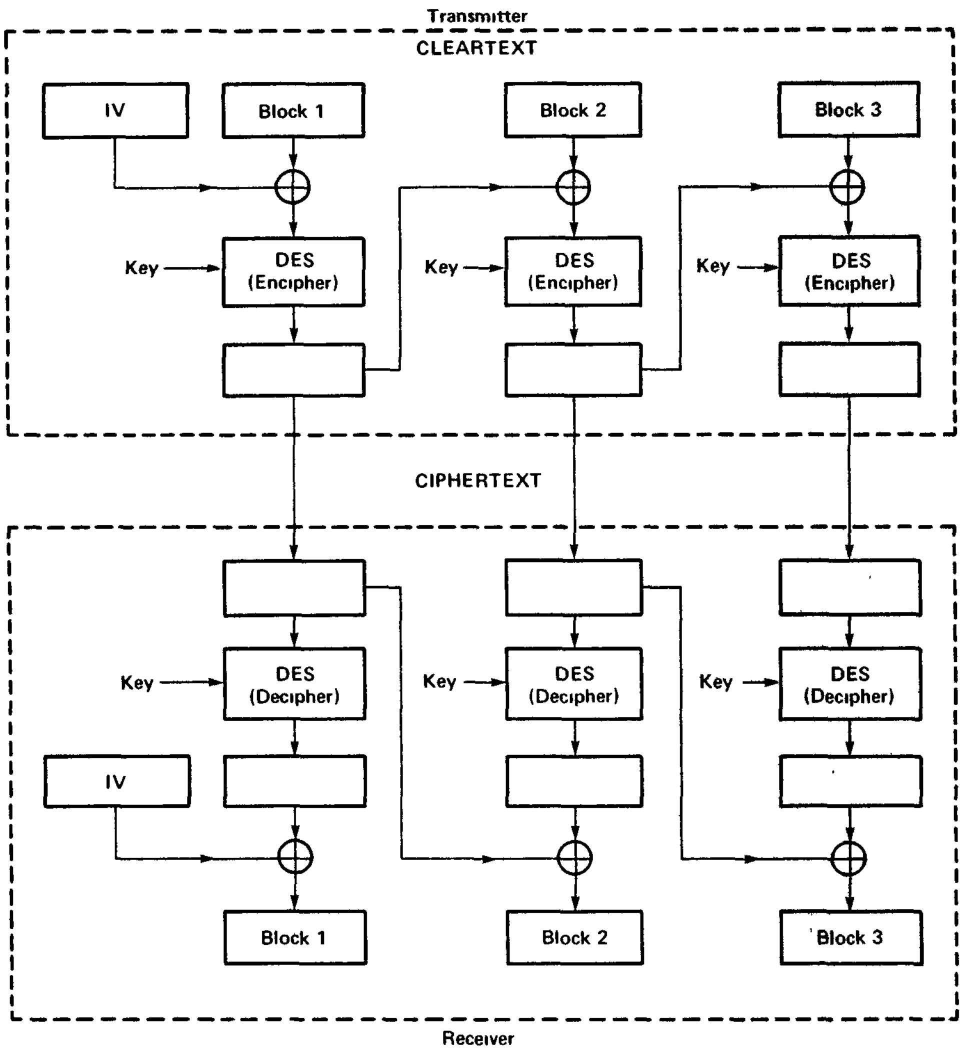

Figure 9. The CBC mode of the DES

The use of the CFB mode with strings 8 bits in length is illustrated in Figure 10.

In the CFB mode, cryptographic synchrony is achieved only if the sender and receiver use the same key and both shift registers contain the same bit pattern. If an error occurs in the ciphertext stream, a portion of the received cleartext will be garbled, but after 64 bits of error-free ciphertext have been received, the cipher automatically resynchronizes itself (since the input shift registers will then again have identical contents).

As mentioned in Section 3.3.2, stream ciphers provide a key stream whose length can be matched exactly to the length of the message to be enciphered, thus avoiding the problems associated with padding cleartext to match block sizes. Unfortunately, the throughput of stream ciphers derived from block ciphers can be substantially less than that of the underlying block cipher. For example, the throughput of the DES operating on 8-bit quanta in the CFB mode is reduced by a factor of 8 or more. This problem can be overcome by employing two quanta to encipher each message—the block size of the underlying cipher and the length of the message modulo that block size. Using two quanta will produce

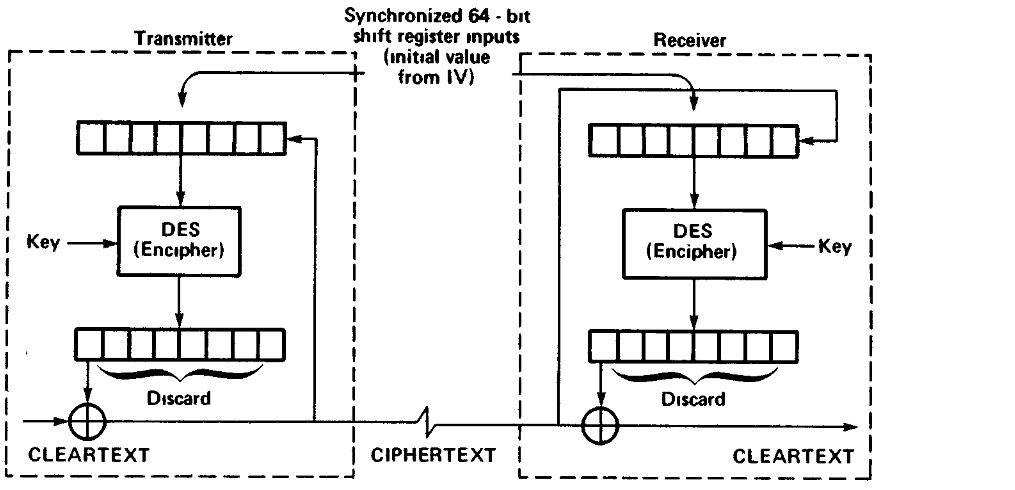

Figure 10. The CFB mode of the DES.

throughput comparable to that of the CBC mode without the need for padding.[11]

3.4.4 OFB Mode

The DES can also be turned into a KAK cipher; this mode of operation is known as the *output feedback* (OFB) mode. As mentioned in Section 3.3.2, the use of KAK ciphers hampers the development of countermeasures to message-stream modification attacks. For this reason, the OFB mode is not suitable as a basis for communication security for most applications in an open-system environment. This is further discussed in Section 6.2.

4. RELEASE OF MESSAGE CONTENTS

This section discusses how end-to-end encryption can be used to protect the contents of messages being exchanged by protocol entities. As described in Section 1.1, N-layer entities communicate by sending N-PDUs as data over an $(N - 1)$-association. An N-PDU consists of $(N + 1)$-layer data, together with N-layer protocol control information. The latter includes an association specifier indicating to which N-association the N-PDU belongs. No two existing associations have the same specifier, but once an association has been terminated, its specifier becomes available for reuse. An N-layer entity (for $N > 3$) can encrypt the entire N-PDU before passing it to the $(N - 1)$-layer for delivery. The N-PDU (which becomes the data portion of one or more $(N - 1)$-PDUs) remains encrypted from end to end on the $(N - 1)$-association.

The amount of information that can be hidden from an intruder depends on the layer in which encryption is done. For example, if encryption is done by the session layer, all transport-layer protocol control information is visible to the intruder, whereas if encryption is done by the transport layer, only network-layer (and below) information can be seen. Note, however, that having encryption done by the network layer, rather than by the transport layer, will not provide any additional end-to-end protection. This is because network-layer protocol control information must, by its very nature, be visible at each node (packet switch or gateway) that a network-PDU traverses.[12] Thus the transport layer is the lowest layer in which the adoption of

[11] This technique also can be applied to the CBC mode of operation to eliminate padding by switching to CFB mode to encipher the end of a message if the message is not an integral number of blocks long.

[12] This visibility requirement, of course, is not true in a broadcast network where there are no discernible intermediate nodes. However, this paper is concerned with security in a general open-system environment, with no assumptions made about the nature of the communications subnet or subnets traversed by a PDU

encryption would provide additional end-to-end protection.

4.1 Key Granularity

A unique key can be used for each association, or a coarser granularity of key distribution can be employed. For example, a different key can be used between each pair of communicating protocol entities, or a single key can be used among an entire group of protocol entities. As the range of use of a single key increases, the amount of information exposed in the case of disclosure of that key also increases, but the task of distributing keys becomes easier. The granularity of key distribution also affects the design of countermeasures against active attacks. As is seen in later sections, using a unique key for each association is a powerful tool for constructing such countermeasures.

