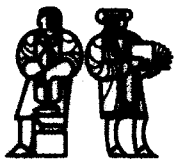
LABORATORY FOR
COMPUTER SCIENCE
(formerly Project MAC)



MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TR-162

ENCRYPTION-BASED PROTECTION PROTOCOLS FOR INTERACTIVE USER-COMPUTER COMMUNICATION

Stephen T. Kent

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

This blank page was inserted to preserve pagination.

MIT/LCS/TR-162

ENCRYPTION-BASED PROTECTION PROTOCOLS FOR INTERACTIVE USER-COMPUTER COMMUNICATION

Stephen Thomas Kent

May 1976

This research was sponsored in part by the National Science Foundation, through a graduate fellowship, and in part by the Advanced Research Projects Agency (ARPA) of the Department of Defense under ARPA order No. 2095 which was monitered by the Office of Naval Research under contract No. N00014-75-C-0661.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LABORATORY FOR COMPUTER SCIENCE
(formerly Project MAC)

CAMBRIDGE MASSACHUSETTS 02139

ENCRYPTION-BASED PROTECTION PROTOCOLS FOR INTERACTIVE USER-COMPUTER COMMUNICATION *

by

Stephen Thomas Kent

ABSTRACT

This thesis develops a complete set of protocols, which utilize a block cipher, e.g., the NBS data encryption standard, for protecting interactive user-computer communication over physically unsecured channels. The use of the block cipher protects against disclosure of message contents to an intruder, and the protocols provide for the detection of message stream modification and denial of message service by an intruder. The protocols include facilities for key distribution, two-way login authentication, resynchronization following channel disruption, and expedition of high priority messages. The thesis presents designs for modules to implement the protocols, both in a terminal and in a host computer system, and discusses the results of a test implementation of the modules on Multics.

Thesis Supervisor: Michael D. Schroeder

---

*This report is based upon a thesis of a similar title submitted to the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, on May 19, 1976 in partial fulfillment of the requirements for the degree of Master of Science.

DEDICATION

To the memory of Chris Hasenkampf, a close friend and colleague.

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor, Professor M. D. Schroeder, for his guidance in the research reported on in this thesis. He contributed immensely to the final form of this thesis, through tireless editing and substantial insight into the fundamental problems that were treated in this document. I would also like to thank Professor J. H. Saltzer for having suggested this line of research and for his continuing advice. I wish to extend my appreciation to Dr. David Clark and Doug Wells who have read portions of this thesis and have helped me through their thoughtful criticism and patient explanation of concepts and operating features of communication systems. Also, I want to express by thanks to all who have aided me in this work through countless discussions of various aspects of this research, especially Allen Luniewski, David Reed, Ken Pogran, Phil Janson, and Professor Ron Rivest.

I wish to extend a special thanks to my wife, Rachel, for her compassion, patience, proofreading, and general assistance during the preparation of this thesis.

This research was conducted in the Computer Science Research Division of the Massachusetts Institute of Technology Laboratory for Computer Science. It was sponsored in part by the National Science Foundation, through a graduate fellowship, and in part by the Advanced Research Projects Agency (ARPA) of the Department of Defense under ARPA order No. 2095 which was monitered by the Office of Naval Research under contract No. N00014-75-C-0661.

TABLE OF CONTENTS

LIST OF FIGURES

This empty page was substituted for a blank page in the original document.

Chapter One

Introduction

This thesis develops protocols to organize the use of encryption to deal with the problem of providing a secure communication path between a user at a terminal and his computation in a remote host computer system. This problem is of major concern as more and more computing is performed interactively via unsecured communication facilities and the value and importance of the data so accessed increases. Secure communication is no longer a concern just for the military. With the introduction of a standard encryption algorithm [NBS] that can be implemented on a single integrated circuit chip, and with the decreasing costs of hardware components, it is now practical to consider using encryption-based measures to protect data enroute from a user terminal to a remote host facility.

Assuming the existence of an intruder, armed with a large scale computer positioned in the connection between a user terminal and a remote host computer, a number of different types of threats may be posed. The intruder may not only passively copy each message transmitted in either direction on the connection, but he may actively disrupt the flow of messages on the connection, modifying, delaying, reordering, and rerouting messages or synthesizing new messages and inserting them into the connection. As the communication path is assumed to be physically unsecured, there is no way that an intruder can be prevented from engaging in such acts, but the protection

measures developed in the thesis do prevent disclosure of message contents, provide detection of message stream modification, and provide detection of denial of message service.

The use of encryption protects against disclosure of the contents of the messages being transmitted on the connection. It also serves to bind together the user level data and a tag that identifies messages, so that an intruder cannot, with a high probability, modify user level data without detectably modifying the tag. The use of such a tag in all messages provides a basis for establishing the authenticity of each message received on the connection. The design of the tag prevents any undetected reordering, deletion, or rerouting of unmodified messages on the connection. It also provides for the highly probable detection of spurious or modified messages introduced into the connection. Protocols are provided, employing special control messages, to distribute encryption keys on the connection, detect intruder attacks involving delay or destruction of message traffic, and resynchronize both ends of the connection in the event of disruption. A protocol also is employed for the secure handling of high priority messages on the connection.

The thesis presents a design for the protection modules needed at both ends of the connection to implement the protocols. At the terminal end, the protection module is simple enough for it to be constructed using a general purpose microprocessor and a special purpose chip for enciphering and deciphering operations. At the host end, the the protection module is constructed in software within the host computer. The only special hardware support assumed for the host module is a machine instruction for performing enciphering and deciphering of message blocks, perhaps using the same chip.

The preferred positioning of the protection modules relative to the various hardware and software facilities typical of existing computer communication system is discussed.

In order to test the completeness of the protection measures designed in this thesis and evaluate their impact on the human interface of a computer utility, a test implementation was carried out on the Multics [MIT] system. Experience with this test implementation indicates that the modules do detect intruder acts resulting in message stream modification or denial of message service and mitigate the impact of connection disruption on the interface presented to the user. The performance degradation resulting from use of the modules, assuming hardware support for the encryption/decryption algorithm, should be negligible for most users.

Related Work

As this thesis is not primarily concerned with cryptographic systems, the work of such people as Kahn [KD1, KD2] and Shannon [Sha] is only indirectly related. It may be the case that work similar in nature to that reported on in this thesis has been carried out by researchers within the Department of Defense, but because such work would be classified I am not aware of it.

In the open literature a number of papers have dealt with the use of encryption for protection of data communicated via physically unsecured channels [Bar, Sav, ScP, Tur]. In particular the work of Paul Baran at Rand [Bar] stands out as an example of a major, systematic study of the problems involved in securing military data communication networks. This study, like others in the area, takes the view of providing secure communication

facilities for a variety of purposes other than user communication with computation in remote host computers. It also places emphasis on protecting the communication system from the threat of traffic analysis, unlike this thesis, and thus assumes the existence of relatively secure intermediate nodes in the communication network to provide link encryption of messages, in addition to end-to-end encryption. A fundamental difference between work of this sort and the thesis is that the former treates the problem as one of securing communication facilities, rather than as a one of providing a secure virtual connection between a user and his computation executing in a remote host computer.

Several papers were generated at IBM in the early seventies, by Horst Feistel et al. [FH1, FH2, FNS, Smi, SNO], dealing with the development of the Lucifer encryption algorithm and its application to remote terminal to host communication systems and to remotely accessed databases. These papers discussed the design of Lucifer and presented a simple protocol for use over half-duplex channels. That work is much closer to the body of this thesis, than the works noted above, in terms of its intended application. However, the protocols described in the IBM papers are suited only for use in half-duplex communication environments and do not treat all of the protection problems, e.g., automatic detection by the host of connection blocking by an intruder and secure transmission of high priority messages, that arise when the encryption protection mechanisms are used for general purpose interactive computing, as opposed to database accessing. Furthermore, the coupling of the encryption protection measures with database accessing seems to violate concepts of procedural layering of system functions. This violation seems to

be a result of trying to use the encryption protection mechanisms to overcome deficiencies in the internal protection mechanisms of the host computer used in these experiments.

More recently, Dennis Branstad, of the National Bureau of Standards, has proposed some protocols for use in authentication, host access control, and distribution of working keys in a network environment [Bra1, Bra2]. Branstad's work does not develop protocols to deal with problems such as message sequencing, automatic resynchronization, and high priority message processing. The protocols proposed by Branstad are described in terms of a particular network environment that does not encompass simple dialup lines of the type used to access many interactive host computers today. The protocols described in this thesis can be used in either a general network or simple dialup environment. Further suggestions for protocols to organize the use of the National Bureau of Standards data encryption standard are expected to be forthcoming shortly from NBS and from other researchers.

Outline of Thesis

Chapter two presents the model of the terminal-host connection that is used in the thesis, and develops the protection goals that characterize the security that can be provided for a physically unsecured connection. The chapter then presents characteristics of cryptographic systems that make them suitable for protecting interactive user-computer communication and selects the NBS data encryption standard as the basis for implementation of the protection protocols.

Chapter three develops an authentication scheme for messages in a full-duplex communication environment. The chapter also deals with protocols for the distribution of keys in support of the authentication mechanism, and presents a protocol for the secure initialization of the channel at login time.

Chapter four develops protection measures for detection of denial of service, when effected by blocking of message traffic on the connection. The chapter also discusses protocols that are used to restore synchrony of the message counters used for authentication on the channel.

Chapter five discusses high priority messages, e.g., "attention" signals. An extension to the connection model developed in chapter two is presented to support high priority messages transmitted from the terminal to the host. A protocol is introduced for handling such messages within the protection framework provided for regular message communication.

Chapter six investigates the factors that influence the positioning of the encryption protection modules in the communication path between a user's terminal and his computation. The primary factors that influence this positioning are security and functionality constraints. Differences in host communication system architectures that are relevant to protection module positioning, especially with respect to support of high priority messages and character echoing, are examined.

Chapter seven presents a detailed discussion of the control structure of both the terminal and host protection modules. The modules are characterized in terms of finite state machines driven by inputs from the user terminal, the user's process, the ciphertext connection and timeouts at the host module.

Chapter eight discusses the test implementation of the proposed protocols undertaken on the Multics system. Some of the design issues associated with actually incorporating a host protection module in a production Multics system are considered. A discussion of the impact of the protection protocols upon the performance of the user-host connection and the host overhead to support the protection protocols is presented.

Chapter nine reviews the conclusions of the thesis and proposes topics for further study, including construction of production terminal and host protection modules, further performance evaluation, and generation of encryption keys.

The appendix discusses the susceptability of the Lucifer and NBS ciphers to a particular form of cryptanalysis, exhaustive key searching with matching intercepted cleartext and ciphertext. Recent research [DH1] indicates that this form of cryptanalysis may be a practical means of attacking the NBS cipher, but that the Lucifer cipher is resistant to such an attack.

Chapter Two

Protection Goals and Encryption

In order to discuss the protection problems associated with physically unsecured communication channels, this chapter presents a model of a terminal-host connection, complete with intruder, and examines specific examples of intruder threats. From this model, the realizable protection goals for such a connection are established. Next, encryption is introduced as a basis for meeting these goals. The thesis does not involve the details of cryptographic systems or cryptanalysis. Rather, cryptographic systems are viewed as "black boxes" that exhibit certain properties germane to providing a secure communication path between a user and a remote host computer. The chapter concludes by discussing the properties that make a cryptographic system suitable for this application and that influence the design of the high-level synchronization and authentication protocols developed in later chapters.

The Terminal-Host Connection Model

For generality, we consider a full-duplex connection between a user terminal and a computer utility. Such a connection has the property that messages may be transmitted in both directions simultaneously. We can further simplify this description by modeling the full-duplex connection as a pair of independent simplex channels, each capable of transmitting messages in one direction only. At this time we shall ignore the physical details of the

connection. Thus, such equipment as line adaptors, modems, front end processors and possible intermediate switching nodes will not be considered here, but will be discussed in chapter six. Rather, we shall identify only three parts of the connection as being of interest at this time: the terminal terminal, the host, and an intruder.

Both the terminal and the host are presumed to reside in secure areas. The terminal may be used at different times by various users with different security requirements and different authorization levels. The host may also provide services to a diverse user community, not all of whom will employ the protection measures described in this thesis.

The intruder will be represented by a large computer, under hostile control, situated in the connection between the terminal and the host. All messages transmitted in either direction on the connection must pass through the intruder. The intruder can perform any processing he desires on the messages-- copying them, delaying them, absorbing them, modifying them, synthesizing new messages or allowing them to pass transparently. Figure 2-1 describes this configuration.

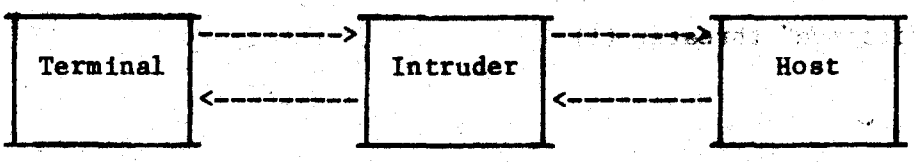


Figure 2-1

General Model of a Full-Duplex Connection with Intruder

Protection Goals

We would like to transmit messages in both directions in a way that makes the presence of the intruder irrelevant to the security of the connection. However, as the model suggests, with a physically unsecure connection the intruder could absorb some or all message traffic in his computer. In a less drastic action, the intruder could delay all message traffic in either or both directions. Acts of this nature can be termed "denial of message service" threats. In our model, with all messages on the connection passing through the intruder's computer, it is not possible to prevent denial of message service and we shall not address the more general problem of countering such threats.

Similarly, as our model suggests, it is not possible to prevent the modification of a message transmitted over the connection or the introduction of a spurious message. Included in the set of spurious messages are not only bit strings constructed by the intruder, but also messages previously intercepted by the intruder. Acts such as these can be designated as "message stream modification" threats. (1)

---

(1) One may also term acts of this nature "active" wiretapping threats, in contrast to "passive" wiretapping threats that involve no intervention in the transmission of message traffic but merely involve listening in on the conversation.

With these limitations in mind, we can establish three goals for protection measures applied to a physically unsecured connection:

1. Prevention of release of message contents
2. Detection of message stream modification
3. Detection of denial of message service

We will now examine various intruder threats to determine what form of protection measures are required to achieve these goals. (2)

Encryption techniques have been used primarily as countermeasures to threats of message contents disclosure [KD2]. By enciphering messages transmitted between the terminal and the host, this first goal can be achieved within the limitations of the enciphering scheme used and subject to security violations external to our model, e.g., the loss of the key by the user. The enciphering is controlled by a key held by both the user and the host, and the ability to decipher a message is based exclusively on possession of the key. Modifying our earlier terminal-host connection model to include an encryption protection module (EPM) at the terminal end and suitable encryption facilities at the host end results in the configuration shown in Figure 2-2. The protocols used to establish an enciphered communication path between the

---

(2) A form of intruder threat that does not fall within these three categories is referred to as traffic analysis. This passive threat involves analysis of patterns of message traffic, or examination of address headers in multiplexed channels, without actually reading the contents of the multiplexed channels, in an effort to determine the nature of the conversation taking place. Countermeasures against traffic analysis threats usually involve the generation of "dummy" messages at each end of the connection in order to maintain a constant rate of message traffic and link-to-link encryption of messages to prevent an intruder from reading message headers. Although the protocols developed in the thesis will support such additional countermeasures, threats of this type will not be treated.

terminal and the host computer, by exchanging messages enciphered with the same key, are discussed in chapter three.

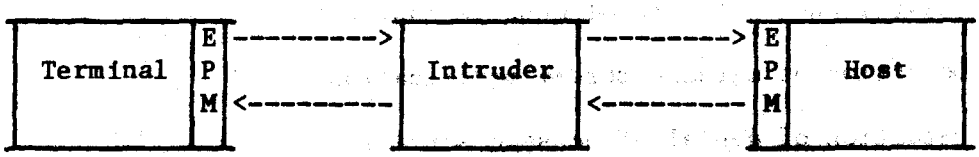


Figure 2-2

Connection Model with Encryption Protection Modules

In order to achieve the second goal noted above, detection of message stream modification, some mechanism must be employed that permits a message to be verified as authentic. In this context authenticity implies not only that the message received was sent by the other end of the connection, but further that the message is the next one in the sequence of messages currently being transmitted. By associating with each message a tag that is then enciphered along with the message, the problem of message authentication can be attacked. Chapter three proposes a scheme for tagging messages that is the basis of a simple authentication technique for use in a full-duplex communication environment.