When an N-entity receives an encrypted N-PDU, it must be able to determine the key under which the PDU was encrypted. How this is done depends on the granularity of key distribution. For example, suppose a different key were used for each pair of communicating N-entities.[13] Because no two pairs communicate on the same $(N - 1)$-association, the $(N - 1)$-association on which the N-PDU arrives implicitly identifies the sending N-entity, and therefore the key to use for decryption. In contrast, suppose a unique key were used for each N-association. To decrypt a received N-PDU in this case, one must determine the N-association to which the PDU belongs. Thus, the receiving N-entity must be able to read the association specifier in the NPDU before it can decrypt the rest of that PDU.

One way this can be done is to encrypt all of the PDU, except the association specifier, under the association unique key, and to encrypt the specifier itself under a key of coarser granularity. Another approach is to simply send the specifier in the clear. The implications of these two approaches are discussed in Section 5.

[13] That is, each pair of communicating N-entities has a unique key assigned to it, and this key is used for all N-associations provided by that pair to its users in the $N + 1$ layer

4.2 Masking Data Patterns

The various encryption modes must be used with care to ensure that information that will aid a cryptanalytic attack is not revealed. For instance, as mentioned in Section 3.4.1, in the ECB mode each block of cleartext in a message is independently enciphered. Because identical cleartext blocks produce identical ciphertext blocks, block-size data patterns falling on block boundaries are exposed. For this reason, and for others discussed in Section 6, the ECB mode is not appropriate for use in most contexts. It may be usable in specialized circumstances, where the information to be protected will fit in a single block. In fact, Proposed Federal Standard 1026 [National Bureau of Standards 1980b] limits its use to the cryptographic protection of keys and IVs.

4.2.1 Pattern Masking in CBC Mode

In the CBC mode, the pattern exposure problems of the ECB mode can be overcome. In this mode, the ciphertext produced for a cleartext block is a function of that block and of all preceding blocks in the message. Because of this dependence, identical cleartext blocks in two messages result in identical ciphertext blocks only if they have identical prefixes—that is, only if all preceding cleartext blocks in those messages were also identical. Thus all data patterns can be masked in the CBC mode by ensuring that every message encrypted under the same key begins with a unique prefix.

The implications of this statement vary with the protocol involved. In some, the properties of the protocol automatically provide each message with a unique prefix; in others, it is provided by appropriate use of the CBC IV. There are three classes of protocols to be considered.

First, consider protocols in which PDUs always arrive in order, with no duplicates and no losses (e.g., most protocols above layer 4). In this class of protocols, PDUs do not need to be independently encrypted and decrypted. Instead, the entire sequence of PDUs moving in one direction on an association can be enciphered as a single cryptographic message. Since this means that each association consists of only two messages, only two distinct IVs are needed to mask data patterns. Each IV is used only once during the life of the association—when encrypting the first PDU sent in a given direction. These IVs must be different from those used by all other associations employing the same key and must be agreed upon by the communicating entities when the association is established. Although it would appear that if a unique key is used for each association, the same pair of IVs could be used for all associations, for reasons discussed in Section 4.3, this is unacceptable.

In both the second and third classes of protocols, PDUs may be deleted or duplicated, or may arrive out of order. In such protocols, PDUs must be encrypted and decrypted independently; that is, they must be distinct cryptographic messages. Thus to hide data patterns in the CBC mode, every PDU encrypted under the same key must begin with a unique prefix.

The second class consists of those protocols in which data pattern hiding will happen automatically because of the PDU format they employ. That is, the first block of each PDU contains information, such as a sequence number and sender identification, that provides an association unique prefix. In these protocols, the same CBC IV can be used to encrypt all PDUs belonging to a given association. The CBC IV must be different, though, from those used by all other associations employing the same key. It must be agreed upon by the communicating entities when the association is established. The above discussion also applies to protocols in which the prefix uniquely identifies the association, but not the direction of motion. In this case, though, a pair of IVs must be used for each association. Although it would appear that if a unique key is used for each association, the same IV could be used for all associations, for reasons discussed in Section 4.3, this is unacceptable.

The third class consists of protocols that do not automatically provide a unique prefix. In such protocols, uniqueness can be guaranteed by using a different CBC IV for each PDU.[14] As above, these IVs must be unique during the life of the encryption key. Since a different IV must be used for each PDU, that IV must be transmitted along with the PDU.

4.2.2 Pattern Masking in CFB Mode

The preceding paragraphs indicate the care that must be taken when using the CBC mode. Similar care must be taken when using the CFB mode. In the CFB mode, as in the CBC mode, if two messages have identical cleartext prefixes, they will have identical ciphertext prefixes. For the first and third classes of protocols defined in Section 4.2.1, the CFB mode imposes the same constraints on IV values as does the CBC mode. More stringent constraints are imposed on the second class of protocols.

Recall that the second class consists of protocols in which each PDU must be independently encrypted, and in which the first block of each PDU contains an association unique item. When the CBC mode is used with such protocols, all PDUs belonging to a given association may be encrypted with the same IV without revealing data patterns. This is not true when the CFB mode is used. Problems arise in the CFB mode if the same IV is used when encrypting two PDUs, even if the first quanta of those PDUs differ. In particular, the resulting ciphertext will differ only in those bit positions that differ in the cleartext. That is, identical cleartext portions of those first quanta will have identical ciphertext, producing patterns visible to an intruder—especially if the quantum size is large.

To make matters worse, using the same IV when encrypting two PDUs can reveal information about their first quanta even when those quanta are entirely different. This comes about because the key stream used to encrypt the first quantum is a function only of the IV. Let A and B be the first quanta of two PDUs encrypted using the same IV. Let X be the quantum of key

[14] Alternatively, all information in blocks preceding the first block containing an association-unique item must be nonsensitive.

stream generated by that IV, and let $\oplus$ denote modulo 2 addition. The first ciphertext quanta produced will be $A \oplus X$ and $B \oplus X$, respectively. The intruder can intercept both of these quanta and compute $(A \oplus X) + (B \oplus X)$ which is $A \oplus B$. From this, any knowledge the intruder has about the contents of A or B can be used to determine the corresponding contents of the other. And of course, if two PDUs have identical prefixes, the intruder can use this same technique to gain information about all quanta up to and including the one in which they differ.

For these reasons, when PDUs are independently encrypted in the CFB mode, a different IV must be used for each PDU.[15] As above, these IVs must be unique for the life of the encryption key. Since a different IV must be used for each PDU, that IV must be transmitted along with the PDU.

4.3 Other Limitations on the Choice of IVs

Section 4.2 outlined some IV value restrictions that are needed to mask data patterns. This section discusses considerations other than pattern masking that further restrict the IV selection process.

4.3.1 A Chosen-Plaintext Attack

This section will show that to avoid a certain type of chosen-plaintext attack, the following requirements must hold for CBC IVs:

- a different IV must be used for each association;
- IVs must be pseudorandomly chosen;
- IVs must either be protected from disclosure or be different for each PDU of each association.

These requirements hold even if a unique key is used for each association. Consider the case in which the following conditions hold:

C1: A unique key is used for each association.

C2: A different IV is used for each association (but all PDUs on a given association are encrypted with the same IV).

C3: IVs are pseudorandomly chosen.

C4: IVs are not protected from disclosure.

Let A be an association, let IV_a be the IV used on that association, let U_i be the first block of the ith PDU sent in a given direction on A, let $E(\text{X})$ denote the ciphertext of X, and let $\oplus$ denote modulo 2 addition. We assume (C4) that the intruder can determine IV_a. In addition, he can observe $E(\text{IV}_a \oplus U_i)$ for all i (this is simply the ciphertext of the first block of the ith PDU). If the intruder can also force known patterns K_i to be sent on A and to appear in the first blocks of PDUs,[16] then the intruder can also observe $E(\text{IV}_a \oplus K_i)$. Thus, if the intruder is able to find a K_m such that $E(\text{IV}_a \oplus K_m) = E(\text{IV}_a \oplus U_n)$, he can determine U_n.

Finding the right K_m may not be difficult, depending on the protocol involved and on the information transmitted on the association. For example, much of U_n might be a known constant. If so, the space of possible values would be small and values of K_m could be chosen to search this space exhaustively. Or, for another example, the intruder might have some idea of possible U_n values, and could choose values of K_m to make a series of educated guesses.

In both cases, the proper choice of K_m depends on the intruder knowing IV_a before choosing K_m. If IV_a were kept secret, such an attack could not be made. Or, if a different, pseudorandom IV were used for each PDU, the attack would be prevented. In this case, the IV would not need to be secret as long as the intruder was unable to predict its value before choosing K_m.

[15] Alternatively, all information in quanta up to and including the first quantum containing an association-unique item must be nonsensitive

[16] This is not as farfetched as it might seem at first glance For example, consider the following scenario in which the intruder I is also a legitimate network user Suppose user X has an encrypted link L between his terminal and host H, that I wants to subvert Also suppose that X has a network mailbox on H I can watch L, determine IV_a, and send mail messages to X containing the appropriate K_i. When X issues a command on H to read his mailbox, the K_i will be encrypted and sent on L. It is not unreasonable to assume each mail message could be sent as a separate PDU, and that K_i could appear in the first block of that PDU

If condition C1 is relaxed, the attack is simply made easier. If either C2 or C3 is relaxed, similar sorts of chosen-plaintext attacks can be made, even if the IV is kept secret. Thus, all of the requirements stated at the beginning of this section are necessary.

4.3.2 An Exhaustive Table Attack

As mentioned in Section 4.2, if only pattern-masking considerations are taken into account, there seem to be circumstances in which a constant CBC IV can be used for all associations. Section 4.3.1 has given one example of why this is not desirable. In this section we present another argument against the use of a constant IV. In particular, we show that using a constant CBC IV may allow an intruder to examine the first PDU of an association and determine the encryption key being used for that association.

Let IV_c be the constant CBC IV, let $Ek(X)$ be the ciphertext of X encrypted under the key k, and let $\oplus$ denote modulo 2 addition. Now suppose that the protocol employed has the property that the first block B of the first PDU of every association is constant. Then, for each association, the intruder is able to observe $E_m(B \oplus IV_c)$, where m is the key used for that association.

Note that $B \oplus IV_c$ is constant for all associations. Knowing this, the intruder can compute $Ek(B \oplus IV_c)$ for all 2^{56} possible keys k. He can store this information in a table in such a way that he can use $Ek(B \oplus IV_c)$ to look up k. This table need only be computed once. Given this table, the intruder can subvert any association simply by recording the ciphertext of the first block of the first PDU of that association and using that ciphertext to look up the key being employed. B does not even need to be constant for this attack to be successful. If B ranges over a small number of values, a separate table can be computed for each possible value of B.

To greatly reduce the power of this exhaustive table attack, it is enough to use a separate IV for each association. This eliminates the intruder's ability to compute one table that will allow him to subvert all associations. The intruder can still compute an exhaustive table for $B \oplus IV_i$, for some given i, and wait for an association that uses IV_i to come along. If and when one does, the intruder can determine the key being employed, and subvert the association. Or the intruder can observe the IV being used on a given association, record all the PDUs sent on the association, compute the exhaustive table for that particular $B \oplus IV$, and then decipher the recorded PDUs. To prevent these sorts of attacks, IVs should be protected from disclosure and pseudorandomly chosen. (The latter requirement effectively eliminates the possibility that an intruder might guess the value of the IV employed on a given association.)

Similar sorts of attacks can be performed if the CFB mode with a quantum size of 64 bits is employed, since the intruder can observe $B \oplus Ek(IV)$ and from this compute $Ek(IV)$. He can then use this value to look up k in an exhaustive table for IV.

4.3.3 Summary of IV Requirements

Even if the attacks discussed in the previous two sections seem infeasible, the countermeasures needed to prevent them are not particularly onerous. For this reason, it is strongly recommended that the following restrictions be imposed on CBC IVs:

- CBC IVs must be pseudorandomly selected.
- The pseudorandomly chosen CBC IV must be assigned during the association initiation process and then used to encrypt all PDUs on that association.
- CBC IVs must be protected from disclosure.
- CBC IVs must be protected from undetectable, predictable modification.[17]

For CFB IVs the following restrictions are strongly recommended:

[17] This restriction does not follow from the discussion above The reasons for requiring it are discussed in Section 6.2, but it is included here for completeness.

- CFB IVs must be pseudorandomly selected.[18]
- Each PDU on a given association must be encrypted using a different CFB IV.
- CFB IVs must be protected from disclosure.[18]

Note that this is a stronger set of restrictions than that required by Proposed Federal Standard 1026 [National Bureau of Standards 1980b]. In particular, the latter does not require that CFB IVs be protected from disclosure.

5. TRAFFIC ANALYSIS

Traffic analysis countermeasures are concerned with masking the frequency, length, and origin–destination patterns of the message traffic between protocol entities. The precision with which an intruder can analyze these patterns determines the amount of information that he can gain from that analysis. As discussed in Section 2.1, appropriate link encryption techniques can mask all such patterns and thus prevent all traffic analysis attacks. This is not true with end-to-end techniques. As will be seen below it is easy to achieve a certain level of protection, but beyond this the techniques become clumsy and increasingly expensive, and cannot, in any case, completely prevent all forms of traffic analysis.

In an open-system environment, end-to-end techniques can limit the precision of origin–destination analysis but cannot entirely prevent it. The precision with which such analysis can be done depends on the layer in which encryption is performed. For example, if encryption were performed in the presentation layer, an intruder could determine which presentation, session, and transport entities were involved in a given association. Performing encryption in the transport layer would limit the intruder to observing patterns at the network-address level. That is, he could tell which transport-layer entities were exchanging messages, but not which (or how many) higher level entities were doing so.

[18] This is true at least when a quantum size of 64 is used.

Masking these host-level patterns is infeasible in an open-system environment. Doing so would involve encrypting the actual destinations of network PDUs using a network-wide key and sending all PDUs to all hosts on the network.[19] This would, of course, cause an extreme reduction in the effective bandwidth of the network and waste large amounts of processing power in the hosts. Thus, the most that end-to-end measures can practically do is to limit origin–destination analysis to the host level.

The preceding discussion assumes that the entire transport PDU is encrypted, including the association specifier. In some cases this can be accomplished simply by encrypting the entire PDU with the same key. As discussed in Section 4.1, the situation becomes more complicated if a unique key is used for each association. In this case, the receiving transport entity must be able to read the specifier in a PDU before it can decrypt the rest of that PDU. One solution is to encrypt all of the PDU, except that specifier, under the association unique key, and to encrypt the specifier itself under a key of coarser granularity.

This is a slight oversimplification. If the specifier contains information from which the identities of the entities at each end of the association can be determined, encrypting it will hide these identities. However, if the specifier is encrypted by itself, it will always produce the same ciphertext, allowing an intruder to determine which PDUs have the same specifier. Since specifiers are reused, this does not precisely identify the association to which a PDU belongs, but it does reveal useful patterns. To mask such patterns completely, the specifier must be encrypted together with some other quantity that changes with each PDU.

Separately encrypting the association specifier complicates the protection mechanisms. Another approach is simply to send the specifier in the clear and accept the fact

[19] This is perfectly feasible in a broadcast network where all network PDUs can, in fact, be seen by all hosts However, as mentioned earlier, we are concerned with security in an open-system environment and make no assumptions about the nature of the communications subnet or subnets traversed by a PDU.

that a somewhat more precise level of traffic analysis is possible. The increase in precision can be reduced by ensuring that the specifier does not identify the entities at each end of the association. Intruder changes to a specifier sent in the clear do not cause security violations. The specifier merely indicates which key should be used to decrypt a PDU; if it is changed, the PDU will be decrypted with the wrong key. This will be detected by the mechanisms described in Section 6.

End-to-end measures can also limit the precision of message frequency and length pattern analysis. If encryption were performed in the transport layer (to limit origin–destination analysis, for instance), the intruder would already be limited to examining such patterns at the host level. That is, he could determine how many messages were sent from one host to another, at what rate they were sent, and what their lengths were, but he could not relate this to higher level entities residing on those hosts. As noted, if association specifiers are sent in the clear, a somewhat more precise level of traffic analysis is possible.

If further protection were desired, the transport layer could maintain a predefined pattern of message frequencies and lengths between each pair of hosts in the network (or perhaps only between those hosts for which such protection is desired). This would be done by generating dummy messages of appropriate lengths and padding actual messages to meet the desired pattern. A sublayer could be defined to generate dummy messages, to pad actual messages on the sending end, and to delete dummy messages and strip off padding on the receiving end. This technique is workable if only a small percentage of hosts employ it, particularly if the level of message traffic to be maintained is low. Widespread use is probably infeasible since it would reduce effective network bandwidth significantly and waste considerable amounts of host processing power.

In summary, end-to-end techniques can effectively and efficiently limit all forms of traffic analysis to the host level. Beyond the host level, further limitations on information release become increasingly expensive and are probably not necessary in nonmilitary environments.

6. MESSAGE STREAM MODIFICATION

In this section we discuss methods that can be used to detect message-stream-modification (MSM) attacks. As discussed in Section 1.3.3, message-stream modification refers to attack on the integrity, authenticity, and ordering of PDUs passing on an association. Integrity means that a PDU has not been modified while passing on the association. Authenticity means that the source of the PDU is the entity at the other end of the association. And ordering means that the PDU can be properly located in the stream of PDUs moving from the source to the destination entity.