In order to achieve the third goal, detection of denial of message service, request-response protocols will be introduced to permit automatic, time-controlled monitoring of the integrity of the connection by the host. These protocols will be developed in chapter four.

The protection measures used in this thesis to achieve all three goals are based on encryption. As well as masking the user-level data from the intruder, encryption indivisibly binds the data to the control information

required to achieve the other two goals. We now shall examine some properties of cryptographic systems to determine which systems are suitable for this application and to develop an understanding of the nature of the security provided by encryption.

Terminology

A cipher is an algorithmic transformation performed on a symbol-by-symbol basis on any data. Although there are technical distinctions between the terms encipherment and encryption [KD2, Sha], the two terms will be used interchangeably throughout this thesis to refer 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 while the output from the algorithm is designated as ciphertext. The transformation performed on the cleartext to encipher it is controlled by a key. To be of use in a communications context, there must also exist a matching decryption algorithm that reverses the encryption transformation when presented with the same key. Figure 2-3 shows the general form of such a cryptographic system.

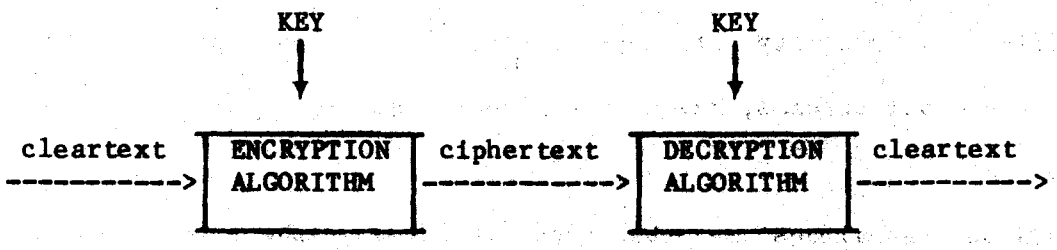


Figure 2-3

"Black Box" Model of a Cryptographic System

Two major classes of encryption techniques that have been used in modern, non-voice telecommunications and digital computer applications are stream and block encryption. The former method performs bit-by-bit transformations on the cleartext under the control of a stream of key bits, usually using some easily reversible operation, e.g., addition modulo 2. The latter method enciphers fixed-sized blocks of bits under the control of a key that is frequently the same size as, or somewhat larger than, the blocks being encrypted.

Stream Ciphers

Stream ciphers have an advantage that they can operate on a stream of cleartext in real time, enciphering each bit as it is generated by combining it with a bit from a key stream. A stream cipher in which the key stream consists of random bits as long as the combined length of all messages that are ever to be transmitted using this stream, a Vernam cipher, constitutes an unbreakable cipher [KD1, KD2, Sha]. In practice, the volume of communication traffic and the logistic difficulties associated with providing each user with a sufficient quantity of keys cause most stream ciphers to utilize pseudo-random bit streams, based on a fixed-length key, that have very long periods.

Various techniques may be used in stream ciphers to generate the key stream. The source of these bits may be completely independent of the cleartext stream, e.g., a pseudo-random number generator primed with a small initial key or a tape that is to be used only once. With such an independent key stream, changes to individual bits in the ciphertext do not propagate to

other portions of the ciphertext stream. This is an advantage in that transmission errors that alter the values of bits of the ciphertext do not affect the ability of the receiver to correctly decipher subsequent transmissions. (3) This characteristic is a disadvantage in constructing message stream control protocols because it fails to bind together user-level data and control information.

Stream ciphers can also be constructed in which the key stream is a function of the cleartext or ciphertext and uses some initial, "priming" key [Sha]. Ciphers employing this approach achieve interbit dependence that can be used to detect errors in transmitted ciphertext, as such errors interfere with the correct decipherment of subsequent transmissions. Transmitted ciphertext can also be used as input to key stream generation in self-synchronizing ciphers that achieve interbit dependence but resume correct operation following transmission errors, after some fixed number of unaffected bits are received [Sav]. Even with the use of self-synchronizing stream ciphers, an error in the received ciphertext may result in damage to multiple messages.

Block Ciphers

In contrast to stream ciphers, block ciphers transform entire blocks of bits under the control of a key. 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) is 2^n.

---

(3) Undetected insertion or removal of bits from the ciphertext stream results in a loss of deciphering ability in ciphers of this sort.

A block cipher maps the space of cleartext blocks into the space of ciphertext blocks. In order that the deciphering of a block yield an unambiguous cleartext block the mappings must be reversible, hence one-to-one and, in this case, onto, because the sizes of the spaces are equal. Thus, we can view a block cipher under the control of a single key as defining a permutation on the set of n-bit blocks. There are $(2^n)!$ distinct permutations on the set of n-bit blocks. In practice it is not feasible to implement a block cipher that realizes all of the possible permutations because of the size of the key required and the logical complexity of the cipher. In the block ciphers we shall discuss, only a small fraction of the permutations, e.g., on the order of the size of the text spaces, is used.

For all values of n, the block size, a block cipher is equivalent to the classical "simple substitution" cipher, and when n is 7 or 8 the block corresponds to a single character from some small alphabet and this equivalence becomes very apparent. This system is known to be very weak, not because of the structure of the system, but because of the small size of the blocks usually used. The cipher is subject to analysis of the frequency distribution of individual blocks, for comparison with the known frequency distribution of characters in large samples of cleartext. By increasing the size of the block so that n is on the order of 50 or 100, and by constructing the cipher so that the frequency characteristics of the components of the block are concealed, such frequency distribution analysis becomes infeasible because the size of the effective alphabet has been increased to 2^{50} or 2^{100}, and the resulting cryptographic system is very good.

The Lucifer system developed at IBM is an example of a block cipher scheme using 128-bit blocks and equal size keys [FH1, FH2, FNS, Smi, SNO]. Each bit of ciphertext in a block generated by the Lucifer algorithm is a function of each bit of the key and each bit of the cleartext block. A difference of only one 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 one bit of either the key or the ciphertext will result in changes in an average of 50% of the bits of the deciphered cleartext.

Because of this sensitivity of the block to modification, the inclusion of a k bit error detection (or identification) field in a cleartext block provides a basis for detecting modification of the block with a probability of undetected error of $1/(2^k)$. This means that any error in a block propagates within the block to such an extent that its detection can be made extremely likely, yet subsequent blocks are unaffected by the error. Feistel claims that because this interbit dependence within a block is functionally non-linear, it is difficult to use the dependence as an aid in deciphering the blocks [FNS].

For block ciphers, synchrony of the two ends of the communication channel is required only to the extent that each must load the same key and the blocks must be correctly delimited. Higher-level message stream synchronization, e.g., correct ordering of blocks, can be accomplished by protocols that use sequence numbers embedded within the blocks. Resynchronization at that level, we will demonstrate in chapter four, is possible without transmitting special unenciphered messages.

Choice of a Cipher Scheme

An encryption algorithm used for securing a user/computer communication channel must conceal the contents of transmissions and provide a basis for effectively implementing various authentication and synchronization protocols. While both stream and block ciphers can conceal effectively, block ciphers seem to provide a simpler to use basis for the protocols. In order to detect various intruder threats, the protocols associate with each message certain information that identifies the message as genuine. The encryption algorithm must bind together the user's messages and the protocol information so that any attempt to tamper with the message will be reflected in the protocol information. In the event of intrusion or error, the protocols should allow re-establishing higher-level message stream synchrony without going outside of the encryption scheme. These combined requirements appear to indicate that a block cipher similar to Lucifer would provide a natural basis for the development of the protection protocols, since it provides substantial interbit dependence in each block while limiting the impact of errors to single, well-defined blocks. (4)

A block encryption algorithm has been proposed as a Federal Information Processing Standard (FIPS) by the National Bureau of Standards [Bra2, NBS]. This algorithm operates on 64-bit blocks, uses a 64-bit key (5) and employs

---

(4) This should not be construed as an indication that stream ciphers, especially auto-synchronizing ones, cannot be used as the foundation for protocols similar to the ones presented in this thesis. Rather, block ciphers such as Lucifer appear to form a more natural basis for fixed-length message protocols of the type presented in this thesis.

(5) Although a 64-bit key is used with the NBS algorithm, only 56 bits of this key are actively used in the encryption algorithm and NBS has recommended that

many of the same design principles used in the 128-bit Lucifer. If this algorithm is adopted as a FIPS, it will probably become a de facto industry standard as well. Already software is being offered that performs the encryption as specified by this algorithm [Bri], and hardware implementations of the algorithm using a single large-scale integrated chip are being planned. Thus, the protection protocols and mechanisms developed in this thesis will be examined in the context of probable use of this encryption algorithm, although the protocols are not restricted to the particular block or key size associated with the NBS proposed standard.

Although this cipher appears resistant to cryptanalysis, recent work by Diffie and Hellman [DH1] indicates that automated, exhaustive searching of the key space is not unreasonable for an analyst provided with adequate resources and small amounts of intercepted ciphertext and partial matching cleartext. This thesis is not concerned with the topic of cryptanalysis and assumes that the cipher scheme used as the foundation for the protection protocols is resistant to cryptanalytic attacks. In order to better understand the nature of the weakness noted by Diffie and Hellman, the appendix contains a brief discussion of exhaustive searching of the key space in the case of the Lucifer and NBS ciphers. In chapter three we shall note, in some instances, how this characteristic of the NBS cipher might affect the protection protocols.

Summary

This chapter presented a model of a physically unsecured terminal-host connection and established goals for the protection that we shall attempt to

---

the remaining eight bits act as parity bits to be utilized for error detection in key generation, distribution and storage [NBS].

provide through the use of encryption and the protocols developed in later chapters. We have examined some properties of cryptographic systems and have chosen a particular block cipher as the basis for the development of protection protocols. This type of cipher is well suited to the application because of the high degree of interbit dependence it provides for each block and because of the independence of each block with respect to propagation of errors.

A specific example of this type of cipher has been proposed as a Federal Information Processing Standard and, if adopted, will provide a broad basis for exchange of encrypted information. Thus, we will adopt it as the basic cryptographic system upon which further protection mechanisms will be constructed. However, the protocols presented in this thesis can be used with other block encryption schemes that provide suitable cryptographic protection.

Chapter Three

Message Stream Authentication

Having chosen, in chapter two, Lucifer-style block ciphers as the basis for implementing protection protocols, this chapter presents a simple scheme for authenticating messages that uses the properties of such ciphers. This authentication scheme achieves the goal of detection of message stream modification through independent message sequence numbering on each channel. This chapter also presents a protocol for changing keys that supports the message authentication scheme and that serves as a basis for a time-dependent, two-way authentication login protocol. The message authentication scheme further serves as the foundation for protocols that detect denial of service and that resynchronize the connection following disruption of communication. These last two protocols are presented in chapter four.

Message Modification Threats and Authentication

Part of the protocol information enciphered as part of each message to verify its authenticity is a tag. (1) Although there are a variety of forms that this authenticator tag may assume, (2) we are interested only in designs

---

(1) Although a logical unit of correspondence may be so large as to require several encrypted message blocks for its transmission, for simplicity the term "message" will refer to the logical contents of one block.

(2) For example, verification of a message may be based not on the knowledge of the exact bit pattern contained in the tag, but rather on the tag satisfying some computational or structural constraints, e.g., it may always contain twice as many "0" bits as "1" bits or it may be a cyclic redundancy check of the rest of the block contents.

that require the tag to consist of a bit pattern that must precisely match a pattern held by the receiver of the message. When used in a block enciphered with a Lucifer-type algorithm, such tags are optimal with respect to utilization of block space in that a $\underline{k}$ bit tag conveys precisely $\underline{k}$ bits of authentication information and can be forged by an intruder with probability of $1/(2^k)$.

It can be argued that such a tag is not necessary to the authentication process, especially when an encryption scheme with high degree of interbit dependence is being employed, since a spurious message would not decipher into meaningful cleartext. While this argument has some merit when considering messages received by the user at his terminal, it does not seem that most software systems exhibit a corresponding ability to make intelligent judgements as to the meaningfulness of messages. Moreover, messages directed to the user may admit to a wide range of "meaningful" contents when they represent answers to a virgin problem or consist of random numbers. Thus, we insist that authentication be based on the use of some form of message tagging.

To prevent an intruder from modifying a message and not the tag associated with it, it is necessary that the tag be attached to the message in such a manner that modification of any part of the encrypted block is very likely to result in modification of the message tag. The use of a block cipher system of the type discussed in chapter two, and placement of the tag in the message block achieve this desired result of message tag and message interdependence.

We shall distinguish two classes of message stream modification attacks: attacks that involve modification of genuine message blocks or synthesis of new blocks, and attacks that involve modification of the message stream through manipulation of genuine, intact blocks. Attacks of the first type can be detected because of the interdependence of the authenticator tag and the remainder of the block as noted above. In the latter category are acts such as deletion of blocks, insertion of copies of old blocks, and rerouting of blocks back to their sender. We will now discuss the design of an authenticator tag that permits detection of such attacks.

To detect these message stream modification attacks, we propose that each tag consist of a non-cycling bit pattern that is predictably recognizable by the receiver, logically chaining each message to its transmitted predecessor, and a bit identifying the origin of the message, the terminal or the host. We also require that if messages are removed or destroyed, examination of the tag on successfully received messages can be used to determine the number of messages so lost, for purposes of user notification, auditing, and possible higher level retransmission. Thus, this predictable sequence of patterns used in the tags must be capable of being mapped analytically into a strictly monotonic sequence that is dense in the integers. (3) Using this scheme, the receiver of a message is expecting a particular tag and any other tag will result in rejection of the message as spurious. Tags of this sort can be used

---

(3) Here we mean "analytically" in the sense that a receiver of messages should be able to compute the value of the tag that will appear in the ith message in the sequence using only his knowledge of the tagging scheme and the value of the first tag.

to perform the task of message authentication in conjunction with message sequencing and origin identification.

In the original Lucifer implementation, designed for use on half-duplex connections, Feistel proposed the use of message authentication tags [FH1, FNS]. The tag consists of bits from fixed positions in the last ciphertext block received, or from the last block transmitted if this is the first message in an incoming group, and thus was predictable by the receiver. Because half-duplex connections do not allow simultaneous transmission in both directions, this scheme can use this simple form of message chaining to authenticate message traffic in both directions. Since the tag bits used for chained authentication are a function of the contents of each previous message block, Feistel has argued that there is little chance of repetition, although there is no guarantee of this. Moreover, there is no apparent means for a receiver to ascertain the number of messages lost, should a message arrive out of sequence.

In light of the requirements set forth above for a tag design that enforces strict message sequencing and lost message accountability, it appears that consecutive numbering of messages, starting from zero, transmitted on each of the channels provides the simplest acceptable form of tag sequencing. (4) In order to fulfill the requirement of tag uniqueness (non-cycling tag sequence), the tag must be large enough to not "wrap around" during the lifetime of the key.

---

(4) The inclusion of the counter assures that each ciphertext block is different, even if the same text is transmitted multiple times. In situations where the blocks are used to transmit individual characters, this tag design prevents the cipher from becoming a weak substitution cipher on single characters.

Each end of the connection maintains two counters, one referring to the number of messages transmitted by that end of the connection and the other keeping track of the number of messages received. The transmission counter for a channel is used as the source of the sequence number portion of the tag for messages transmitted on that channel. (5) These counters must never cycle during the lifetime of the key and efforts should be made to insure that different connections have little chance of using the same key.

This tag design provides sure detection of any attempt to modify message traffic through rerouting, reordering or deletion of genuine messages on this connection. The design provides probabilistic detection of any attempt by an intruder to either synthesize a message block with an acceptable tag or to modify the contents of a genuine message block without affecting the tag. Using the Lucifer or NBS algorithm, the probability of erroneous authentication of a message modified in this fashion is no greater than $1/(2^k)$ if a k bit tag is employed. (6) Figure 3-1 illustrates this type of tag architecture. (The type field indicated is used to distinguish control messages associated with the protection protocols developed later.)

---

(5) This counter arrangement may also be viewed as associating two counters with each channel, recording the number of messages transmitted and received on that channel. This use of counters corresponds to the concept of eventcounts as described by Reed and Kanodia in [RK].

(6) If there is any predictability to the contents of the messages, the probability of erroneous authentication by the user is even lower as the intruder cannot systematically force "meaningful" user-level message contents either.