The need to provide protection from passive attacks is apparent. It could be argued that most users and many application programs can easily detect MSM attacks without the inclusion of explicit countermeasures in the communications protocols involved, especially when an encryption scheme with suitable error propagation characteristics is used. But, many application programs are not prepared to detect attacks of this sort, and many message streams do not contain the proper kind of redundant information to allow such detection. Even messages directed to a user may admit a wide range of "meaningful" contents. In addition, providing detection measures in a communications protocol obviates the need for each application programmer to devise an application-specific means of detection. It also makes it unnecessary to check many sets of security measures for correctness. Finally, it ensures that the correctness of security measures need not be rechecked each time an application program is changed.

MSM attacks can cause PDUs to arrive out of order or to be modified, lost, or duplicated. Restoring cryptographic synchrony after such events can be difficult and expensive if the ability to decrypt a PDU depends on having successfully decrypted all previously sent PDUs. For this reason, protocols that implement countermeasures to MSM attacks will generally encrypt each PDU independently. Independent PDU encryption is assumed in the discussion that follows.

6.1 Overview

Measures to detect the three forms of MSM attacks are hierarchically organized. The most fundamental measures are those that ensure message integrity. These measures involve cryptographically binding an error detection code to a PDU so that changes to that PDU are detected with high probability, as are PDUs decrypted with the wrong key. Such measures are the foundation on which authentication measures are built. Ordering measures rely on both integrity and authentication measures.

Measures to ensure message authenticity are based on the ability of the receiver to determine, with high probability, the association to which a PDU belongs, regardless of what an intruder does. They involve uniquely identifying an association for all time and binding this unique ID (either implicitly or explicitly) to every PDU passing on that association. The integrity measures are relied on to ensure that the binding is inalterable by an intruder.

Measures to ensure message ordering are based on the ability of the receiver to determine, with high probability, the position of a PDU in the stream of PDUs moving in one direction on an association. They involve binding to each PDU a sequence number that identifies that PDU's position in the stream. The integrity measures are relied on to ensure that the binding is inalterable by an intruder. The authentication measures are relied on to filter out PDUs that were not sent by the other end of the association at all.

6.2 Message Integrity

As mentioned above, message integrity measures involve cryptographically binding an error detection code to each PDU in such a way that

Property I. Changes made to an encrypted PDU as it traverses the network will be detected with high probability when the PDU is decrypted.

Property II. A PDU that has been decrypted using the wrong key will be detected with high probability.

Measures that ensure Property I generally ensure Property II also. This is because decrypting a PDU with the wrong key produces a result at least as garbled as that produced by decrypting a PDU that has been modified en route. However, Property II is explicitly stated because it is one of the foundations on which authentication measures are based.

6.2.1 Interactions between Error Detection and Encryption

Communication protocols have long used error detection codes to detect transmission errors. A code is generally attached to each PDU. It allows the receiving protocol entity to detect which PDUs were modified en route by a transmission error. This, of course, does not protect the integrity of the PDU against a deliberate attack. An intruder can make an arbitrary change to a PDU, compute what the new value of the error code should be, and replace the old error code with the new. Message integrity measures involve binding the error code to the PDU in a way that prevents an intruder from doing this.

Encrypting the error code together with the rest of the PDU does not necessarily guarantee integrity. The combined properties of the error detection algorithm and the encryption algorithm must be taken into account. For example, if a KAK cipher is used, changing a bit in the ciphertext merely complements the corresponding bit in the decrypted cleartext. The intruder might be able to take advantage of this to make changes that are invariant under the error detection algorithm employed, that is, changes that generate the same error code value.

This is certainly the case for longitudinal parity checks, such as the one presented in Proposed Federal Standard 1026. It may also be true for more sophisticated codes, such as cyclic codes. In fact, for longitudinal parity checks, the intruder can complement any and all bits of the PDU, complementing appropriate bits in the error code, if necessary, to make the changes undetectable. This is why KAK ciphers, such as the

OFB mode of the DES, are not especially good bases for countermeasures to MSM attacks.

As another example, suppose the ECB mode of the DES is used. Then, since each block of a PDU is independently encrypted, switching blocks in an encrypted PDU merely switches the corresponding blocks in the decrypted PDU [Ehrsam et al. 1978]. Such a change is undetectable by a longitudinal parity check. This, together with the pattern-revealing problem discussed in Section 4.2, is why the use of the ECB mode must be severely restricted.

Finally, suppose that the CBC mode of the DES is used to encrypt PDUs and that each PDU contains a longitudinal parity check whose size is an integral multiple or fraction of the CBC block size. Until recently, this combination was thought to be sufficient to ensure message integrity. Jueneman [1983] has shown that this is not the case; in particular, he has shown that an intruder can undetectably exchange two ciphertext blocks of an encrypted PDU. (Analogous attacks can be effected under the CFB mode.)

To envision this, consider Figure 11, which shows the CBC algorithm being applied to several blocks of a PDU (for simplicity, it is assumed that the size of the checksum is the same as the block size). Let $\oplus$ denote modulo 2 addition. Let $X = B_2 \oplus C_2 \oplus D_2$. The following paragraph will show that X, and therefore the entire checksum, is unchanged if ciphertext blocks B and C are switched.

Before switching, $B_2 = A \oplus B_1$, $C_2 = B \oplus C_1$, and $D_2 = C \oplus D_1$. Thus $X = A \oplus B_1 \oplus B \oplus C_1 \oplus C \oplus D_1$. If blocks B and C are switched, then $B_2 = A \oplus C_1$, $C_2 = C \oplus B_1$, and $D_2 = B \oplus D_1$. So $X = A + C_1 \oplus C \oplus B_1 \oplus B \oplus D_1$. Since $\oplus$ is commutative, X is unchanged.

6.2.2 *An Approach to Message Integrity*

These examples reveal a fundamental requirement of message integrity mechanisms, namely, that an intruder change to an encrypted PDU must generate a change in the decrypted PDU that is not invariant under the error detection algorithm being employed. The CBC and CFB modes of the DES have this property when used with an appropriate error detection algorithm. In particular, in either mode (except for a CFB boundary condition discussed below), inverting a ciphertext bit affects 64 bits of decrypted cleartext unpredictably. That is, each of those 64 bits has a probability of approximately one half of being changed. Thus, a bit inversion to an encrypted PDU introduces, in effect, a random burst error into the decrypted PDU. If each PDU includes an error code that has good burst error detection properties, such attacks on message integrity have a high probability of being detected. If, in addition, the error detection code is sensitive to bit ordering, then an attack of the sort detailed in Section 6.2.1 will also be detected with high probability.

For example, if an N-bit cyclic code [Peterson and Weldon 1972] is used, the probability of such an attack being undetected is on the order of $1/(2^N)$, since that is the probability that such a code will not detect an arbitrary burst error or reordering.[20] This shows that the probability of such an attack being undetected can be made arbitrarily small. Of course, increasing the size of the error code decreases the effective network bandwidth, since the code must be included in each PDU.

Note that if a PDU encrypted using the CBC or CFB mode is decrypted using the wrong key, every bit in that PDU is changed with a probability of approximately one half. This, again, is equivalent to introducing a random burst error, and thus its probability of detection is high. If an N-bit cyclic code is used, the probability that this type of event is unrecognized is on the order of $1/(2^N)$.

6 2.3 *Boundary Conditions*

As mentioned above, there is a CFB boundary condition that must be kept in mind. In general, in the CFB mode, changes to a ciphertext quantum of a PDU complement the corresponding bits of the cleartext quantum and then cause unpredictable

[20] Further security can be had by employing a secret value to initialize the shift register of the cyclic code.

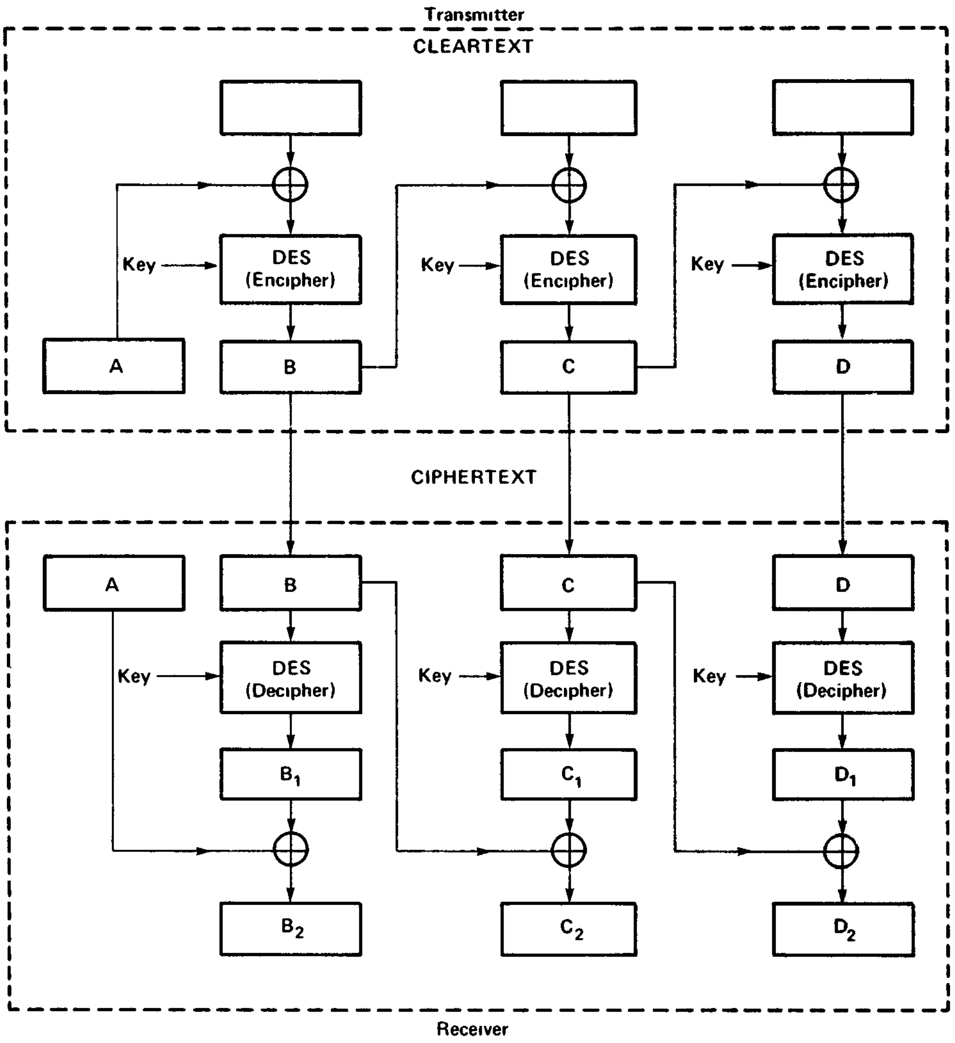

Figure 11. Block mode applied to several blocks of a PDU.

changes in the next 64 bits of decrypted cleartext. If changes are made to the last quantum, there are no bits left in which to propagate unpredictable changes. Thus, an intruder can make predictable changes to the last quantum of a PDU without causing unpredictable changes to other parts of that PDU.

Because of this, the last quantum of a PDU encrypted using the CFB mode should contain only padding bits and/or error code bits. The padding bits are ignored. Changes to the error code bits are not a problem; being able to make a predictable change to an error code, without being able to make such a change to the information whose validity it checks, does not help an intruder. Protocols that put the error code at the end of the PDU are thus unaffected as long as the quantum size is no greater than the length of the error code. If a larger quantum size is used, these protocols must pad the PDUs appropriately.

The CBC mode has a different sort of boundary condition, one that requires that the CBC IV be protected from changes that are both predictable and undetectable. Failure to do so allows an intruder to make predictable changes to the first block of each PDU encrypted with that IV.

To see this, let K be the IV and C the cleartext of the first block. Let $\oplus$ denote modulo 2 addition, and let $S = C \oplus K$. By examining the CBC algorithm, one can see that the sending protocol entity computes S, encrypts it using the DES, and transmits the resulting ciphertext. The receiving entity decrypts the ciphertext to produce S and then computes $S \oplus K = C$. Equivalently, if $S(i)$, $K(i)$, and $C(i)$ denote the ith bit of S, K, and C, respectively, then $C(i) = S(i) \oplus K(i)$. From this, the properties of modulo 2 addition reveal that $\overline{C(i)} = S(i) \oplus \overline{K(i)}$ where the overbar denotes bit complementation. This means that if an intruder can predictably change bits in K, he can predictably change the corresponding bits in C. Thus K must be protected when transmitted.

Note that to ensure security, it is not necessary that intruder changes to CBC IV values be detectable, as long as they are unpredictable. An unpredictable change to a CBC IV will generate an unpredictable change in all bits of the first block of the PDU, and will thus be detected by the normal PDU integrity mechanisms. Therefore it is not necessary to attach an error detection code to a transmitted CBC IV. It is enough to encrypt it so that an intruder cannot predictably change it.

The CFB mode does not have this boundary condition. Any change, predictable or not, to a CFB IV will generate an unpredictable change in the first 64 bits of the PDU. Thus CFB IV values do not need to be protected from modification.

6.3 Message Authenticity

Attacks on message authenticity involve the insertion of spurious PDUs into the stream of PDUs moving in one direction on an association. Such attacks include

- inserting a PDU synthesized by the intruder,
- playing back a valid PDU from another association,
- playing back a valid PDU that had previously been sent in the opposite direction on the same association.[21]

The goal of message authentication measures is to detect such attacks, by making it possible for the receiving entity to reliably determine the association to which a PDU belongs and the direction in which it is moving. These measures are built on top of the integrity measures discussed in Section 6.2.

The approach to detecting the first two types of attack is to identify uniquely an association for all time, and to attach inalterably (either implicitly or explicitly) this unique ID to every PDU passing on that association. For reasons discussed in Section 6.4, an association may be sequentially assigned a number of unique IDs; that is, it may have a different unique ID at different times in its life. However, no unique ID will ever be used for more than one association. Thus, from the unique ID, one can reliably determine to which association a PDU belongs.

If a separate key is used for each association, the key provides an implicit unique ID. That is, the key used to encrypt a PDU uniquely identifies the association to which it belongs. Thus Property II of the message integrity measures ensures that PDUs from other associations will be detected. In this context, synthesized PDUs are equivalent to PDUs from other associations and will also be detected.[22]

If more than one association uses the same key, each of these associations must be assigned a local ID that distinguishes it from all the others. Each PDU belonging to one of these associations must contain that association's local ID. In this case it is the local ID, together with the key, that forms the unique association ID. Property I of the message integrity measures ensures that an intruder cannot undetectably alter

[21] Playing back a valid PDU that was previously sent in the same direction on the same association is an attack on message ordering This is discussed in Section 6 4

[22] The discussion in the above paragraph assumes that keys are never reused. This is a reasonable assumption since the key space of the DES is very large and, for proper security, keys should be random or pseudorandomly chosen. Thus, the probability of a key being reused is very small. As long as this is the case, the occasional reuse of a key does not effectively reduce security, since the intruder has no way of detecting key reuse and thus cannot take advantage of it. If the probability of key reuse were high, the intruder could simply guess which key was being used.

the unique ID by modifying the local ID in a PDU. Note that the association specifier described in Section 4 can be used as the local ID if that specifier is not sent in the clear and if specifiers are unique for the life of the key. Otherwise, a separate local ID field is needed in each PDU.

The unique ID allows the receiver to determine which association a PDU belongs to, but it does not indicate the direction in which the PDU is moving. Thus an additional mechanism is needed to detect the third type of attack on PDU authenticity. There are several approaches to this problem.

One approach relies on the way in which IVs are chosen. Recall from Section 4.2 that, for many protocols, distinct IVs or sets of IVs, must be used in each direction on an association. In such protocols, if an entity receives a PDU that is a playback of one it previously sent, it will decrypt it with the wrong IV. Property I of the integrity measures ensures that this will be detected, since changing the IV is equivalent, in this sense, to changing the encrypted PDU itself. Thus, such protocols automatically detect the third type of attack on PDU authenticity.

A different approach must be found for protocols that do not use distinct IVs for each direction on an association. The simplest approach is to add a field to each PDU indicating the direction in which it is moving. Property I of the message integrity measures ensures that an intruder cannot undetectably alter this field. In many protocols, the association specifier indicates in which direction a PDU is moving. This is sufficient if that specifier is not sent in the clear and if specifiers are unique for the life of the key. Otherwise, a separate direction indicating field is needed in each PDU.

6.4 Message Ordering

Attacks on message ordering involve disrupting the stream of PDUs moving in one direction on an association. Such attacks include

- deleting PDUs from the stream;
- altering the order of PDUs in the stream;
- duplicating PDUs in the stream, that is, recording a legitimate PDU from the stream and later playing it back into that stream.

The goal of message-ordering measures is to detect such attacks by making it possible for the receiving entity to reliably determine the position of each PDU in the stream of PDUs being sent by its correspondent peer. (The message authenticity measures are relied on to filter out PDUs that do not belong to this stream at all.)

Message ordering is achieved by including a sequence number in each PDU that indicates the position of the PDU in the stream of PDUs moving in one direction on an association. Property I of the integrity mechanisms ensures that any attempt to change a sequence number will be detected. In this way, the receiving entity can detect missing, duplicated, or out-of-order PDUs.