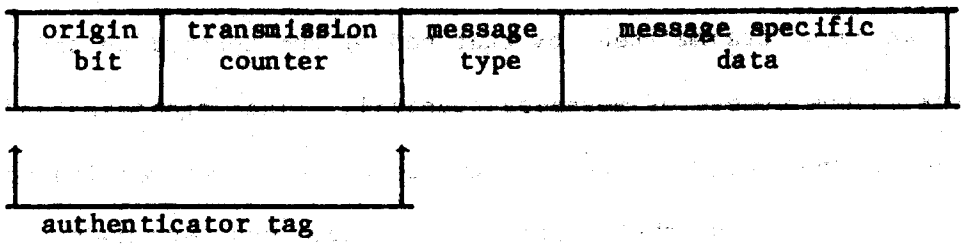


Figure 3-1

Generic Format of Message Blocks

A characteristic of both this tag scheme and the original Lucifer authentication technique is that they provide an intruder with the cleartext of a portion of each message block: the tag. We alluded to the nature of the problem in chapter two and the appendix provides a more detailed discussion of the subject. From the key searching discussion in the appendix, it is apparent that this knowledge alone is adequate for an intruder to determine the key that is being used by attempting to decipher several intercepted blocks under a single key and checking for a match on the tag field of all of the blocks. In the case of relatively small key spaces, like the NBS algorithm's 56-bit effective key, this may constitute a significant threat to the security of the system.

Although attempts could be made to overcome this problem in the tag scheme imposed above by concealing the tag, this is probably not worthwhile. (7) In fact, interactive user-computer dialogs tend to contain many messages

---

(7) The tag could be enciphered under a separate key using a block size equal to the tag size and then inserted in the message block and enciphered along with the message data. If the tag bits were the only portion of the block known to the intruder, this would substantially increase the work involved to break the key.

that are very predictable by a sophisticated intruder, e.g., stylized login and error response messages from the host. Because these messages contain adequate amounts of known information for an intruder to use in a key space search it appears that efforts to conceal the tag portion of a message for this reason are not fruitful. Rather, a cipher should be used for which exhaustive key searching is an impractical cryptanalytic technique.

Key Distribution Protocols

Because the tag value described above must never cycle, the tag must be large enough to uniquely identify the maximum number of messages that are to be transmitted over either of the channels during the lifetime of the key. Rather than having the size of the tag determined by the expected maximum message traffic volume on one of the channels over some extended time interval, e.g., a month, a year or the lifetime of the host system, it seems appropriate that the primary factors in determining the size of the tag should be the probabilistic degree of protection desired for the channel and the portion of the block capacity devoted to the tag. This motivates the concept of changing keys as a means of controlling the size of a tag.

If keys are randomly generated bit strings, then messages enciphered under one key effectively represent random bit strings when deciphered under a different key. Thus, messages enciphered under the control of a key different from the one currently in use on a connection pose no more of a threat than messages synthesized by an intruder using randomly generated bits. Moreover, if there is no easy way to use knowledge of a previous key to discover a key currently in use, or vice versa, the changing of keys establishes a "firewall"

around the data transmitted under each separate key. Thus, there is additional impetus to limit the lifetime of a key in order to minimize the volume of message traffic that would be compromised in the event a key is discovered.

If the key lifetime extends over more than one login session, then it is also necessary to be able to restore the counters used by both the terminal and the host so that the message tagging can resume from the point where it was terminated. (8) It is undesirable to require both ends of the connection to retain the values of the counters from the last login session for each user or to have the host retain these values and transmit them to the terminal in cleartext as part of an initialization procedure. These approaches are undesirable primarily because interactive sessions do not always terminate in an orderly fashion, due to communication equipment or host failures. Even when sessions do terminate in an orderly fashion, a system crash at the host could result in the loss of the counter values and thus prevent or compromise subsequent logins. Thus, it would be especially convenient if a key lifetime were no longer than one interactive session, so that the problem of assigning the correct values to the message counters could be eliminated. If a different key were used for each login session, then the message counters could be set to zero at the beginning of each session.

Unfortunately, despite the advantages noted above, there are logistical difficulties associated with frequent key changes. A new key must be

---

(8) If the counter values are not restored properly at the beginning of each terminal session, but rather set to some fixed initial value or some value that may already have been used in previous message exchanges, then messages recorded from earlier sessions could be inserted into the connection by an intruder and would be erroneously authenticated by the protection modules.

distributed to the user via some secure channel, e.g., registered mail or bonded courier. One convenient medium that has been proposed for user key recording is magnetic stripped plastic cards [Smi]. Changing keys by issuing new cards or recalling and changing old cards entails substantial time and cost, making changing keys for each terminal session impractical. This points to the need for transmitting a new key over the terminal-host connection. The new key would have to be enciphered using some key already held by both ends of the connection. There are two basic approaches that may be used to transmit new keys: chained key changes and two-level key distribution systems.

With the chained key approach, a new key is enciphered under the last key that was issued and replaces that old key for all communication until another key change occurs. This forms a chain of key changes and, if an intruder discovers one key in the chain, he can easily decipher all messages subsequently transmitted as he can follow the chain of key changes. (9)

Using this chained key technique, if this new key were recorded in place of the old one on the magnetic stripped card, then a loss of this new key by the host in a crash would preclude further enciphered communication until a new key could be issued via some channel external to the system. The likelihood of key loss by the host is enhanced by the fact that the key held by it is changing frequently, so that backup media may not have the most

---

(9) Given the exhaustive key searching techniques from the appendix, it is also possible for an intruder to work backwards through the key changes, using the identity of the discovered key as known data enciphered under the previous key, to disclose the contents of all intercepted interactive sessions. This possibility is not a new vulnerability since during any key lifetime there will be enough information available to an intruder in the form of predictable message authenticators to break each key by exhaustive search anyway.

recent copy. Also, the recording of a new key on the user card at the terminal requires the introduction of equipment capable of reading and writing on the magnetic stripped cards, increasing the cost and complexity of the terminal modules and making them more prone to failure.

Using a two-level key distribution system, each new key is transmitted enciphered under a distinguished key used only for issuing new keys, thus preventing an intruder from working forward through the key changes. (10) Some protocol must be established to allow both ends of the connection to know when to use the distinguished key to decipher a new key. This protocol may be implicit, e.g., by issuing a new key only at the beginning of an interactive session, or it may require transmitting a message, enciphered under the key currently in use, indicating that the next message will be a new key enciphered under the distinguished key.

In order to avoid the difficulties associated with a simple, chained, key change protocol, a two-level key distribution system will be used at the beginning of each login session, and a chained key change approach will be

---

(10) Here, too, an intruder using exhaustive searching could work backwards through the protocol used to issue new keys, after discovering one key, and discover the distinguished key. If he could discover this distinguished key, an intruder could then easily decipher each key change and disclose the contents of all conversations, or impersonate the user in future interactions. The basic protection against this threat must come from a key space large enough to preclude exhaustive searching. When too small a key space is the problem, as is the case with the NBS cipher, some measure of extra protection for the distinguished key can be obtained by using a special protocol for initial key loading. Single blocks with no authentication information can be used to transmit a series of intermediate keys each enciphered under the previous key. This protocol increases the work required to discover the distinguished key linearly with the number of intermediate keys. Yet it is used only at the beginning of the session, so that the impact on channel utilization is minimal.

used during the session. (11) The distinguished key held by the user and the system on a long term basis will be designated as the primary key. It will be used only to issue a new secondary key at the beginning of each login session. The secondary key will be used for the encryption of regular message traffic. A secondary key also can be transmitted under the control of a previously transmitted secondary key, thus allowing use of multiple, chained secondary keys during a single interactive session.

The primary key for a user will be recorded on his magnetic stripped card and will be retained by the host in much the same way a password is retained by many systems. The protocol for changing from the primary key to a secondary key, and for later secondary-to-secondary chained key changes, requires the host to transmit the secondary key in a pair of enciphered messages, each containing half of the new key. (12) After the terminal receives a secondary key, it changes to the new key, resets the message counters, and sends a message to the host confirming receipt of the new key. The host has changed over to the new key and reset its counters after sending the new key messages, so it is ready to receive this confirmatory response.

The key change messages have the same general format as other messages, including an authenticator tag. In the case of a chained change from one secondary key to another, the tag need not be based on current counter values, but can be a static, known value, e.g., "0", as such key changes occur only

---

(11) An example of the use of both types of key change protocols in the same system is provided by the protocols used with the IBM 3612 consumer transaction facility [IBM2].

(12) If the key is approximately the same size as a message block, as is the case for Lucifer and the NBS cipher, then the key will not fit in one block because of the inclusion of an authenticator tag and message type information.

once during any secondary key lifetime. By employing the convention that the message in a key change protocol can be authenticated regardless of message counter values, secondary key changing can be utilized in error recovery procedures, when message counter synchrony is lost. This use of key change protocols will be explored further in chapter four. (13)

In the case of the primary to secondary key change associated with the start of a terminal session, extra authentication measures are required, as a single primary key is used to encipher the initial secondary keys for multiple sessions. The tag that authenticates these primary to secondary key changeover messages has the logical requirement to present a unique, predictable patter for each login attempted during the life of the primary key. Without such use dependent authentication, an intruder could masquerade to a user as the host by playing back the initial key change messages recorded during an earlier session. The login authentication protocol described in the next section meets this requirement without reintroducing the need for users to provide a different authenticator for each login. With this login protocol, key change messages still use fixed tags, and a regular data message bearing the date and time provides the unique, predictable pattern.

---

(13) When key changes are used in situations that are full-duplex, as with chained secondary keys, some form of synchronization must be employed to co-ordinate the key change on both channels so that no outstanding messages are deciphered under the wrong key. Co-ordination can be achieved by having the terminal respond with a distinguished message when it has received a message indicating that a key change is about to take place. Such a distinguished message, which should be authenticatable independently of message counter context and is issued only once under any key, provides a reference point for the key change by the host. Through the use of this kind of protocol, and by monitoring the values of the message counters in use at the host to detect impending counter wraparound, it is possible to automatically change secondary keys so that the secondary key lifetime can be adjusted to the size of the tag and the message traffic volume on the channel.

Login Protocol

Commonly used protocols for logging into a host are designed to effect a time-independent, one-way authentication. (14) Only the identity claim of the user is verified by the host by requesting a secret password (or other personal identification) known only to the user. Below is a two-way authentication scheme based on encryption techniques and the protocols proposed in this chapter. It is a variant of schemes discussed by Feistel [FH1] and by Saltzer and Schroeder [SS1]. The login protocol is presented from the view of a user accessing a host computer with no mention of an intermediate connection through a network access device. Use of this protocol in a network context is discussed in the next section. This protocol takes advantage of the key distribution protocol described above to reduce the amount of work performed by the user.

1. The user enables his terminal and establishes a connection to the host.

2. The host responds in cleartext confirming the connection by sending the host name.

3. The user transmits in cleartext his login identifier, then he inserts his magnetic stripped plastic card containing his (primary) key and enables the encryption module.

4. The host locates the user's primary key using the login identifier presented in cleartext. A new (secondary) key to be used during this session is created and transmitted using the standard key change protocol described in the previous section.

---

(14) Such an authentication procedure permits an intruder to masquerade as the host because it fails to require proof of identity from the host. Even if encryption is employed, the user could be confused or tricked by an intruder playing back recordings of previous logins because of the lack of time dependence in the login protocol.

5. The terminal deciphers the key change messages and loads this initial secondary key as the host also switches to this new key. The terminal then transmits a message confirming key receipt. The host, upon receipt of the confirmation is ready to engage in secure communication with the user. All communication from this point on will be carried out using the new key.

6. In order to demonstrate the time integrity of the connection to the user, the host now transmits the current date and time, in ciphertext, under the new key. The host has already been assured of the time integrity of the connection because of the correct receipt of the confirmation of key change message sent by the terminal under the new key.

7. The terminal module deciphers the date and time message under control of the new key and displays it on the terminal, permitting the user to judge the identity claim of the host and the time integrity of the connection.

This login protocol prevents an intruder from "spoofing" either the user or the host through the use of old recorded login sessions. Although a conventional password authentication procedure can be followed after completion of the protocol, it is not necessary if possession of the primary key is accepted as an identifying ticket. Note that the use of a different secondary key for each session carries an implicit form of verification of the time integrity of the connection from the host's viewpoint, thus there is no need for the user to respond with the time and date message as part of the login sequence.

Key Distribution in Networks

The terminal-host connection model presented in chapter two is a very general one, applicable to situations in which a host is accessed from dedicated or switched telephone lines or in general network environments. Below, we examine a scheme for authentication and key distribution designed for a specific network environment, and we see how the login and key

distribution protocols developed in this thesis can be used in such an environment.

Branstad has proposed a scheme for initiation of secure network communication [Bra]. In that scheme, user terminals and host sites on the network each hold keys that are used for identification and for secure distribution of working (secondary) keys. The Network Access Controller (NAC), a special host computer located in a network security center, acts as a verifier of user (and terminal) identity and as an intermediary in the distribution of the keys. The NAC holds the distinguished keys of all users (and terminals) and host sites, and generates and distributes the working keys used for user/host communication.

The key distribution protocol used by Branstad does away with the requirement that each host hold the primary keys of all possible users; rather the NAC acts as a repository for all permanent keys. This has an advantage in that the compromise of a single host does not result in the compromise of the primary keys of all users who ever use that host. Similarly, it avoids the need for a user to isolate his primary key from this danger by using a distinct primary key for each host with which he communicates. (15)

---

(15) Diffie and Hellman have suggested a modification of this scheme in which three controllers are used and each distributes a working key to the user and the intended host [DH2]. The controllers are addressed with different permanent keys by both the terminal and the host, and the working keys returned by the controllers are combined using an exclusive-or operation to form the final working key. The scheme has the advantage that the compromise of a single security controller does not result in disclosure of the final working key used by the terminal-host pair. It does entail the possession of two additional keys by the user, but this does not seem to be a major drawback as long as all three keys can be contained on a single magnetic stripped card. It also requires that all three controllers be operational or that a protocol be used to handle the case when one or more controllers are down.

Although the key change and login authentication protocols proposed in this chapter do not assume the existence of network access controllers, it is possible to use these protocols in conjunction with such controllers by allowing the controllers to pose as a host to the terminal and as a terminal to the host. Once the login authentication protocol has been carried out in this fashion between the terminal and the controller and between the controller and the host, the controller need only switch the connection so that the controller is no longer part of the connection between the terminal and the host. (16) Of course a different key would be used if one were to communicate with a host directly as opposed to going through the controller, for in the latter case the host uses its own key to establish the connection to the controller rather than employing the user's key. The important point here is that the protection protocols need not be different for these two different modes of host access, although the keys supplied to the protection modules may differ.

Summary

The authenticator tag design proposed in this chapter, consisting of a flag identifying the channel on which the message is to be transmitted and a counter of the number of messages transmitted on this channel, provides a simple means of detecting a wide range of message stream modification threats.

---

(16) By chaining subsequent secondary key issuances rather that using the primary key for a two-level key change, the key change protocol described in this chapter is usable in network environments as envisioned by Branstad. In such environments it is important that key changes occurring after the login can take place without intervention on the part of the network access controller.

The key change protocol described above permits the use of an authenticator tag that can be of moderate size as it need only be large enough to uniquely identify messages over the lifetime of one secondary key, an interval that is never longer than one terminal session, and to provide a specified probabilistic level of protection against erroneous authentication of spuriously generated messages.

The key distribution protocols described permit the use of a primary key for extended time periods without sacrificing security, by employing a key change protocol and by using a secondary key for the bulk of interactive session message traffic. This key change protocol is compatible with key distribution scheme proposed by Branstad and by Diffie and Hellman for network access controller environments. Over the lifetime of any one secondary key, any message that is recorded by an intruder and injected into a channel out of order can be positively detected. The removal of one or more messages from a channel by an intruder can be positively detected as soon as any succeeding message is received. Messages from previous terminal sessions provide no better basis for evading the authentication scheme than do messages synthesized by the intruder from randomly generated bits.

Finally, the login authentication protocol presented in this chapter provides a means of initializing the connection to a secure state with a minimum of user effort.

Chapter Four

Detection of Denial of Service and Resynchronization

In chapter three we adopted a tag design and protocols for authenticating messages in order to achieve the goal of detection of message stream modification. This chapter discusses protocols based on request-response messages and timeouts to detect denial of message service effected by connection blockage, and presents methods to resynchronize the message counters at both ends of the connection.