To ensure that the third type of ordering attack is detected, sequence numbers may not be reused during the life of an association unique ID. This ban on reuse prevents an intruder from undetectably playing back an old PDU after the sequence numbers have cycled.[23]

One approach to the problem of sequence number reuse is to allow sequence numbers to become arbitrarily large, so that they never cycle during the life of the association. This can be done in two ways. The first way is to use a fixed-size sequence number field large enough that sequence numbers will never cycle in practice (64 bits should more than suffice). The second way is to use an extensible sequence number field, that is, a field whose size grows as needed. The use of a large fixed field reduces network bandwidth in the early life of the association more than would the use of an extensible field. On the other hand,

[23] This must also be done to detect an obscure version of the second type of ordering attack. In such an attack, the intruder holds back all PDUs sent until the sequence numbers cycle. He then switches one or more pairs of PDUs that have the same sequence number Such an attack would not be detected. If the sequence numbers are used for recovery from transmission errors, as well as for detection of MSM attacks, such an attack cannot happen. This is because reliable recovery from transmission errors requires that a sequence number not be reused until the receipt of the previous PDU that used it is acknowledged.

it is harder to process an extensible field than one of fixed size.

Another approach is to allow sequence numbers to cycle, and to change the association unique ID (either the local ID or the key) when this occurs. As mentioned in Section 6.3, this does not adversely affect the message authentication mechanisms as long as the new ID is also unique.

The first approach avoids the need to change unique IDs in the middle of an association (which may involve considerable overhead). On the other hand, when large numbers of PDUs must be transmitted, the first approach consumes more network bandwidth than the second approach. In fact, if a large fixed field is used instead of an extensible one, the first approach consumes more bandwidth, regardless of the amount of traffic on the association. This problem could be reduced by allowing the size of the fixed field to be negotiated at association initiation time. In this way the size could be matched to the amount of traffic to be sent. Finally, it may be desirable to limit the number of PDUs sent under one key to make cryptanalytic attacks more difficult. To accomplish this, the second approach is preferable to the first.

6.5 Concluding Remarks on Message-Stream Modification

This section has shown that to detect MSM attacks, a protocol must be able to detect deleted, duplicated, and out-of-order PDUs, as well as spurious PDUs from other associations and PDUs whose contents have been modified. To recover from transient MSM attacks,[24] the protocol must further be capable of retransmitting missing or modified PDUs, throwing away spurious and duplicate PDUs, and resequencing out-of-order PDUs. This capability requires fairly elaborate protocol mechanisms.

Most protocols above the transport layer rely on that layer to ensure that PDUs arrive in order, with no duplicates or losses. Thus, they are not able to cope with MSM attacks. Changing one of these protocols to cope with such attacks would complicate it considerably, and would duplicate mechanisms already present in the transport layer to deal with transmission errors. This argues that the proper way to handle MSM attacks is to augment the existing transmission error recovery mechanisms of the transport layer. Since countermeasures to MSM attacks can also protect against release of message contents, it would seem that the proper locus for all end-to-end security measures discussed so far is the transport layer.

7. DENIAL OF MESSAGE SERVICE

As mentioned above, denial-of-message-service attacks can be viewed as persistent MSM attacks. Such attacks include

- discarding all PDUs passing on an association in either or both directions,
- delaying all PDUs passing on an association in either or both directions.

The countermeasures discussed in Section 6 can detect some, but not all, forms of denial-of-message-service attacks. In particular, they cannot reliably detect such attacks if they begin while the association is quiescent, that is, when no PDUs are outstanding in either direction. In general, in such a situation, a protocol entity at one end of an association has no way of determining when the next PDU should arrive from its correspondent peer entity. The entity is thus unable to detect a denial-of-service attack that completely cuts off the flow of PDUs from its peer entity.

In many cases, the entity attempting to send PDUs will detect the attack but it has no way of notifying the other entity. The receiving entity will remain unaware of the attack until it attempts to send PDUs itself. In some situations, such an event may never occur. This could happen, for example, if one entity were an editor program waiting for a user request and the correspondent peer were a user at a terminal. The user would be able to detect such attacks, since the editor would not respond.

[24] Persistent MSM attacks become denial of service attacks that can be detected but not recovered from. Such attacks and measures to detect them are discussed in Section 7.

The editor, on the other hand, would wait forever for the next user request.

An additional countermeasure is needed to detect such attacks. It entails adding a request-response mechanism [Kent 1976] to the MSM countermeasures. This mechanism can be incorporated either into the same layer as the MSM countermeasures or into a higher layer.

The request-response mechanism involves the periodic exchange of a pair of PDUs between peer entities to verify that an open path exists between them. To do this, a timer is added to each end of the association; each timer periodically triggers the transmission of a request PDU that forces a response from the other end. Lack of a response indicates that a denial-of-service attack is taking place. As a refinement, each of these PDUs can contain a description of the sender's understanding of the state of the association. Including this description may allow more rapid recovery from transient denial-of-service attacks.

Increasing the frequency with which request-response PDUs are exchanged will reduce the time interval during which a denial-of-service attack will remain undetected. Unfortunately, this increased message traffic also reduces the effective bandwidth of the network.

8. SPURIOUS ASSOCIATION INITIATION

The two preceding sections have dealt with active attacks that can occur after an association has been established. In this section we discuss active attacks that can occur during the association initiation process itself. These attacks are referred to as spurious association initiation attacks and fall into two categories:

- attempting to establish an association under a false identity,
- playing back a recording of a previous legitimate association initiation sequence.

To counteract the first type of attack, an association must be initiated in a way that supports secure identification of the principals[25] at each end. Verification of identity is a complex issue that interacts with issues of user authentication and access control that are beyond the scope of this paper. As is seen below, though, a portion of the identification problem must be dealt with during association initiation itself.

To counteract the second type of attack, association initiation must include a mechanism to verify the time integrity of the association, that is, to verify that the association initiation attempt is being made in real time. Both types of attacks are similar in nature to MSM attacks; however, the context of association initiation requires the use of additional countermeasures.

8.1 Authentication of Principals

To detect the first kind of attack, it must be possible to authenticate the identities of the principals involved in an association initiation attempt. The ability to encipher messages under a specific key carries with it an implicit form of authentication. In particular, only a possessor of a given key is able to encipher messages under it. Thus, authentication of principals can be based on the appropriate distribution and protection of secret encryption keys.[26]

The precision with which a principal can be identified depends on the granularity of key distribution. For example, if a single key is employed by a subset of the principals in a network, then these principals form a secure virtual subnet. Knowledge of the key identifies the members of the subnet without distinguishing among them. To

[25] The term *principal* refers to the entity held accountable by the security policy in force [Saltzer and Schroeder 1975]. Depending on the granularity of the protection mechanisms employed, a principal can be a process, a user, a terminal, a host, or a network. For example, if principals are hosts, security policies are enforced at the host level. That is, the protection mechanisms do not distinguish between users (processes, terminals) on the same host.

[26] In a public-key cipher, encrypting a message under the sender's secret key authenticates the sender, while encrypting it under the receiver's public key preserves its secrecy. Thus in such ciphers, authentication and secrecy are separable, independent functions, whereas in conventional ciphers they are closely integrated.

be able to authenticate the identity of each individual principal, a finer level of granularity is needed. This can be achieved by using a different key for communication between each pair of principals. Such a scheme is known as *pairwise disjoint key distribution.*

8.1.1 Hierarchic Key Distribution

If pairwise disjoint key distribution is used, each principal involved in an association can reliably verify the identity of the principal at the other end. But this approach, by itself, has the problem that two principals always use the same key (call it the long-term key) when communicating with each other. To extend this approach to allow a per-association key,[27] the key must be securely distributed to each end of the association. One method of distribution is to transmit the per-association key at association initiation time, encrypted under the long-term key. The mechanisms discussed in Section 6 then protect the per-association key from undetected modification.

Keys held for long periods of time and used exclusively for the transmission of per-association keys are referred to as *master, primary,* or *key-encrypting* keys. One or more keys used during the course of a single association are referred to as *working, secondary,* or *data-encrypting* keys. Thus, master keys are used to authenticate principals and to protect transmitted working keys, while working keys are used exclusively to encrypt PDUs on a single association. Using a master key as a bootstrapping mechanism in the distribution of working keys is referred to as *hierarchic key distribution.*

As mentioned in Section 6, to ensure security, working keys must be pseudorandomly chosen from the key space. A number of approaches may be used to generate keys [Matyas and Meyer 1978]. For example, one can encrypt a nonrepeating value under the master key and use the resulting ciphertext as the working key. This paper does not consider key generation issues in detail.

[27] Earlier parts of this paper have shown that the use of a distinct key for each association minimizes the amount of information exposed in case of key disclosure and simplifies the design of security measures.