Detection of Denial of Message Service

As noted earlier, in our model of the terminal-host connection it is not possible to prevent an intruder from denying message service. Denial of message service can refer to a wide spectrum of intruder attacks, from complete disruption or blockage of the connection to the removal or modification of a single message. The authentication protocols presented in chapter three already provide a means of detecting denial of message service that occurs as a result of message stream modification. The receipt of an unauthenticatable message can indicate removal or modification by an intruder of intervening messages on a channel. If an intruder entirely blocks message flow on one or both channels, however, the protocols of chapter three provide no help in detecting the disruption. In this section we develop a request-response protocol that can be used to verify connection integrity to

the end that initiates the request. The protocol will also be used in resynchronization procedures discussed later in this chapter.

The request-response protocol involves the exchange of a pair of messages. The message issued to initiate this exchange will be designated as a request for status message. A message issued in response will be termed a status message. (A status message is also issued by the terminal to inform the host of successful completion of a key change as discussed in chapter three). Under normal operating conditions, both of these message types are authenticated in the same fashion as regular data messages. Associated with the transmission of a request for status message is a timeout. If a status message is not returned within the specified time interval, the requestor of the request considers denial of service to have occurred.

The use of a request-response protocol by the host and the terminal differs. In the case of the host, automatic generation of a request for status message at fixed intervals is required because the host has no means of predicting the arrival rate of messages from the terminal. The absence of messages from the user neither confirms or denies channel blockage. Thus, a timer in the host will initiate such requests at a rate dictated by user specifications. The timeout period for awaiting a response is adjusted according to communication channel delays. (1)

Compared to the host, the user is in a better position to detect a denial threat as evidenced by a lack of response to his commands. A user can check

---

(1) During the periodic connection integrity check, transmission is not suspended by the host after a request for status message is sent. This contrasts with the use of the request-response protocol for resynchronization as discussed in the next section, where transmission is suspended while awaiting the responding status message.

the status of the connection by manually issuing a request for status message, and being informed of the receipt of a confirming status message. By having the user initiate the request and judge when the response is overdue, we avoid the need to include a timer in the terminal protection module with the attendant increase in cost and complexity. Below we shall see that transmission of a request for status message by the terminal module will cause the message counters for the connection to become synchronized, thus this method of allowing a user to initiate a check of the integrity of the connection also provides the user with a means for manually resynchronizing the connection.

Resynchronization

Message tags and the request-response protocol provide the means to detect denial of message service. We now consider connection resynchronization following such a disruption. Since we have noted earlier that denial of message service cannot be prevented within the context of our model, it is reasonable to ask why any attempt at resynchronization should be made, as such action appears to be no more than an attempt at prevention. One justification is that if an intruder is disrupting the connection, then automatic resynchronization forces the intruder to continue his attack in order to continue the disruption, possibly making easier the task of locating the source of the disruption.

Another reason for attempting resynchronization is that connection disruption may be the result of a communication system failure not induced by an intruder. Although the encryption control modules are envisioned as not

assuming primary responsibility for recovery from transmission errors and similar low level communication system errors (see chapter six), it is still prudent to provide for resynchronization measures to be used in response to such errors. By providing mechanisms for resynchronization, the protection system becomes more robust in the face of some types of failures by lower level communication system components and permits the use of the protection system in environments that provide varying levels of error recovery. In particular, communication systems may implicitly assume that the user can manually resynchronize the connection if lower level mechanisms fail. The use of encryption and the authentication protocols described in this thesis precludes such manual resynchronization by the user, thus some automatic resynchronization protocol is required.

We will enhance the request-response messages described in the last section to allow their use for resynchronization as well. Both the request for status and status messages will now contain, as data, the reception counter at the end of the connection that transmits them, in addition to the transmission counter that is included in the authenticator tag. Figure 4-1 illustrates this message format (2) and labels the two channels and the message counters for use in the discussion that follows.

---

(2) The origin bit in the tag is omitted from the figure and the discussion that follows for clarity.

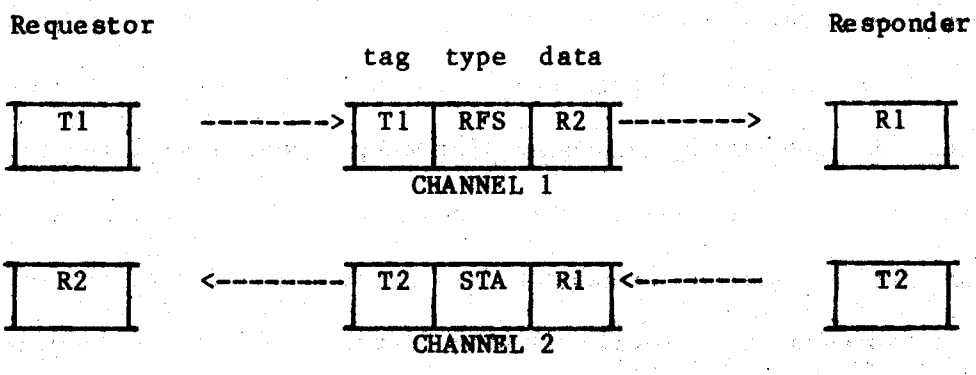


Figure 4-1

Model of Request-Response Resynchronization

We designate the sender of the request for status message as the requestor and the other end of the connection the responder. Referring to figure 4-1, the channel from the requestor to the responder is channel 1 and the other is channel 2. The requestor maintains the transmission counter for 1 (T1) and the reception counter for 2 (R2) while the responder maintains the transmission counter for 2 (T2) and the reception counter for 1 (R1). The actions of the requestor and responder described below are independent of both the identity of the requestor, either the host or the terminal, and of the circumstances that precipitated the invocation of the protocol, either a channel integrity check or a resynchronization attempt.

The requestor prepares the request for status message with the value of T1 as the authenticator tag and the value of R2 in the data part. T1 is incremented and the message is transmitted.

The responder, upon receipt of a request message the tag of which matches R1, increments R1 and sets T2 to the maximum of T2 and R2 (from the request

message). He prepares a responding status message with the value of T2 as the tag and with R1 in the data portion. T2 is incremented and the message is transmitted.

The requestor accepts as valid any status message the tag of which matches R2 or the data portion of which matches T1. (The reason for the alternate authentication possibility is described below.) Upon receipt of such a message, R2 is set to one greater than the maximum of R2 and T2 (from the status message). We will now examine how the request-response protocol, as amended, performs to correct various connection disruptions.

First we note that if no messages have been removed from either channel, the adjustment of T2 will not change its value and the adjustment of R2 will be the same as if any regular message had been received. Thus, if the protocol is invoked as part of a connection integrity check or in response to the receipt of an unauthenticatable message, and the counters are not actually unsynchronized, (3) the request-response exchange will occur with no ill effects.

Now we examine how the request-response protocol accomplishes resynchronization under circumstances when synchrony has been lost. We first consider the case of message stream modification on one channel, which is noticed by the requestor receiving an unauthenticatable message (on channel 2). In the unlikely case that T2 is lower than R2, which requires a previous erroneous authentication of one or more messages injected by an intruder or a module malfunction, then T2 should be incremented to match R2. This is

---

(3) Receipt of an unauthenticatable message resulting from injection of a message on a channel by an intruder does not affect counter synchrony.

accomplished by the request-response protocol since the the request for status transmitted by the requestor contains the value of R2. The responder will increment T2 to match R2 and send a response that will be authenticated based on the corrected value of T2. The discrepancy in counter values is logged by the responder after receiving the request message with R2 in it.

If an unauthenticatable message is received on channel 2 because one or more messages have been modified or removed from that channel, then R2 will be smaller than T2 and should be adjusted upward to agree with T2. T2 should not be decremented to agree with R2 as that would permit the retransmission of old messages by the intruder, until as many old messages were sent by him as had been removed. (4) The responder must inform the requestor of the value of T2, but he cannot send a message that will be authenticated by a tag that matches R2 without reusing a tag. This is where the alternate authentication procedure for status message is employed, allowing the requestor to accept the response and increment R2 to match T2.

For the alternate authentication procedure to work properly, it is necessary that the requestor suspend transmission pending receipt of the status message. Otherwise, T1 will not match the R1 value that was transmitted in the status message. (5) This is not an unreasonable restriction on the requestor as failure to receive a prompt response to the

---

(4) Such intruder retransmission could interfere with valid user-host communication as it may not be practical for the communication system, especially at the terminal, to retain old messages for retransmission and new messages that might be transmitted under already used message tags may be different from the removed messages.

(5) If additional bad messages are received by the requestor, they are logged but no more request for status messages are transmitted, so as not to interfere with the alternate authentication procedure.

request message is indicative of more serious problems. Upon receipt of the response, the number of messages removed or destroyed is logged by noting the difference between the tag and R2.

The resynchronization scenarios described above presume that synchrony has been lost on only one of the two channels and that no active denial of service via message blocking or message modification is occurring on either channel. If synchrony is lost on both channels before the resynchronization procedure is complete, or if messages are being blocked or modified on either channel, then the procedure will not succeed, leaving the requestor(s) of the request-response protocol waiting for an authenticatable status message. This situation will be detected by the automatic timeout for the status message in the case of a host initiated resynchronization. In the case of a user or terminal initiated resynchronization via the request-response protocol, the next automatic integrity check from the host will detect the failure to resynchronize.

Once the host becomes aware of the problem a second level of recovery strategy is employed. A new key will be issued by the host and message traffic will resume from that point. This is possible because the key change messages are authenticated independently of counter synchrony. Although this key change approach to re-establishing synchrony may seem a drastic one, it seems justified in light of the circumstances which are required to invoke it. (6) Because severe disruption of the connection results in this change of key,

---

(6) Unfortunately, resorting to a key change deprives the user of the information describing the extent of message loss as reported through the use of the request for status and status messages. The information could still be provided if the status message sent in response to completion of the key change, or some other special message sent immediately thereafter, carried

it reduces the desirability of such an attack for an intruder who is trying to subvert the protection measures. A timeout is also associated with the key change protocol, setting a limit on how long the host will wait for the confirmational status message, so that a failure to successfully issue a new key within an appropriate time interval will result in abandoning the connection. By associating a user specifiable limit with the number of times this form of resynchronization will be attempted during one login session, the user can maintain control over the use of resources in such recovery procedures and can cause the protection system to abandon the terminal-host connection.

Summary

We have described a hierarchic approach to dealing with resynchronization and have integrated this approach with denial of message service detection. This integration is achieved by using a request-response protocol as the basis for both resynchronization and detection of channel blockage. When the host or terminal attempts to establish synchrony after receipt of an unauthenticatable message, first an attempt is made to restore synchrony by initiating the request-response protocol on the other channel. If synchrony has not been lost or has been lost on only one channel, then this procedure will succeed, verifying the time integrity of the connection. If this procedure fails, or if a periodic connection integrity check fails, a key change is initiated by the host. Even if synchrony has been lost on both channels, the key change can succeed and establish a new reference point for

---

information about the values of the terminal reception and transmission counters before the key change occurred.

resumption of message exchange in a secure environment. Should the host not receive confirmation of the key change, within an appropriate time interval, the assumption is made that denial of message service is actively occurring, either as an intruder threat or as a result of a serious communication system failure, and the connection is abandoned.

The protocols presented in chapters two, three, and four will be described in greater detail in a sample implementation in chapter six.

Chapter Five

High Priority Messages

The discussion so far has ignored the need to support high priority messages sent by the user to the host to effect some urgent control function, e.g., to halt a runaway user process or to stop unwanted output arriving at the terminal. This chapter extends the connection model to include high priority messages and develops protocols for handling them.

Extending the Terminal-Host Connection Model

Most interactive computer systems embody the concept of a high priority message sent by a user at his terminal to his computation at the host. The specific messages used with different systems and subsystems vary. We presume that the texts of the various high priority messages are embedded in the user data sent on the terminal-to-host channel, and that some high priority message processing (HPMP) facility in the communication system at the host scans all user data received on a connection, recognizes the high priority messages, and acts on them. Because the host communication system may employ buffering between the HPMP facility and the rest of the connection, it is frequently necessary to provide some means of alerting the HPMP facility that a high priority message has arrived at the host, so that the HPMP facility can search the buffered input for the message. The protocols developed in this chapter are designed to provide an appropriate signal, regardless of the buffering

strategies employed in the host. In the next chapter, the host response to the signal, given various buffering strategies, is discussed.

The basis for the high priority message protocols is the addition of a special "attention" channel to the connection model, as illustrated in Figure 5-1. The attention channel is used only to signal the host end of a connection that a high priority message has been sent on the regular terminal-to-host channel. Care must be taken in the implementation of the host end of a connection not to buffer the attention channel, so the host protection module is never blocked from noticing a signal on the attention channel pending some asynchronous event. Note that this additional channel, like the other two, may actually be implemented in a variety of ways by low level communication system protocols, including the multiplexing of a half or full-duplex connection. (1) Because the attention channel is modeled as a separate channel, an intruder may have no difficulty in distinguishing messages transmitted on it from regular message traffic. Thus we cannot conceal the transmission of high priority messages and must be content to prevent the intruder from perpetrating undetectable acts of message stream modification or denial of message service on this third channel.

---

(1) In situations where a separate physical channel is not available to support transmission of high priority messages, some form of "out-of-band" signal may be used to simulate the transmission of a message on this channel. One commonly used protocol for transmitting a high priority message on a half-duplex connection involves sending a "line break" on the connection so that the terminal may gain control of the connection. The terminal can then send the text of the high priority message, having forced a line turnaround to occur.

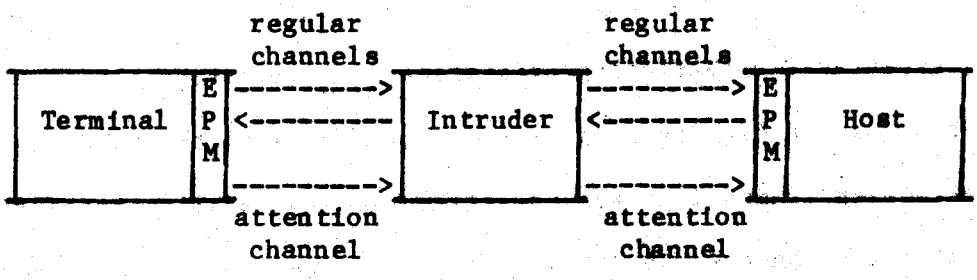


Figure 5-1

Connection Model Augmented to Include the Attention Channel

Protocols for High Priority Messages

The protocol presented below for the transmission of high priority messages permits wide latitude in the number and nature of messages sent and the buffering strategy used in the host. It is derived from the technique used in the ARPANET host-to-host protocol for transmission of high priority messages [ARP]. Two new control message types are introduced to support this protocol: attention and data mark messages. The attention message is the only message transmitted on the attention channel. The data mark message is transmitted on the terminal-to-host channel.

Three steps are involved in the transmission of a high priority message. First, the text of the high priority message is sent on the regular terminal-to-host channel. Next, an attention message is constructed and transmitted by the terminal protection module on the attention channel. (2)

---

(2) In environments where an existing communication system protocol is used to support transmission of high priority messages, the attention message is transmitted in conjunction with this existing protocol and serves to securely authenticate the existing protocol's claim of receipt of a valid attention message.

Finally, a data mark message is constructed and sent on the regular terminal to host channel. (3)

The host protection module must be farther out on the connection than the HPMP facility, as high priority messages must be deciphered before the HPMP facility can process them. Thus, the attention message serves to notify the protection module that a high priority message is enroute, while the data mark message locates the end of the text of the high priority message in the regular channel and marks the position in this channel that corresponds to the transmission of the attention message on the attention channel. (4) Upon receipt of an attention message and the matching data mark, the host protection module signals the HPMP facility of the arrival of the high priority message. Discussion of the details of the signalling, and other interaction with the host communication system in conjunction with the processing of high priority messages, is deferred to chapter six, as these details are dependent on the buffering strategy employed in the host.

Since the attention channel is distinct from the other two channels, it has a distinct pair of message counters associated with it. The transmission counter for this channel is located at the terminal end of the connection and the reception counter is at the host end. An attention message tag consists

---

(3) In systems that use only one type of high priority message, e.g., a "quit" on Multics, no text related to the high priority message need precede the data mark message. Receipt of the data mark message is sufficient to transmit the desired control signal and mark the position in the regular terminal-to-host channel that corresponds to the transmission of the attention message.

(4) As the data mark message is a protection module control message, it does not appear in the cleartext output from the protection module, and it may need to be translated into a data mark character to delimit the high priority message text for processing by the HPMP facility.

of the usual terminal origin identification and a transmission counter value that indicates the number of attention messages that have been transmitted since the initialization of the attention channel. Because attention messages are sequenced on a separate counter, they can be received and authenticated independent of messages transmitted on the regular channel. (5)

Each data mark message carries an authenticator tag of the same form as other messages on the regular terminal-to-host channel. Included in the data portion of a data mark message is the value of the attention message transmission counter at the time the data mark message was transmitted. This serves to associate data mark and attention messages. Hence, a given data mark message can be correctly paired with a matching attention message, despite interference on any channel. This design of the data mark and attention messages also links together, for detection of denial of message service, the attention and regular channels.

Figure 5-2 illustrates the use of the protocol described above in the transmission of a high priority message. High priority message text in a user data (DATA) message, an attention (ATT) message, and a data mark (DMK) message are shown enroute to the host. The message formats displayed are the same as in chapter four: tag, type, data. Values for the regular terminal-to-host transmission (Tc) and reception (Rc) counters and the attention message counters (Ac) also are shown.

---

(5) We shall see in chapter six that this is a necessary property for the attention message because of problems associated with recognition of enciphered attention messages by facilities further out than the protection module.

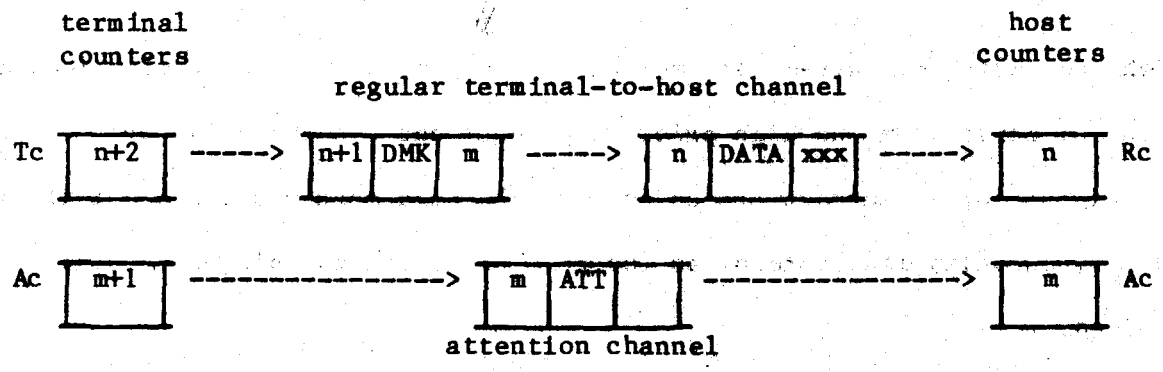


Figure 5-2

High Priority Message Transmission Scenario

Although resynchronization and integrity checking could be carried out for the attention channel separately, these functions can be performed simultaneously for all three channels without introducing any new message types. This is accomplished by including the appropriate attention channel message counter value in request for status and status messages (6) and expanding the counter update procedures to include this additional channel.

This extension of the resynchronization protocol is not complicated since this new channel does not enter into the alternate authentication scheme for status messages. Receipt of a data mark or attention message that does not have an acceptable authenticator tag, or receipt of a message on the wrong channel, results in initiation of the resynchronization protocol just as does receipt of any other "bad" message. A new context for initiating the resynchronization protocol now exists: receipt of a data mark message for

---

(6) The attention channel transmission counter is included in the data portion of a request for status or status message transmitted by the terminal while the reception counter for the high priority message channel is included in such messages when transmitted by the host.

which no matching attention message has arrived. This situation indicates denial of service on the high priority channel and is handled by accepting the high priority message preceding with the data mark message and initiating the resynchronization protocol as though a unauthenticatable message had been received.

Summary

This chapter extended the connection model of chapter two to include high priority messages and the facilities necessary to process them. A new channel from the terminal to the host was added, and two new message types, attention and data mark messages, were introduced to support transmission of high priority messages. The data portion of request for status and status messages was extended to contain the values of the message counters for this new channel. The resynchronization and detection of denial of message service protocols were modified to include the new channel.

Chapter Six

Communication System Interfaces

In this chapter we refine our communication path model, examining it not simply as a terminal-to-host connection, but rather as a connection between a user and his computation. Our point of view in examining this connection is based on the research of computer input/output systems by Clark [Cla]. With this view in mind, we answer the question of where to position the protection modules with respect to the various hardware and software modules at both the user and computation ends of this connection. The strategy we adopt is to position the modules to encompass all multiplexed system facilities, as well as all physically unsecured facilities. This simplifies the task of verifying the security claims of a system by restricting the appearance of cleartext to environments that are private to a single user. Also discussed are the impact of different input buffering strategies on host protection module structure, methods for promptly recognizing high priority messages, and methods for echoing characters efficiently.

Effect of Security and Functionality on Positioning

Two major factors influence the positioning of the protection modules in the connection between the user and his remote computation: security and functionality.

With respect to security, the encryption modules provide protection from certain forms of intruder threats directed against that portion of the logical

connection that is "between" them. Certainly all of the physically unsecured portion of the connection need be between the modules, but it also is useful to encompass certain physically secure parts of the communication system. The design and verification of the correct operation of the portion of the communication system that is between the protection modules is simplified because that portion cannot compromise the connection any more than the intruder of our model.

Of special interest are the parts of the communication system, whether physically unsecured or not, that are multiplexed among many users. A fundamental principle in the design of secure systems is the avoidance of unnecessary common mechanism [SS1], for mechanisms that are common to more than one user provide a potential path for unwanted user interaction. Because the protection modules are associated with individual logical connections, they need not be implemented in a multiplexed facility of the communication system. Indeed, the encipherment provided by the protection modules can assure the logical separation of individual connections as they pass through various multiplexed facilities. Examples of communication system hardware facilities that frequently are multiplexed among many connections, and thus should be positioned between the protection modules, are terminal concentrators and host front end processors (FEP's). Examples of software facilities that frequently are multiplexed are buffer management modules for multiplexed channels. Thus, we will position the protection modules so as to encompass all multiplexed facilities in the communication system, allowing the protection modules for a single connection to operate in an environment that is private to that connection. This positioning strategy is illustrated in

Figure 6-1, which shows the path through various communication system modules that might be followed by a typical connection.

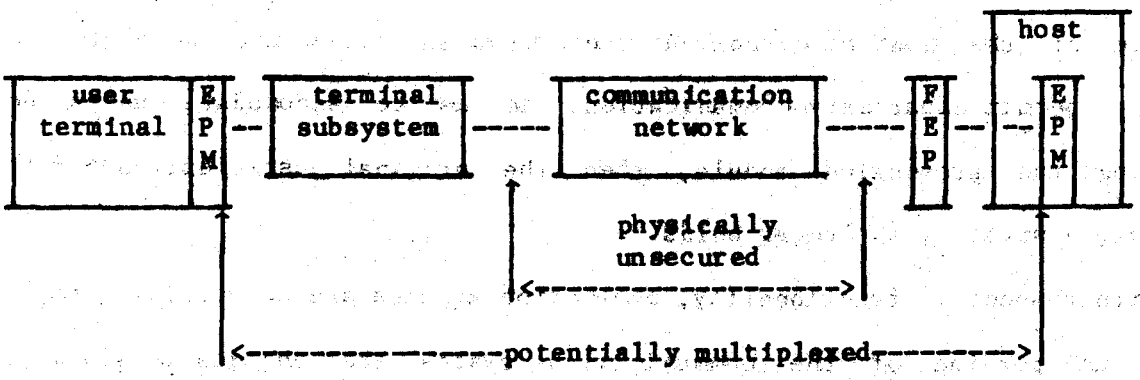


Figure 6-1

Protection Module Positioning Strategy

A different view of security can lead to an alternative positioning strategy. If the major security concern is preventing messages from ever appearing in a physically unsecured environment in cleartext, and it is considered less important to prevent leakage among logical connections, then it can be argued that the modules should be positioned at the boundaries between the physically secure and unsecure portions of the communication path. Then input/output can be forced to pass through the encryption algorithm, thus assuring that any data that enters the unsecure environment is protected from unauthorized disclosure. This alternative positioning strategy will almost always result in multiple individual cleartext connections being handled in a multiplexed facility somewhere. We believe that improved software verification techniques and careful system design will make less desirable this particular hedge against failures by host or terminal systems to prevent messages from appearing in a physically unsecured environment in cleartext.

Moreover, as we are interested in providing secure communication for hosts that have diverse user communities, this strategy seems unattractive as not all users may have terminals equipped with protection modules. If provision is made at the host to circumvent the encryption scheme and the protection module to permit cleartext communication, so as to accomodate users not utilizing the protection module, then the original justification for the alternative strategy no longer holds.

With respect to functionality, protection modules are constrained to be below the portion of the communication system that engages in syntactic processing of message contents. These constraints of the communication system functionality are primarily a factor in positioning of the protection module at the host, as almost all processing of this nature is performed at the host. With respect to output from the host, encryption can be performed only after such transformations as device-specific code conversion, white-space optimization, and formatting. With respect to input to the host, messages must be deciphered before such transformations as canonicalization, break character detection, erase-kill processing, translation, escape sequence processing, character echoing, and high priority message recognition can be performed. (1)

---

(1) Character echoing and high priority message recognition will be discussed in detail later in this chapter. Canonicalization refers to the arrangement of input data into a form that removes the ambiguities introduced by the use of carriage motion control characters [SO]. Break characters delimit the effects of erase-kill processing and canonicalization and cause the input to be forwarded to higher levels for possible further processing. Escape sequence processing refers to the transformation of multi-character sequences used to enter characters that have control meanings without invoking the associated control functions, into their single character representation. Formatting of output involves conversion of tabs to spaces for terminals that do not support hardware tabs and insertion of newlines in output when strings are longer than

At the terminal end of the connection, the security requirements and functionality constraints dictate positioning the protection module between the terminal and the rest of the communication system. Such components as terminal concentrators, line adaptors, and modems will be "further out" on the connection than the protection module. This strategy provides substantial flexibility in configuring terminal subnetworks in which not all the terminals may be using the protection modules. At this end of the connection, it seems reasonable to implement the protection module in hardware, as this end of the protection system has been designed to require a minimum of processing power. With the current capabilities of large scale integration, it seems plausible that the protection module hardware could be fabricated using a microprocessor and a special chip for the encryption algorithm.

At the host end of the connection, the security requirements and functionality constraints will usually require implementing the protection modules in software. (2) By implementing this protection module in software, the memory protection machinery in the host computer can be used to provide a private environment for the execution of the protection module for each connection, and the protection modules will be beyond any multiplexed buffers managed by the host operating system software. (3)

---

the line length of the target terminal. White space optimization refers to replacement of multiple spaces with tabs and of multiple line feeds with form feeds.

(2) The addition of a hardware encryption/decryption instruction to the host instruction repertoire may be required to obtain efficient operation.

(3) The host's memory protection machinery also may be used to protect the modules from user level programs that may damage or circumvent them. The user level programs might inflict damage as a result of errors or might be "Trojan horse" programs [SS1] supplied by an intruder to subvert the modules and permit the intruder to assume control of the user's computation by disabling

Implementation of the protection modules at the host as software modules private to each user computation also has two advantages with respect to the design and verification of the modules themselves. First, at this level in the software of the host, modules can usually be implemented in an environment that is conducive to the design of a well-structured protection module, permitting the use of high level, structured programming languages and multiple-process (rather than interrupt) organization of the control structure. (4) This means that the modules can be simple in design and, consequently, their correctness may be easier to verify because they need not deal with irrelevant communication system details. Second, it may be possible to isolate many of the characteristics of the physical connection from the protection module, presenting it with a simple virtual connection interface. The communication system configuration characteristics need not be programmed into the modules. For example, although the protocols are designed to operate in a full-duplex environment, they can be utilized on either half or full-duplex physical connections if the interface presented to the modules reflects a virtual full-duplex connection.

---

or subverting the protection module. Whether or not the host protection module is part of the security kernel [Sch] of the host system depends upon the security policy to be enforced. It will be part of the kernel if the security policy requires certain users to employ a system supplied protection module; otherwise not.

(4) Such facilities might not be available if the host protection module were implemented in a front end processor or in a restricted environment in the lowest levels of the operating system.

Buffering Strategies

Any communication system for connecting users with interactive computations must deal with the fundamental problem of synchronizing the arrival of messages from a user with the demands for input from his computation. Many systems achieve the necessary synchronization by providing one or more buffers in the connection between the user and his computation, thus allowing the user to work ahead of its demands for input. (5) The positioning of these buffers has impact on the structure of the host protection module, which impact we will now explore.

Figure 6-2 illustrates possible buffer positions. In this figure, the box labelled EPM is the host encryption protection module for the connection, and that labelled CMM is a connection management module that performs the various required syntactic transformations on the input following decryption, including recognition and processing of high priority messages. For different communication system organizations, buffers may appear at positions A, B, and C in any combination. A buffer at any of these positions can provide the required synchronization of arriving input and demands for input from the computation.

---

(5) This synchronization problem also can be handled by explicitly prohibiting the user from entering data at his terminal until his computation is ready for that data. A communication system can enforce such synchronization by transmitting a control character to the terminal to "lock" the keyboard when the computation enters a state where it is not accepting input, and then transmitting another control character to "unlock" the keyboard when the computation is ready to accept input. If this approach to achieving synchronization is employed, the following discussion about buffering and its impact on the protocols is irrelevant.

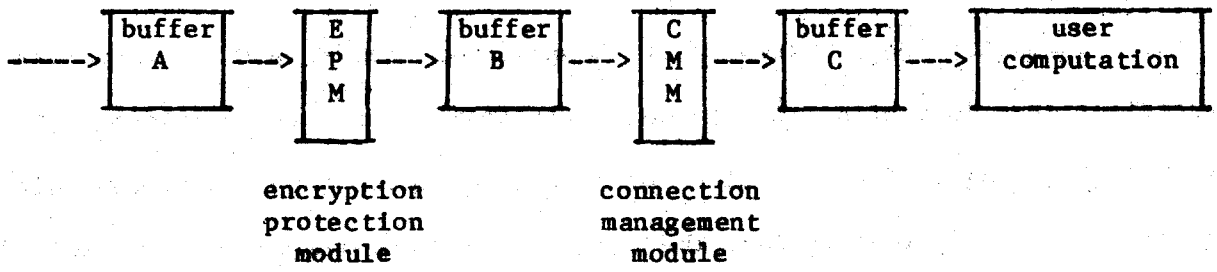


Figure 6-2

Buffer Position Possibilities for Host Input Channel

Buffer A represents the buffering of input to the host in front of the protection module, perhaps by a front end processor or by operating system facilities. Because this buffer is between the protection modules, it may be part of a common buffer management mechanism that supplies messages upon demand to all protection modules in the host. This buffer is not necessary and its presence only complicates the operation of the protection module, as we discuss below. Buffer B is also not necessary if the connection management module is implemented so that it immediately accepts the cleartext output from the protection module. As will be seen in the next section, buffer B complicates the processing of high priority messages. Buffer C holds input processed by the connection management module but not yet requested by the user's computation. Location C is the preferred position for the buffer that synchronizes data arrivals with computation demands for input.

Response to Timeouts

Buffer A interferes with the processing of timeouts used to detect the failure of a status message to arrive within a predetermined interval. When

buffer A is employed, the protection module first must request and examine all messages in buffer A before deciding that the occurrence of the timeout really represents a failure to receive a status message. Thus, with respect to processing of status timeouts, it is preferable for the protection module always to receive input from the connection upon its arrival at the host, without the existence of buffer A. Such an arrangement is possible because the cleartext output from the protection module can be forwarded to buffer B (or to buffer C if buffer B is not employed).

High Priority Message Processing

In order for a high priority message to have its desired impact, the host must recognize and process it quickly upon receipt. Quick processing is no problem if buffers A and B are not present, for the connection management module will notice high priority messages as they arrive, independently of the rate at which the computation demands input. (6) In this case the high priority message protocols of chapter five are not needed. The host protection module can still match data marks to attention messages and keep track of the various counters, but it need not signal the connection management module when an attention/data mark pair arrives. (7)

---

(6) The standard communication system flow control protocols prevent overflow in buffer C, as their action is not inhibited by the presence of the protection modules.