8.1.2 Working Key Lifetime

Recall (from Section 1.2.2) that an association refers either to a connection or to the exchange of messages via the connectionless data transmission service [Burruss et al. 1981]. The unit of data transferred in a single invocation of the connectionless data service is referred to here as a *transaction.* Thus, an association is either a connection or a stream of transactions.

If the association is a connection, a working key is generated when the association is initiated. It is used until the association is terminated, or until the PDU sequence numbers cycle. If the sequence numbers cycle, a new working key is generated and the life of the association continues.

If the association is a stream of transactions, there is no explicit association initiation or termination; each transaction is completely independent of all other transactions. For this reason, it is not clear how to bound the life of a working key. In many ways each transaction behaves like a distinct association; for example, ordering between separate transactions is not guaranteed, and each transaction's PDUs may be sequence numbered independently from the PDUs of other transactions. But using a separate working key for each transaction is clearly unacceptable, since the very purpose of the connectionless service is to transmit data with minimum overhead. Consequently, some other solution must be found. Alternatives include using a working key for a certain period of time or for sending a certain number of transactions. Depending on the design details of the particular connectionless data service involved, both alternatives may be effectively identical to using the same working key for more than one association.

8.1 3 Proliferation of Master Keys

A problem with pairwise disjoint distribution of master keys is that the number of master keys needed tends to increase rapidly as the size of the network increases and the granularity of the keys decreases. For example, if pairwise disjoint distribution is carried out at the level of individual hosts, complete communication among N hosts requires on the order of N^2 keys. If this form of key distribution is carried out at the level of terminals or of individual users, the number of master keys involved can become staggering.

Managing such large numbers of keys can be both cumbersome and expensive. For example, if the master key list on a host is subverted, all users on that list must be notified. If a user loses his list of master keys, all hosts on that list must be notified. In addition, a convenient way for a user to carry his personal master keys might be on a magnetically stripped card, similar to many credit cards in common use today. The number of distinct keys that can be recorded on such a card limits the number of hosts the user can access. This problem can be reduced by using higher recording densities, more magnetic material per card, or multiple cards, but it cannot be eliminated. Thus, to reduce the proliferation of master keys significantly, the concept of trusted intermediaries known as *key distribution centers* has been developed [Branstad 1973, 1975].

8.1.4 Key Distribution Centers

A key distribution center (KDC) is a secure host dedicated to acting as a trusted intermediary in the establishment of secure associations. A conventional KDC (i.e., a KDC that handles conventional ciphers) holds one master key for each principal that uses its services. A principal that wants to initiate an association first contacts the KDC and indicates the target of the association. The KDC generates a working key and sends it to both the initiator and the target.[28] The copy of the working key sent to the initiator is enciphered under the master key of the initiator, while the copy sent to the target is enciphered under that of the target.[29] In this way, each association participant need hold only one master key, and yet mutually suspicious participants are protected, just as if pairwise disjoint key distribution were employed.

Key distribution centers are needed for public-key ciphers as well. It has been suggested that telephone book listings of public keys would suffice [Rivest et al. 1978]. Unfortunately, such a static approach is not appropriate in an open-system environment. First and most important, the frequency of key changes, additions, and deletions will be high owing to changing user populations, the need to replace lost keys, and the need to change keys periodically to reduce the impact of key exposure. Second, as the use of open systems becomes widespread, the size of such key phone books would become unmanageable. Finally, public keys tend to be very long, so manual key entry would be cumbersome and error prone. For these reasons, the automated key distribution and maintenance services of a KDC are necessary.

A public-key KDC provides a tamper-free channel for the distribution of public keys. The initiator of an association contacts the KDC to retrieve the public key of the target. The KDC then transmits this key to the requester, enciphered under the KDC's secret key. The requester uses the public key of the KDC to decipher this response. In this way an intruder is unable to undetectably forge or modify a response sent from a public-key KDC to a principal. When the target perceives an association initiation attempt, it engages in a similar exchange with the KDC to retrieve the public key of the initiator.

Both conventional and public-key KDCs require an initial "face-to-face meeting"

[28] For ease of presentation, this section is written as if interactions took place directly between a principal and the KDC. In reality, the interaction is between the KDC and a protocol entity of the layer that provides communication security services.

[29] This is but one of several possible ways to structure principal/KDC interactions. The working key copy, enciphered under the master key of the target, could be sent to the initiator and then forwarded to the target as part of the first association initiation message. This approach eliminates synchronization concerns from the key distribution mechanism, but it may not be appropriate for symmetric protocols in which either side can originate the association initiation process.

with a representative of each principal they serve.[30] At this meeting, the principal's representative first establishes his identity with the KDC. Then, if the meeting is with a conventional KDC, he receives (or presents) his master key. Otherwise, he presents his public key and receives, in turn, the public key of the KDC. Note that while both procedures involve a tamper-free key exchange, only the former requires secrecy in the exchange as well.

Once this initial meeting has taken place, a principal can establish an association with any other principal with the assistance of the KDC, as described above. Note that a response message from the KDC must be reliably bound to the request that generated it. It can be so bound by having the request contain a unique identifier which the KDC then attaches to the associated response. Failure to employ such a mechanism exposes the principal to attacks in which recordings of prior principal/KDC interactions are played back, possibly resulting in the use of old, exposed keys and subsequent masquerading and/or data release. The preceding discussion assumes that the KDC and the principal communicate using a transaction-oriented mechanism. Another approach to this problem would be to maintain a secure connection between the principal and the KDC. A detailed description of such a KDC/transport-entity protocol is presented in Voydock 1981.

As hinted above, a major advantage of KDCs is that they allow master keys or public keys to be changed with relative ease. A principal must change his master or public key through a procedure equivalent to the initial principal/KDC meeting. However, once this new key has been placed in the KDC, it will automatically be employed in all future association initiation attempts. If a less centralized key distribution approach is employed, notifying all principals affected by a key change can be a time consuming and difficult task. Until it receives such notification, a principal will continue to use the old, possibly exposed, key.

Successful attacks on a KDC can cause serious problems. For example, release of a master key from a conventional KDC results in potential exposure of all working keys distributed under that master key, and thus all data encrypted under those working keys. It also permits an intruder to pose as the principal whose key has been compromised, or to pose as the KDC in interactions with that principal. Modification of entries in either a conventional or public-key KDC permits an intruder to pose as the principal whose entry has been modified.

Release of public keys in a KDC has no ill effects since these keys are freely provided to requesters in any case. However if the secret key of the KDC is compromised, an intruder can pose as the KDC, and transitively, as any principal. This can result in release of information, as well as masquerading. If the secret key of the KDC is compromised, a new KDC public key must be distributed to all principals, a formidable task requiring a face-to-face meeting between the KDC and each principal. The face-to-face meeting could be eliminated if a way were found to validate open announcements of new KDC public keys. The problem with open announcements is the difficulty of preventing an imposter from making a public announcement of a bogus key, or causing such an announcement to be made.

This section has simply touched on some of the issues involved in designing KDCs. Some of these issues are further discussed in Heinrich 1978.

8.2 Detecting Playback Attacks

The second type of spurious association initiation attack is to play back a recording of a previous legitimate association initiation sequence. Counteracting such attacks involves verifying that the association initiation attempt is being made in real time.

One way to do this is to make use of a challenge–response mechanism similar to

[30] Because a principal may not necessarily be a person, this paper refers to meetings between a KDC and the *representative* of a principal. Also, depending on the security requirements of the network, the face-to-face meeting may be replaced by some equivalent procedure, such as the use of a bonded courier or registered mail.

the request–response mechanism discussed in Section 7.[31] This mechanism is employed after the two sides have agreed on a key to use, but before the transmission of user data begins. At this point, each side sends the other an encrypted challenge PDU containing a unique bit pattern (perhaps a real-time clock reading). The other side must send an encrypted response PDU containing a value that is some predefined transformation of the unique bit pattern contained in the challenge. When both pairs of challenge–response PDUs have been exchanged, each side knows that the other has replied in real time.

Note that the above mechanism involves the exchange of four PDUs over and above those needed to establish the key to be used. By integrating this mechanism with the key distribution mechanism, the total number of messages involved can be reduced. How this integration would be done is intimately related to the details of the particular key distribution mechanism employed [Needham and Schroeder 1978].

9. SUMMARY

Threats to communication security fall into two general categories: passive attacks and active attacks. The purpose of a passive attack is to bring about the unauthorized release of information; the purpose of an active attack is to cause either unauthorized modification of information or unauthorized denial of use of resources. The two types of attacks are duals; that is, passive attacks usually cannot be detected but can be effectively prevented, whereas active attacks cannot be prevented but can be reliably detected.

There are two basic approaches to communication security: link-oriented security measures and end-to-end security measures. The former measures provide security by protecting message traffic independently on each communication link, whereas the latter provide uniform protection for each message from its source to its destination. End-to-end measures seem to be more appropriate for use in an open-system environment.