(7) If input synchronization is accomplished through the use of keyboard locking, a high priority message is usually sent by transmitting an out-of-band signal to the host. The host then responds by sending the control character that causes the keyboard to be unlocked, allowing subsequent transmission of the high priority message text from the terminal. In this case, although the data mark message is not necessary, the attention message

If buffer B is present, the connection management module may not notice high priority messages as they arrive. In this case, the protection module must signal the connection management module when a high priority message has arrived. The protection module, upon receipt of a data mark, does three things: increments a counter of data mark messages received, places a data mark character in buffer B, and signals the connection management module. The data mark character is placed in the buffer so that the connection management module knows when to stop processing input from buffer B. The counter of the number of data mark messages received is used by the connection management module, in conjunction with a counter of the number of data mark characters it has examined, in order to synchronize data mark characters and signals from the protection module. (8)

Finally, if buffer A is present, some facility must be provided to recognize attention messages and forward them to the protection module, bypassing buffer A, and the protection module must request and examine the contents of buffer A to locate the data mark message. Figure 6-3 illustrates this configuration, depicting the protection module, buffer A, and the attention message recognition (AMR) facility of the communication system.

---

can be used to authenticate the out-of-band signal used by the standard communication protocol.

(8) This is an example of the "wakeup waiting" problem as described by Saltzer [Sall].

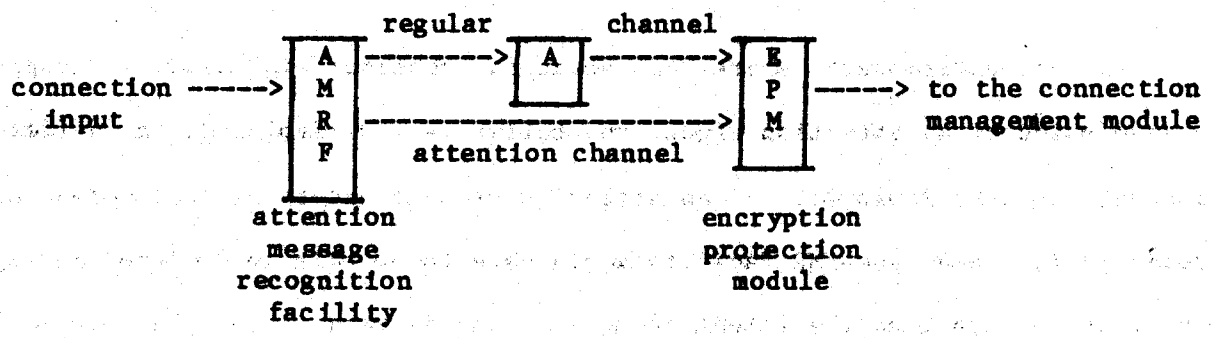


Figure 6-3

Attention Message Recognition

If the communication system employs a special protocol for signalling on the attention channel under regular (unencrypted) circumstances, then this same protocol can be used in conjunction with the transmission of the attention message to notify the protection module that a high priority message is enroute. Under such circumstances, the AMR facility takes the attention message that was sent using this standard protocol and forwards it to the protection module for processing. This attention message is given to the protection module in front of regular input that may be in buffer A, since the attention message logically belongs on the attention channel. The protection module can decipher and authenticate the attention message and request the contents of buffer A. These contents are processed by the module to locate the data mark message. If the data mark message is not located in the buffer contents, an integrity check is initiated, resulting in flushing the connection to the host protection module and locating of the data mark or timing out and changing keys.

In an environment where no standard protocol is used to support transmission of an attention signal and buffer A is employed, a different approach must be employed. If an attention message had to be deciphered to be recognized, then the AMR facility would have to be able to decipher messages in order to recognize the attention message and forward it to the protection module. As buffer A and the AMR facility may be part of the common mechanism of the communication facility, this is not acceptable and below we show how to ameliorate this situation.

In chapter five we saw that attention messages are constructed using only the value of the attention message transmission counter, the terminal origin identifier, and the type identifier for attention messages. Thus, the host can construct the enciphered image of the next attention message that will be transmitted by the terminal, under the current secondary key, and pass this bit pattern to the AMR facility as the basis for recognition of an enciphered attention message. (9) Upon arrival of an attention message that matches the template, the AMR facility forwards it to the protection module ahead of any messages in buffer A. The protection module processing from this point is same as if a standard communication system protocol had been used in conjunction with the transmission of the attention message. (10)

---

(9) A new attention message template must be distributed at the beginning of each session, after every key change, and whenever the value of the host attention message counter changes. The host protection module can distribute several templates to the AMR facility at one time, corresponding to the series of attention messages to be transmitted by the terminal module. This eliminates the likelihood of an attention message arriving and not being recognized by the AMR facility because the facility has not yet received the next template from the host protection module.

(10) Note that even if the enciphered attention message template has been compromised by the communication system and the attention message received by

Should an attention message be removed from the attention channel, the next attention message transmitted by the terminal will not match the template held by the AMR facility and will not be recognized as an attention message. A similar situation arises if an attention message is modified enroute to the host. In either case, examination of the "bad" attention message by the host protection module, in the course of normal message processing, results in a channel resynchronization, and the user is notified of the loss of the attention message. The maximum delay that can occur in recognition of an attention message under these circumstances is dictated by the timeout used for periodic connection integrity checking (see chapters four and six). (11)

By using the mechanism proposed above to solve the problem associated with attention message recognition, we are able to use the host protection module whether or not buffer A is present and whether or not a standard communication system protocol is used in conjunction with the transmission of attention messages.

Echoing

The term "echoing" is applied to a variety of character processing techniques performed on asynchronous communication lines usually operating in full-duplex mode. In its simplest form, echoing may merely involve the

---

the protection module is fraudulent, the module will not be tricked into disrupting the input to the user's computation (as long as no input is discarded by the protection or the connection management modules) because there is no matching data mark message to confirm transmission of the attention message. The connection integrity check, initiated by the host when it fails to locate the data mark message, will detect this injection of the attention message and resynchronize the connection.

(11) If this timeout is set to a short enough interval, then it may not be necessary to propagate an attention message template as noted above.

transmission back to the terminal of every character sent to the host. This type of echoing is sometimes designated as echoplex mode and is used primarily as a means of verifying the reception of characters transmitted over voice grade lines. More elaborate echoing may involve a substitution for some characters on a one-for-one basis or even a variable length substitution-for-characters received from the user terminal. (12) Additionally, echoing may be co-ordinated with host output messages so that asynchronous interactions do not result in haphazard mixing of user input and host output on the user terminal display. The echoing connection seems to belong in the connection management module of the communication system hierarchy, for it must analyze cleartext. Such placement of the echoing function, however, can cause inefficient use of connection bandwidth and potentially unacceptable real time delays for the user.

First, we note that the use of the protection protocols eliminates a fundamental reason for employing echoplex mode echoing. This is because use of the protection modules guarantees, with high probability, that the characters received by the user's computation have not been altered in transit. (13) Thus, as long as some means is provided for displaying each typed character on the terminal, so that the user can determine if he has

---

(12) This last characterization of echoing includes techniques that analyze terminal input in an effort to complete the composition of an input line, or a portion thereof, on behalf of the user. Such processing is very sensitive to the subsystem with which the user is interacting and thus is usually performed within the user's process at the host [Bob].

(13) Because the host is not actively echoing each character typed by the user, this configuration does not provide the rapid detection of severance of the connection that host-based echoing provides. This may be a problem in situations where the user is typing text for which he expects no response from his computation, e.g., entering text into a file for later editing.

typed what he thought he typed, there is no need to involve the host in echoplex mode echoing.

If host-based echoing is used with the protocols developed in this thesis, because the echoing is more sophisticated than echoplex mode echoing, each character input by the user would be enciphered in a separate message block and transmitted to the host, where the block would be deciphered and any required echo processing would be performed. The result of that processing would be enciphered in a message block and transmitted to the terminal where it would be deciphered and displayed. Thus, each character transmitted by the terminal would go through the encryption/decryption algorithm a total of four times under these circumstances. (14) This encryption overhead, when added to the round trip transmission time and host processing delays usually associated with echoing, may constitute an unacceptable real time delay for a user at his terminal. Of course it should be remembered that the user generally transmits data to the host at a much lower rate that he receives it and the effective bandwidth provided by this approach to echoing may be acceptable if the protection modules are fast enough.

In many hosts echoing is performed by some multiplexed facility, e.g., a front end processor. For the security reasons noted earlier, it is not desirable to permit a multiplexed facility to contain the host protection module in order to perform echoing. Because the echoing performed by a multiplexed facility is usually relatively simple, as opposed to sophisticated echoing that requires a private, host-based process, the solution presented

---

(14) This transmission of blocks containing single characters results in block space utilization of about 5% and 10% for Lucifer and NBS block sizes respectively.

below alleviates the problem of multiplexed facility echoing, as well as reducing transmission and encryption overhead.

As an alternative to host-based echoing in situations not requiring extremely sophisticated echo processing, we propose the addition of an echoing module to the protection module located at the terminal end of the connection. The degree of sophistication provided by such a module can vary over a wide range depending upon the desires of the user community. Details of local echoing procedures have been developed as the Remote Controlled Transmission and Echoing (RCTE) Option in the ARPANET TELNET protocol [ARP] for use in situations where the time delay associated with conventional remote echoing is considered unacceptably long, e.g., in satellite connections from continental users to the Aloha system in Hawaii, or when the host does not wish to be burdened with the extra processing. The Telnet system also provides a host level protocol option for such local echoing [TCC]. The concept of using a microprocessor to implement such a local echoing module has already been suggested in connection with packet radio networks [KaR]. This approach to echoing eliminates the real time delay and inefficient block space utilization problems noted above and does not require the participation of any multiplexed facility in the echoing.

If a private process or task is provided to monitor terminal input and the connection management module is contained in this process, then sophisticated forms of echoing can still be provided by directing the terminal echoing module to transmit (for echoing) only those characters that require special processing. This minimizes the impact of echo processing on the connection performance since most characters are locally echoed and only a

few require echo processing by the host. Since sophisticated echo processing usually entails the use of private tasks or processes dedicated to monitoring terminal input, this scheme does not imply a drastic extension of the functionality already provided in such environments.

Summary

In this chapter we have examined factors influencing the positioning of the encryption modules in the communication system. By positioning the modules above the level of multiplexed facilities in the communication system, the security guarantees provided by the modules cover much of the communication system. This results in reduced complexity in verifying the secure operation of both the protection modules and encompassed portions of the communication system, and increased flexibility in configuring diverse user terminal networks. Problems associated with recognition of attention messages in various host communication environments were examined and techniques of supporting high priority message transmission in all of these environments were presented. Problems associated with a broad spectrum of echoing techniques were examined and it was proposed that, in the case of simple echoing on asynchronous lines, some variant of a remote controlled transmission and echoing protocol be employed to reduce real time delays and to improve bandwidth utilization.

Chapter Seven

Control Structure of the Protection Modules

This chapter consolidates the discussion of the earlier chapters by presenting a description of both the terminal and host protection modules. This detailed description brings out aspects of the interaction of the protection protocols that is not evident from the independent descriptions of the protocols in earlier chapters.

Message Formats

Seven types of messages were introduced or implied in the discussion of protocols in earlier chapters. Formats for these message types are presented in Figure 7-1. No specific message block size is presumed in this description, thus such details as the width of the various fields and unused space will be ignored. (1) These message formats can be used with either the 128-bit Lucifer blocks or the 64-bit NBS blocks.

As indicated in chapter three, all messages have the same general format, consisting of origin identification, transmission counter, message type, and data fields. The host is identified by a "1" in the origin field and the terminal is identified by a "0". The data field contains information specific to a given message type and the message type field classifies the message as a

---

(1) In particular, relative field widths do not imply actual size relationships among fields.

data (DATA), status (STA), request for status (RFS), key change (KC1 & KC2), attention (ATT), or data mark (DMK) message.

name	origin	trans. counter	type	data field	
DATA	0/1	Tc	CC	characters	
STA	0/1	Tc	STA	Rc	Ac
RFS	0/1	Tc	RFS	Rc	Ac
DMK	0	Tc	DMK	—	Ac
ATT	0	Ac	ATT	—	
KC1	1	00...0	KC1	1st half of new key	
KC2	1	00...0	KC2	2nd half of new key	

Figure 7-1

Message Formats

Data messages are used to transmit the character strings that represent explicit user-computation correspondence, including the text of high priority messages. The transmission counter of the sender forms DATA.Tc. In the type

field. To prevent confusion, type field values for the other six message types are numbers bigger than the character capacity of a data message. Several data messages may be needed to transmit a user level logical unit of correspondence. Because the number of characters contained in the data field is indicated in DATA.CC, no special conventions are required for indicating the end of the used portion of the data field.

The authentication tag of a status message contains the same information as in a data message, while the type field identifies the message as a status message. STA.Rc in the data field contains the value of the regular message reception counter of the sender and STA.Ac contains the value of the attention message counter from the sender's end of the connection.

The content of a request for status message differs from that of a status message only in the type field.

In a data mark message, the standard transmission counter (DMK.Tc) field is used but the origin is always "0", indicating the terminal as sender. The data field contains the value of the terminal's attention message transmission counter in DMK.Ac.

In an attention message, the origin is always "0", the transmission counter (ATT.Ac) field contains the value of the terminal's attention message transmission counter and the data field is not used.

Two types are used for key-changes. The origin field is always "1", indicating the host as sender, and the transmission counter field contains some constant value agreed upon by both ends of the connection, e.g., "0". The data field contains half of the new key (KCx.Key), the first half arriving in the first key-change message and the second half in the second.

Control Structure of the Modules

Although there are many ways the modules can be viewed and implemented, we have chosen to describe each module as a single process, using message style interprocess communication facilities for the interfaces to the terminal, the user process in the host, and the communication system. An actual implementation may use multiple processes and/or processors for each module. We have not described a multi-process(or) implementation of the modules so that we may omit the details of avoiding contention over the counters Tc, Rc, and Ac that could result from asynchronous processing of messages on the three channels of a connection.

Each protection module can be viewed as consisting of three operating states: the normal state, the bad-message state, and the key-change state. (2) The normal and bad-message state are very similar in both modules, while the key-change state is module specific.

Two functions are used frequently by both modules: message packaging and error logging. Message packaging consists of incrementing the message transmission counter, combining this counter value and the origin identification bit to form the tag, appending the message type field and data field of the message, then enciphering the completed message block. A packaged message is ready for transmission on an outbound channel. The data field and the type field of the message are supplied to the part of the module that packages the message. In the case of the terminal module, there is also an indication of whether the attention or regular message counter is to be

---

(2) The terminal module also contains a transient starting state, the key-wait state.

used and, implicitly, whether to use the regular or attention channel for transmission of the message.

Error logging is an implementation dependent function. At the host, logging can be accomplished by recording error messages in a file associated with each connection. At the terminal, logging may be accomplished by generating messages on the terminal display or through lights, audible alarms, etc.

The structure of the two protection modules is quite similar. We shall describe the terminal module first and then describe the host module by noting how it differs from the terminal module.

In the normal state, the terminal module is blocked waiting for both cleartext and ciphertext input. In the bad-message state, entered after the receipt of an unauthenticatable message and subsequent transmission of an RFS, and in the key-change state the module is waiting for ciphertext input only. (3)

We first describe the processing of ciphertext input by the terminal module, examining the transitions between the states and the processing that occurs upon receipt of various message types. Figure 7-2 illustrates the control structure of the terminal module in terms of the three states listed above and should be examined while reading the following discussion.

After transmitting his login identifier in cleartext, the user inserts his primary key and enables the protection module. The terminal module is initialized by loading the primary key as the current key and setting all

---

(3) In these two states, keyboard input is not processed. This may be accomplished by providing a buffer for input typed while the module is in one of these two states, or by "locking" the keyboard.

three of its counters to zero. (4) The module then enters the key-wait state, waiting for the arrival of a ciphertext block containing a valid KC1 message. All other input is discarded until such a message arrives. Upon receipt of a KC1 message, KC1.Key is saved and the module enters the key-change state.

Upon entering the key-change state, the module waits for a message from the connection. The next message received on the connection must be a valid KC2 message or the protection module abandons the connection, logging the error. If the next message to arrive is a valid KC2 message, the saved value of KC1.Key is combined with KC2.Key to form the new current key and Ac, Tc, and Rc are all set to zero. The module packages and transmits an STA message, logs the key-change, and returns to the normal state.

Upon receipt of a message on the user-computation connection in the normal state, the origin bit is checked and, if it does not indicate the host as sender, the message is considered unauthenticatable. The transmission counter field and the message type field are checked and, in the case of a DATA or RFS message, the transmission counter field must match the value of Rc to be accepted. An STA message is accepted if STA.Tc matches Rc or if STA.Rc matches Tc. A KC1 message is accepted if KC1.Tc contains the appropriate constant value, e.g., "0". All other messages are classified as unauthenticatable. Now we explore the processing of each message type.

---

(4) To facilitate the description of the protection modules, the regular message transmission counter for each channel is designated Tc and the regular message reception counter is Rc. The attention message counter at each end of the channel is referred to as Ac.

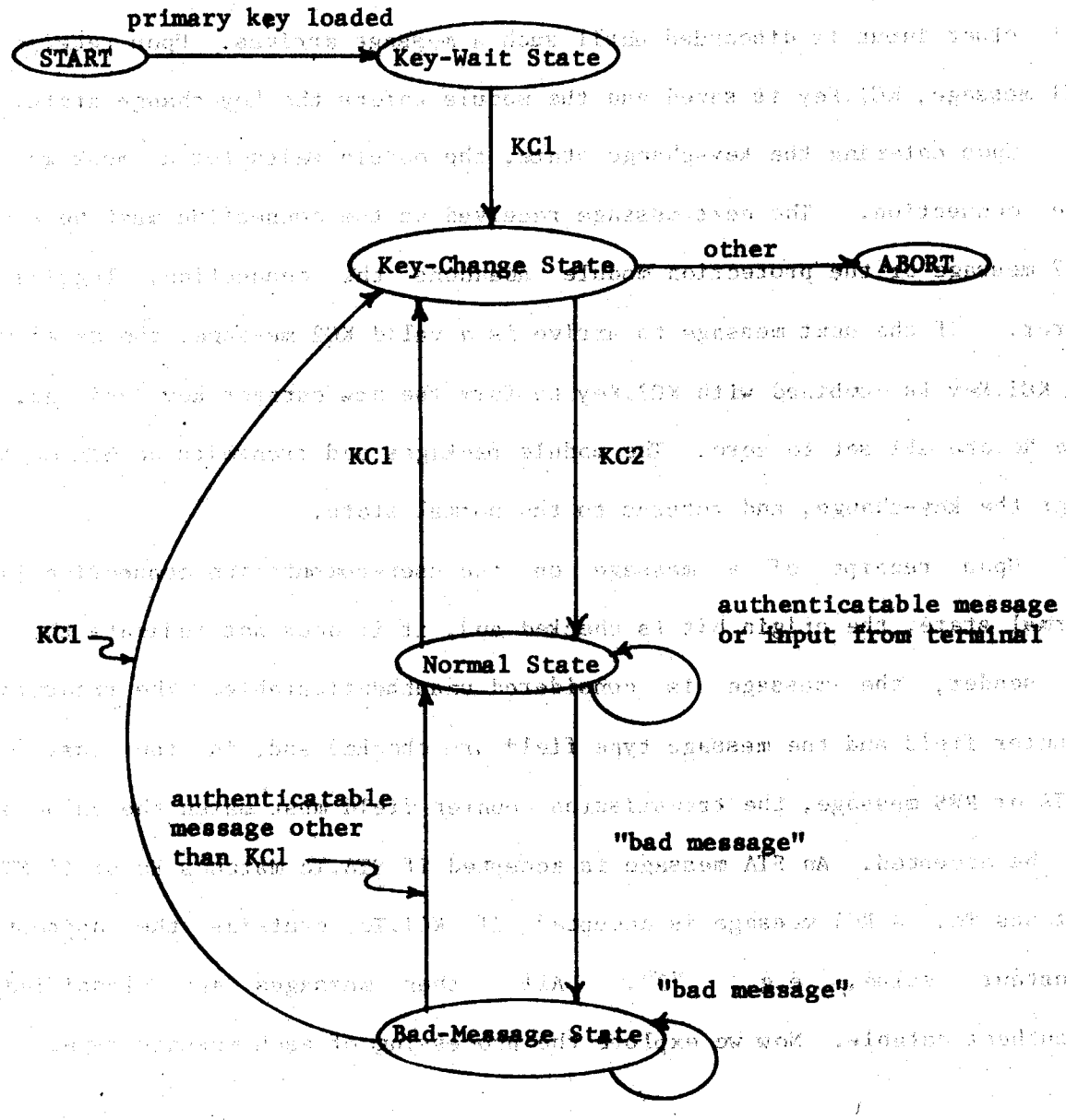


Figure 7-2

Terminal Protection Module Control Structure

A DATA message is processed by removing the number of characters indicated by DATA.CC and forwarding them to the terminal. Rc is incremented and the module enters the normal state.

An RFS message requires logging any errors on the connection as indicated by differences between the pairs (RFS.Ac, Ac) and (RFS.Rc, Tc). Then Ac is set to the maximum of Ac and RFS.Ac, and Rc is set to one greater than the maximum of Rc and RFS.Tc. The data field of a responding STA message is constructed using Ac and Tc and the message is packaged and transmitted. The module then returns to the normal state.

Receipt of an STA message also requires logging any connection errors indicated by differences between the pairs (STA.Ac, Ac) and (STA.Tc, Rc). Then Ac is set to the maximum of Ac and STA.Ac and Tc is set to the maximum of Rc and STA.Tc. The module then returns to the normal state.

At the terminal, when a KC1 message is received, KC1.Key (the first half of the new key) is saved in a temporary location and the module enters the key-change state.

When a bad (unauthenticatable) message is received in the normal state, the module constructs an RFS message, using the values of Rc and Ac, and packages and transmits the RFS. The error is logged and, if the bad message is a DATA message, the module forwards the characters in the data field to the terminal. (5) The module now enters the bad-message state.

---

(5) In order to avoid flooding the terminal with warning messages when one of a series of message from the host is lost or garbled, the module could preface the collection of unauthenticatable messages with a suitable warning. It could then process subsequent "bad" messages without issuing further warnings as long as the arriving messages are otherwise "good" data messages that have authenticator values that are consistent with the first unauthenticatable message received. When resynchronization is effected, another message would

Upon entry into the bad-message state, the module awaits input from the connection. Arriving ciphertext input is deciphered and analyzed as in the normal state. If the input is valid, it is processed as in the normal state and after the processing is complete the module returns to the normal state. Receipt of an unauthenticatable message in the bad-message state results in logging of the error and a return to the bad-message state.

Now we have completed the description of ciphertext processing by the terminal protection module and we turn to cleartext processing. In order to simplify this discussion, cleartext input to the module is assumed to consist of the data field and character count for constructing a DATA message. The interface presented is simpler than if we assumed character-at-a-time input and had to make provision for a separate signal indicating the end of a logical unit of correspondence. Whenever cleartext input is received, the character count and data are combined and packaged into a DATA message and transmitted. The module then returns to the normal state.

The protection module can also receive a control signal from the terminal keyboard indicating that a high priority message is to be sent. (6) Then an attention message is constructed with an empty data field, packaged and transmitted on the attention channel. A data mark message is constructed with

---

be issued by the module telling the user that the "window" of "bad" messages has ended, thus bracketing the "bad" messages for the user. Although this feature is not included in the terminal protection as described in Figure 7-2, it could be included with only minor additions to the module.

(6) The terminal-to-protection module interface we have assumed assures us that previously entered regular keyboard input has already been packaged and transmitted before this control signal is received. Although this precludes the transmission of a high priority message while the terminal is in the bad-message state, this is not considered to be a problem, as it may not be desirable to send a high priority message until the connection has been resynchronized.

the value of Ac as DMK.Ac and is packaged and transmitted on the regular terminal-to-host channel. The module now returns to the normal state.

Finally, we note that the interface to the terminal could provide the ability for a user to force construction, packaging, and transmission of an RFS message while in the normal state. After the RFS message was sent, the module would return to the normal state. This feature is not illustrated in Figure 7-2.

Now we turn our attention to the host protection module, which we describe in terms of its differences with the terminal module. The differences result from the fact that the host is the sender (rather than the receiver) of key-change messages, the receiver (rather than the sender) of attention and data mark messages, and because of the use of timeouts at the host. In order to simplify this description, we assume that the host module always receives a ciphertext block upon its arrival at the host end of the connection (see chapter six), without having to wait for a request from the user computation for more input. (7) We also assume that there is no buffering between the host protection module and the connection management module (CMM), so that it is not necessary to notify the connection management module upon receipt of a data mark message nor is it necessary to transform the data mark message into a reserved character. As an aid in following the discussion that follows, refer to Figure 7-3.

---

(7) This corresponds to a communication system organization in which no buffering of input from the connection occurs before processing by the host protection module (see chapter six).

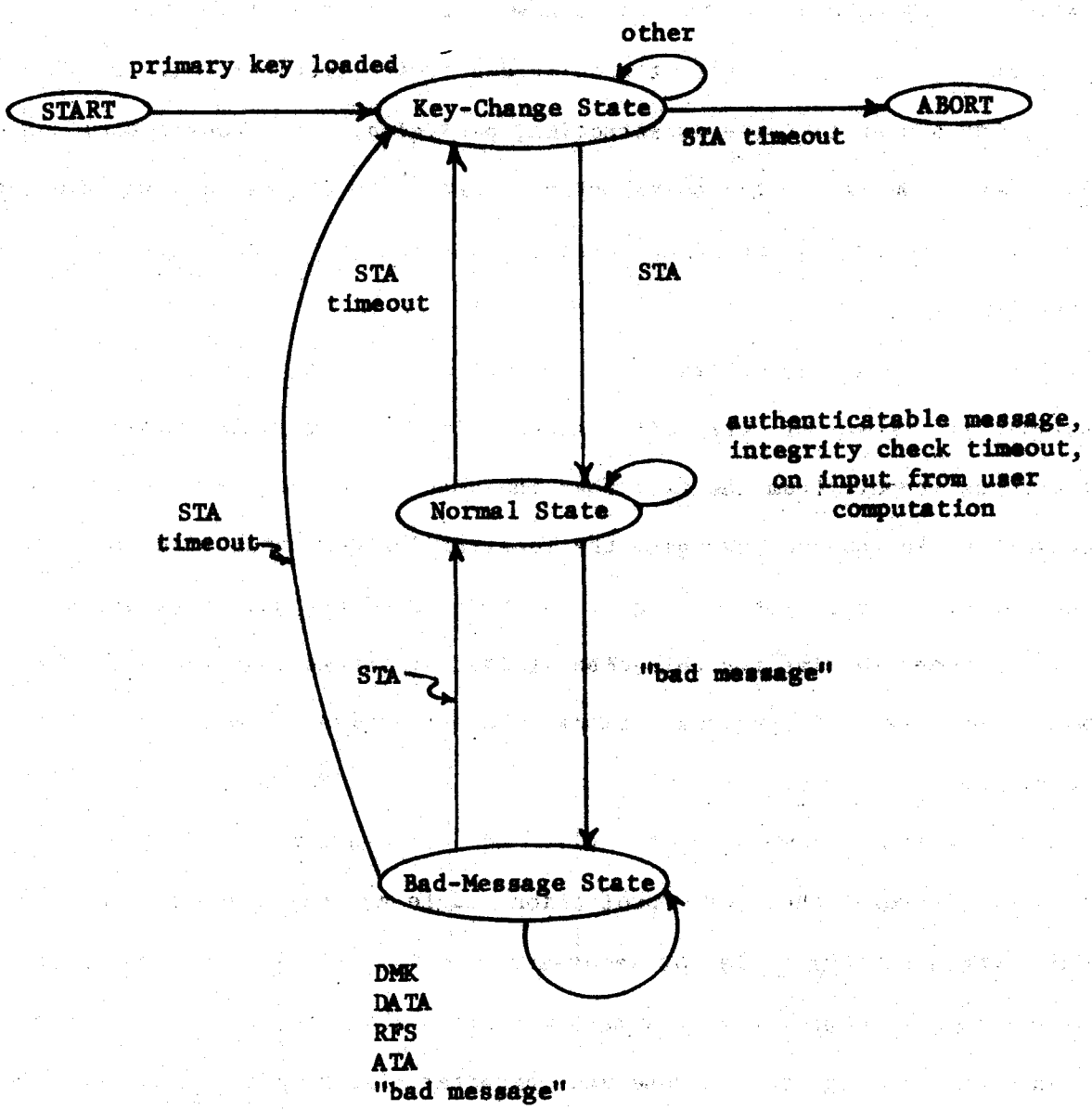


Figure 7-3

Host Protection Module Control Structure

In the normal state, the host module is blocked waiting for cleartext and ciphertext input and a timeout. In the key-change and bad-message states, the module is blocked waiting for ciphertext input and a timeout.

There are two types of timeouts used by the host module, although only one is pending at any instant. (8) The first type, an integrity check timeout, is used to periodically trigger a connection integrity check. The second type, an STA timeout, is used when the module is waiting for an STA message on the connection.

Message authentication by the host module is very similar to authentication carried out by the terminal module. Only messages with an origin bit indicating the terminal as sender are analyzed further. The counter-based authentication criteria at the host are the same as at the terminal for DATA, RFS, and STA messages. DMK messages are accepted under the same criteria as DATA and RFS messages. ATT messages are authenticated based on the value of Ac.

The host module is initialized, after the cleartext login identifier has been received, by loading the primary key, retrieved from a host data base, as the current key and entering the key-change state.

In the key-change state, the protection module generates a secondary key and constructs two key-change messages, each containing half of this new key in its data field. The KC1 and KC2 messages are packaged and transmitted in order. The module changes the current key to be the secondary key just sent

---

(8) Timeouts are modeled in the control structure through the use of two primitive operations: establishing a timeout and cancelling a timeout. Establishing a timeout involves specifying an elapsed time interval after which the timeout wakeup should occur. A timeout that is cancelled will never generate a wakeup.

to the terminal, resets Tc, Rc, and Ac to zero, and logs the start of the login session for this connection. The module establishes an STA timeout and awaits this timeout and input from the connection. The module is waiting for a valid STA message, discarding all other ciphertext input. If the STA timeout occurs before a message arrives on the connection, the module abandons the connection and logs the error. When a valid STA message arrives, the module processes it, cancels the STA timeout, establishes an integrity check timeout, and enters the normal state. (9)

Upon receipt of a DATA message, the host module performs the same processing as the terminal module, in this case forwarding the characters to the user computation via the connection management module.

Receipt of an RFS message results in the same counter adjustment, error logging, and transmission of an STA as performed at the terminal.

Receipt of an STA message results in the same counter adjustment and error logging as performed by the terminal module. The pending STA timeout is cancelled and an integrity check timeout is established.

When an ATT message is received, the exact form of processing is system specific, as noted in chapters five and six. As we are assuming an environment in which ciphertext messages are forwarded to the protection module upon arrival at the host, the module just logs arrival at the host of the ATT message and returns to the normal state, awaiting the DMK message. If an intervening buffer were present, interaction with the communication system

---

(9) The host protection module can maintain the total number of times the key-change protocol has been invoked and compare this value to a user-specifiable limit. If the limit is exceeded, the module will abandon the connection and log the error. This provides the user with a means of controlling the amount of resources expended in resynchronization efforts.

might be necessary to cause input buffered before the protection module to be forwarded to the module, in order to search for the DMK.

When a DMK message is received, DMK.Ac is compared with the host value of Ac to determine if there are any attention messages unaccounted for. If a buffer were present between the protection module and the CMM, a data mark character would be inserted into that buffer, the count of data mark messages received would be incremented, and a signal would be sent to the CMM. In any case, if there are no attention messages unaccounted for, the module returns to the normal state. If one or more attention messages are missing, an RFS message is constructed, packaged, and transmitted and an STA timeout is established.

When an integrity check timeout occurs, the module constructs, packages, and transmits an RFS message and establishes an STA timeout. This timeout will be cancelled only by receipt of a valid STA message or upon entry into the bad-message state. The module returns to the normal state. When a STA timeout occurs, the module enters the key-change state. If an intervening buffer were present, it would first be necessary to ascertain that the STA message was not in that buffer before the transfer to the key-change state was effected.

Upon receipt of an unauthenticatable message in the normal state, the module logs the error, constructs, packages, and transmits an RFS message. The integrity check timeout is cancelled and an STA timeout is established. The module now enters the bad-message state.

Once in the bad-message state, the module waits only for ciphertext input and the STA timeout. Receipt of additional bad-messages results in logging of

their arrival, but no additional RFS messages are transmitted. (10) Receipt of a valid DATA, RFS, ATT, or DMK message results in processing just as in the normal state, but the module remains in the bad-message state. Only receipt of an STA message will cancel an STA timeout, establish an integrity check timeout and return the module to the normal state directly. If the STA timeout occurs, then the module enters the key-change state.

Processing of cleartext input by the host module parallels that of the terminal module and is simplified by the lack of the high priority message signal.