Data encryption is the fundamental technique on which all communications security measures are based. Encryption directly prevents passive attacks by preventing an intruder from observing data in the clear. Data patterns can be masked by using a unique key for each association and by exercising care in the selection of IVs. The protocol layer in which encryption is performed determines the precision with which traffic analysis can be done.

Active attacks fall into three categories: message-stream modification (MSM), denial of message service, and spurious association initiation. All of these attacks can be detected by using an encryption algorithm with appropriate error propagation characteristics.

Measures to detect the three forms of MSM attacks are hierarchically organized. The most fundamental measures are those that ensure message integrity. Measures that ensure message authenticity rely on the integrity measures, and measures that ensure message ordering, in turn, rely on both of the previous measures. MSM countermeasures are based on the use of a unique key for each association, a unique sequence number for each PDU, and an error detection code that can be inalterably bound to each PDU.

Denial of message service is an important type of active attack that is often overlooked. MSM countermeasures can detect some, but not all, forms of this attack. To detect denial-of-message-service attacks that begin when an association is quiescent, some form of request–response mechanism must be employed.

Spurious association attacks take two forms: attempting to establish an association under a false identity, and playing back a recording of a previous legitimate association-initiation attempt. To counteract the first kind of attack, an association must be initiated in a way that supports secure identification of the principals at each end. This can be done through hierarchic key distribution. To counteract the second kind of

[31] Of course, a protocol may already incorporate a suitable association initiation sequence which obviates the need for an explicit challenge–response mechanism [Sunshine 1980].

attack, association initiation must include a mechanism that verifies that the initiation attempt is being made in real time. One way to do this is by using a challenge-response mechanism similar to the request-response mechanism used to detect denial of message service attacks.

ACKNOWLEDGMENTS

Douglas Hunt and Gregory Pearson made numerous useful suggestions and comments throughout the preparation of this paper. Dennis Branstad and James Moulton, contract monitors for the NBS contract, provided useful feedback during preparation of the manuscript. Rachel Rutherford made numerous suggestions that significantly improved the readability of this paper.

The work described in this paper was sponsored in part by the Institute for Computer Sciences and Technology of the National Bureau of Standards under Contract SB80NBS0015.

REFERENCES

ABRAMSON, N. 1970 The ALOHA system—Another alternative for computer communications. In *Proc AFIPS Fall Jt Computer Conf*, vol. 37. AFIPS Press, Arlington, Va., pp 281–285

ANDERSON, J 1972. Computer security technology planning study, vols 1 and 2 ESD-TR-73-51, AF Electronic Systems Division, Hanscom Air Force Base, Mass.

BARAN, P 1964. *Distributed Communications* Vol. 9, *Security, Secrecy, and Tamperfree Considerations,* RM-3765-PR Rand Corp

BRANSTAD, D. 1973. Security aspects of computer networks. In *Proc AIAA Computer Network Systems Conf* (Huntsville, Ala., Apr).

BRANSTAD, D. K 1975 Encryption protection in computer data communications In *Proc 4th Data Communications Symp* (Quebec, Oct 7–9) IEEE, New York, pp 8-1–8-7

BURRUSS, J., ET AL 1981. Specification of the transport protocol BBN Rep. 4590, Bolt, Beranek and Newman, Cambridge, Mass.

CERF, V , AND KAHN, R 1974 A protocol for packet network intercommunication *IEEE Trans Commun* **COM-22,** 5 (May), 637–648

CERF, V., AND KIRSTEIN, P T. 1978. Issues in packet-network interconnection. *Proc IEEE* (Nov).

DEFENSE COMMUNICATIONS AGENCY. 1982. Defense data network program plan.

DIFFIE, W., AND HELLMAN, M. 1976. New directions in cryptography. *IEEE Trans Inf Theory* **IT-22,** 16 (Nov.), 644–654.

EHRSAM, W., MATYAS, S., MEYER, C., AND TUCHMAN, W. 1978. A cryptographic key management scheme for implementing the Data Encryption Standard. *IBM Syst. J.* **17,** 2, 106–125.

FARBER, D., AND LARSON, K. 1972. The structure of a distributed computer system—Communications. In *Proc. Symp. Computer-Communications Networks and Teletraffic* (Apr.). Microwave Research Institute of Polytechnic Institute of Brooklyn, Brooklyn, N.Y.

FEISTEL, H., NOTZ, W., AND SMITH, J. 1975. Cryptographic techniques for machine to machine data communications. *Proc IEEE* **63,** 11 (Nov.), 1545–1554

HEINRICH, F. 1978. The network security center: A system level approach to computer network security, vol. 2. NBS Special Publ. 500-21, Government Printing Office, Washington, D.C.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION 1980. Data processing—Open systems interconnection—Basic reference model. ISO/TC97/SC16 N537, rev Available from American National Standards Institute, New York.

JUENEMAN, R., ET AL. 1983. Authentication with manipulation detection code. In *Proc IEEE 1983 Symp. Security and Privacy* (Apr.). IEEE, New York.

KAHN, D 1967 *The Code Breakers.* Macmillan, New York.

KAHN, R E 1967. The organization of computer resources into a packet radio network. In *Proc Nat. Computer Conf.*, vol. 44. AFIPS Press, Arlington, Va., pp. 177–186.

KARGER, P 1977. Non-discretionary access control for decentralized computing systems. LCS-TR-179, MIT Laboratory for Computer Science, Cambridge, Mass.

KENT, S. 1976. Encryption-based protection protocols for interactive user-computer communication LCS-TR-162, MIT Laboratory for Computer Science, Cambridge, Mass.

KENT, S. T. 1977. Encryption-based protection for interactive user/computer communication. In *Proc. 5th Data Communications Symp* (Snowbird, Utah, Sept. 27–29) IEEE, New York, pp. 5-7–5-13.

MATYAS, S , AND MEYER, C 1978. Generation, distribution, and installation of cryptographic keys. *IBM Syst J* **17,** 2

MERKLE, R., AND HELLMAN, M. 1978. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans Inf. Theory* **IT-24,** 5 (Sept.), 525–530.

MERKLE, R., AND HELLMAN, M. 1981. On the security of multiple encryption *Commun. ACM* **24,** 7 (July), 465–467.

METCALFE, R. M., AND BOGGS, D. R. 1976. Ethernet. Distributed packet switching for local computer networks. *Commun ACM* **19,** 7 (July), 395–404.

MEYER, C., AND TUCHMAN, W. 1972. Pseudorandom codes can be cracked. *Electron Des.* (Nov.).

NATIONAL BUREAU OF STANDARDS. 1977. Data Encryption Standard. Federal Information Processing Standards Publ. 46, Government Printing Office, Washington, D.C.

NATIONAL BUREAU OF STANDARDS. 1980a. DES modes of operation. Preliminary Copy of Federal Information Processing Standards Publ., Government Printing Office, Washington, D C.

NATIONAL BUREAU OF STANDARDS. 1980b. Telecommunications: Interoperability and security requirements for use of the Data Encryption Standard in data communication systems. Proposed Federal Standard 1026, Government Printing Office, Washington, D.C.

NEEDHAM, R M., AND SCHROEDER, M. D. 1978 Using encryption for authentication in large networks of computers. *Commun ACM* **21**, 12 (Dec.), 993–998.

PETERSON, W., AND WELDON, E. 1972. *Error Correcting Codes* 2d ed MIT Press, Cambridge, Mass.

POUZIN, L. 1973. Presentation and major design aspects of the Cyclades computer network In *Proc 3rd Data Communications Symp* (Nov.)

RIVEST, R L, SHAMIR, A, AND ADLEMAN, L 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun ACM* **21**, 2 (Feb.), 120–126.

SALTZER, J., AND SCHROEDER, M. 1975. The protection of information in computer systems. *Proc. IEEE* **63**, 9 (Sept.), 1278–1308.

SAVAGE, J 1967 Some simple self-synchronizing data scramblers. *Bell Syst. Tech J* **42**, 2 (Feb.), 449–487

SHANNON, C 1949 Communication theory of secrecy systems. *Bell Syst Tech. J* **28**, 4 (Oct.), 656–715

SUNSHINE, C 1980. Transport protocols for computer networks In *Protocols and Techniques for Data Communications Networks*. Prentice-Hall, Englewood Cliffs, N.J.

TELENET COMMUNICATIONS CORPORATION 1975. Host interface specifications. Washington, D.C

VOYDOCK, V, AND KENT, S. 1981. Security in higher level protocols: Approaches, alternatives, and recommendations. BBN Rep. 4767, Bolt, Beranek, and Newman, Cambridge, Mass. Also available as National Bureau of Standards 1981, Rep ICST/HLNP—81-19, Government Printing Office, Washington, D.C