Summary

This chapter presented the formats of the seven message types used to implement the protection protocols described in earlier chapters. All of these messages share a common format that permits easy identification and authentication through standard location of the authenticator and message type fields in the message block. The control structure of the host and terminal protection modules is presented.

The host module is more complex that the terminal module, incorporating mechanisms for automatic detection of connection blockage, initiating key-change procedures, and assuming final responsibility for resynchronization efforts, reflecting the greater computational power available at the host. Provision is made for the user to exert influence over the reaction of the

---

(10) The host module can maintain totals on the number of bad messages received and the number of consecutive bad messages received and effect a key change or abandon the connection if these totals exceed user-definable limits. This provides another means of permitting the user to exercise influence over the amount of resources spent in attempting to resynchronize the connection.

protection modules to possible intruder threats or extreme channel error conditions, preventing excessive resource commitment to resynchronization and recovery attempts.

Chapter Eight

Implementation on Multics

This chapter describes the structure and operation of a test implementation of the protection protocols on the Multics system and explains some of the considerations involved in designing an implementation that could be incorporated into a production Multics system. The test implementation was undertaken to test the completeness of the proposed design and to evaluate the impact of the protection protocols upon the human interface of a computer utility.

Structure of the Test Implementation

As illustrated in Figure 8-1, the test implementation uses four distinct processes on the Multics system [MIT] to simulate a user controlling a computation through a connection protected by the modules developed in this thesis. Each process communicates with adjacent processes by means of ARPANET connections [RW].

One process simulates the functionality of the terminal protection module, handling cleartext and ciphertext input as described in chapter seven. This process is created by logging in from any terminal and invoking the terminal module simulation program. It reads input from the terminal through standard Multics input facilities. In order to more accurately simulate transmission loads, erase-kill processing and canonicalization are not

performed on the input stream from the terminal to this module. Instead, such operations are performed at the target process. When echoing is employed, the echo processing is performed on the input stream from the terminal to the terminal module process, emulating the style of remote controlled transmission echoing described in chapter six.

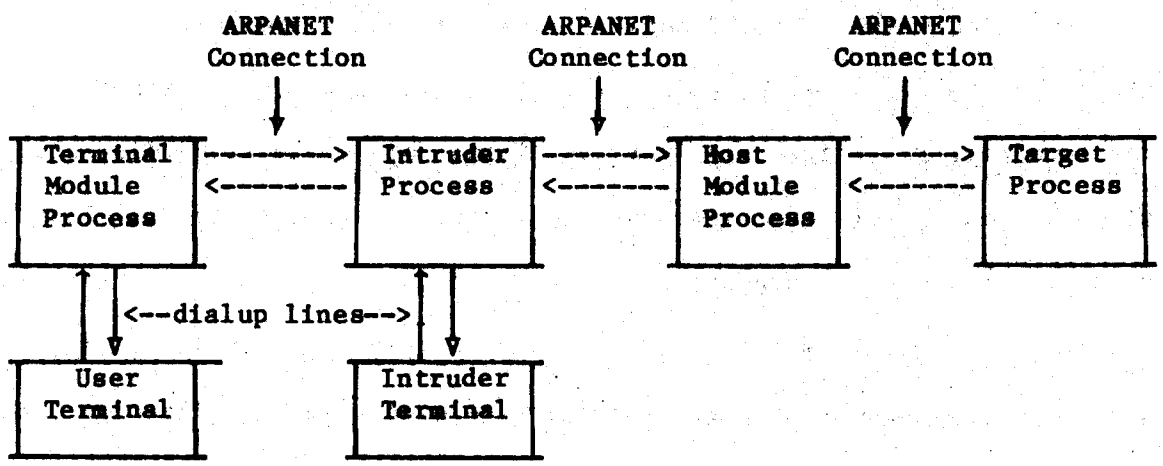


Figure 8-1

Configuration of the Test Implementation on Multics

A condition handler is established in the terminal module process for the "quit" condition, the only high priority message recognized by Multics. Upon receipt of a quit from the terminal, this process transmits a data mark message on the regular terminal-to-host channel. No separate attention channel is employed. There is no need to transmit an attention message, as there are no demand buffers in the ARPANET connection between the terminal module process and the host module process.

The intruder computer in the connection is simulated by a process situated between the terminal and host protection module processes. All message traffic between the two ends of the connection passes through the

intruder process. This process is created by logging in from any terminal and invoking the intruder computer simulation program. The intruder process has the responsibility of logging in the host module process over the ARPANET and initiating the execution of the program that simulates the host module functions within that process. The intruder also acts as the initiator of the ARPANET connection between itself and the terminal module process.

Several commands are provided for the intruder (the person at the intruder terminal) to engage in various forms of connection disruption. Provision is made for the intruder to remove message blocks traveling in either direction over the connection. The intruder can cause a selected message to be copied from the connection and inserted into the connection at any future time. Spurious message blocks can be generated and inserted into the connection at any time. Message blocks from either end of the connection can be rerouted to their sender. The intruder can monitor the traffic on the connection in one or both directions selectively. All of the operations noted above, with the exception of the copy operation, can be performed on one or more message blocks as specified by the intruder in the command. (1)

The host protection module process implements the control structure described in chapter seven and maintains a log of important events that occur on the cipher connection. The protection module log can be examined during or after a login session to review abnormal channel activities as observed by the

---

(1) Note that the intruder does not posess any commands that enable him to engage in actual cryptanalysis of the message traffic he observes. It is felt that the analysis presented in the appendix indicates that such actions are not practically performed in real time during the login session. Moreover, adequate facilities for such cryptanalysis are not available to the author for inclusion in the test implementation.

host. This process creates the target process, logging it in via an ARPANET connection, at the beginning of a test session. The host module process receives output from and forwards input to the target process via this ARPANET connection. Upon receipt of a valid data mark message, the host module process sends a quit to the target process, using this ARPANET connection.

The target process is a regular Multics process in which the user of the test implementation may perform computations just as with any process logged in directly from a remote terminal. The target process acts as though it were attached to a user terminal over the ARPANET, in terms of terminal-specific input/output transformations. It is in the input stream to this process that the functions of erase-kill processing and canonicalization are finally performed.

The login protocol described in chapter three is not implemented. After logging the terminal and intruder processes into Multics in the usual fashion and initiating the execution of the appropriate simulation program in each process, the terminal user merely responds to a query to begin the session. The first output on his terminal after this is the login greeting from the target process logged in for him.

The problem of loading the primary key into the protection modules at both ends of the connection is handled by maintaining a key in a shared segment that both the terminal and host module processes access. This segment does not serve as a communication vehicle between the two processes in the sense of any of the functions that are associated with the module control structure.

The programs for the test implementation were coded in PL/I, with the exception of the encryption algorithm, which was coded in assembly language in order to take advantage of bit manipulation instructions not accessible from PL/I. 128-bit blocks were used and were enciphered using a software version of IBM's Lucifer algorithm [Ben] that is available on Multics. This software implementation can encipher/decipher a block in approximately four milliseconds.

A twenty-four bit authenticator is employed in the messages, along with a six-bit message type field. This permits the transmission of from one to fourteen 7-bit ASCII characters in a user data message. Because user data messages are the most frequently transmitted messages, the authenticator size was chosen to result in a full block for this message type.

Results

The implementation was tested on several occasions with a human controlling the process that simulated the intruder computer. A variety of attacks on the connection were attempted, including message rerouting, message deletion, generation of spurious messages, and insertion of copies of old messages. These attacks were carried out with complete knowledge of the operation of both the terminal and host modules so that very specific types of message stream modification were effected, e.g., deletion of a request for status or status message during connection resynchronization. The protocols performed as expected, detecting each act of message stream modification or denial of service, reporting these acts to the user and the host, and restoring normal communication on the connection if possible.

The impression given to a user of the test implementation must be tempered by several considerations. The delays that tend to degrade the response time to commands issued by the user are a result of several factors, most importantly the seven process schedulings required by a complete roundtrip interaction. The fact that process scheduling is the most important factor in the perceivable delay is evidenced by variations in the delay under different system loads.

During extended periods of input, e.g., while entering text into an editor, or while executing commands that usually have noticeable delays associated with them (the PL/I compiler) no apparent differences in response time are observed. Similarly, while issuing commands that tend to deliver large amounts of output to the terminal, the user of the test implementation is not generally aware of the intermediate processing going on between his terminal and his target process. This is especially true if the user is typing ahead, through his output, so that the response delay can be hidden by the continuing output from previous interactions. Characteristic of the performance of the test implementation under light system loads (30 users) is the fact that it is able to drive a 1200 bps terminal at capacity during output from the host, although it could not drive a 2400 bps terminal similarly.

User experience with the test implementation led to the idea of "bracketing" a series of messages from the host that arrive after the loss or destruction of an earlier host message in the series, rather that repeating an error message with each successive, unauthenticatable host message. It has also been suggested that some means of "replaying" to the user the last good

message (or messages) received should be provided so that he can resume input from a known place, after disruption of the connection. This feature is easily added to the host module software. In a similar vein, is might be desirable for the host module to forward to the terminal the error messages that are being placed in the host cipher log, on a user-controlled, selective basis.

Overall, the performance of the protocols in this test implementation suggests that, if a suitably fast implementation were used, the impact on the human interface of a computer utility should be negligible.

Considerations for a Production Implementation

We now examine how a production version of the host protection module might be incorporated into the Multics system. The discussion is meant to provide some indication of the considerations involved in a production implementation of the protocols in an existing system and should not be construed as a model for all systems, as Multics does not exhibit all of the potential complexity possible in a host communication system.

The description of the internal organization of portions of Multics, as presented below, has been simplified in some places where the loss of detail was felt to be irrelevant to this discussion. The description of the structure of the input/output system reflects ongoing and planned modifications to the Multics Communication System.

Multics employs a front end processor as an interface for dialup communication lines (but not for the ARPANET). This front end processor is not considered secure, implements only very primitive supervisor facilities

and is programmable only in an assembly level language. It is a multiplexed communication facility, as described in chapter six, and it communicates with the central processor via a direct memory interface.

The front end processor buffers terminal input and forwards this input to the central processor upon receipt of a newline character. Thus, it engages in recognition of the newline as a break character. Multics is accessed primarily by asynchronous terminals, and multi-character substitution echoing is performed by the front end processor if requested. Finally, high priority messages in the form of "line breaks" on asynchronous lines are recognized by the front end processor as "quits", causing it to discard any input or output buffers it holds for the signalling line and to notify the central processor of receipt of this high priority message.

In the central processor, two levels of input/output processing are involved: the supervisor level and the user level. At the supervisor level in Multics, input from the front end processor is copied into multiplexed core-resident buffers and then into private buffer areas for each user. Output from user processes is copied into core-resident buffers and transferred to the front end processor. Thus, at the supervisor level, only buffer management is performed. At the user level in the central processor, the transformation operations noted earlier for input (translation, canonicalization, erase-kill processing, and escape sequence processing) and output (translation and formatting) are implemented.

Multics also performs input and output to remote terminals via an interface to the ARPANET. This ARPANET interface does not involve the front end processor, but appears to the central processor as a peripheral device.

The software structure of this interface involves three major levels of processing. At the lowest level is the module that acts as the device handler for the ARPANET IMP to which Multics is connected. This module is interrupt driven and operates in the supervisor, implementing the host-IMP protocol [BBN] of the ARPANET and managing multiplexed buffers of data for the IMP data channel. Logically above the IMP interface module, but still operating in the supervisor, is the network control program, which implements the ARPANET host-host protocol [ARP] and provides for the multiplexing of the network interface among Multics users. Finally, higher level protocols, e.g., file transfer, telecommunication network, and initial connection protocols [ARP], are implemented in each user's process in the user level.

Over the ARPANET, attention messages are transmitted on a separate logical channel and are directed to a special network process for handling. The network process, a trusted, privileged process, determines the user process for which the attention message is destined and handles it appropriately. It also monitors all of the network connections to Multics and handles error conditions raised at the host-IMP protocol level.

The memory protection facilities of Multics provide multiple address spaces, each with eight linearly ordered rings of protection [Sal2, SS2]. The system gives each process its own address space in which the supervisor functions execute in the most privileged rings (0 and 1) and user procedures execute only in the higher rings (4-7).

For a production implementation of the protocols developed in this thesis, we propose that each cipher connection be provided with a separate

process to execute the host protection module. This process would reside in ring two or three of the address space of the corresponding user process. (2)

Sharing the address space with the corresponding user process makes each protection module process relatively inexpensive. Executing in ring two or three protects each module from the user ring programs, but still provides an execution environment that is private for each user connection, above the multiplexed buffers managed at rings zero and one. Finally, by making each protection module a distinct process, it can be simply programmed to manage only one connection, accepting each ciphertext block as it arrives without waiting for demands for input from the corresponding user process.

The front end processor must be aware of the connections that will be using the protection modules, so that it can accept the enciphered input and forward it to the central processor a block at a time. On synchronous communication lines this should pose no problem as entire enciphered blocks can be transparently transmitted using synchronous line control protocols [ISO, IBM1]. On asynchronous lines this may require assembling character-size pieces of a ciphertext block until a complete block is formed. Some form of block framing may also be desired in order to insure that entire blocks are forwarded to the host module, for if block frame synchrony is lost, the

---

(2) While the current process implementation forces each process to have its own address space, an implementation of processes that would permit two or more processes to share an address space in this fashion has recently been developed by Reed [ReD]. Using the current process implementation, one can avoid the cost of a separate process with its own address space for each protection module by multiplexing a single trusted process among all cipher connections. However, this savings is achieved at the cost of increasing the complexity of this process, as it must now manage many connections at once, and violating the security principle of least common mechanism noted in chapter six.

connection must be manually suspended and re-established in order to resume communication.

Of course echoing can no longer be performed by the front end processor and some substitute for this must be provided as outlined in chapter six. On half-duplex lines, line breaks must still be used to terminate output from the host and turn around the line for terminal input, but a quit should be sent to the user process only upon receipt of a valid data mark message. On full-duplex lines, line breaks need no longer be sent since attention messages can be transmitted on the terminal-to-host channel with assurance of being processed rapidly by the host module.

The protection module process would accept input ciphertext blocks upon arrival at the central processor from the supervisor level buffer management software for both dialup lines and ARPANET connections, process them as outlined in chapter six, and place the deciphered input into buffers for user level input processing. Output from a user process would be processed by this module and ciphertext blocks would be forwarded to the supervisor level buffer management software.

We also propose the introduction of a hardware encryption instruction capable of enciphering/deciphering one or more 64-bit blocks using the NBS data encryption standard. Such an instruction would be a logical extension to the multiple-operand extended instruction set used for character and bit manipulation on Multics [Hon]. This instruction could be used to encipher

data both on communication channels and for protection of data stored on removable media, e.g., tapes and demountable disk packs. (3)

Performance Considerations

There are two major areas of system performance that will be affected by use of the encryption protection modules: host system overhead in supporting the protocols and connection bandwidth utilization and delay resulting from their use. Host system overhead involved in supporting the protocols includes the processor and memory resources required to decipher and authenticate incoming messages, to encipher and tag output, and the processing involved in resynchronization, key-change, and denial of message service protocols. The overhead for resynchronization is encountered only when connection disruption occurs and should be considered as a marginal cost, except when such disruption is a major problem. The time dedicated to detection of denial of message service is controllable by parameters that should be user definable, thus permitting the cost of this protection to be controlled by the user, within limits established by installation parameters.

Examination of the control structure of the protection module indicates that most of the time, under usual circumstances, would be spent in the task of regular data message processing. The operations involved in this task are all readily programmable on modern host systems, as long as a hardware enciphering/deciphering instruction is provided. The associated overhead per message block would be on the order of 50-100 microseconds on a large host

---

(3) The details of the operation of such an instruction will vary based on the architecture of the host computer, and the design of such an instruction is a topic requiring further study.

system, given a hardware enciphering/deciphering instruction capable of deciphering a block in 5-50 microseconds. Since a 64-bit block could hold five or six characters when used as a user data message block, this overhead is about 9-20 microseconds per character for full data blocks.

Additional overhead is involved through the use of multiple processes to implement the host protection module functions and other communication system functions, but a comparison between this organization and the current system organization is hard to make. Experience using multiple processes to provide echoing over the network indicates that the overhead involved in such organization is not substantial. The working sets of the processes involved are small and the functions provided are rather simple and execute rapidly.

With respect to transmission bandwidth, it is reasonable to ignore the effects of messages associated with resynchronization, key-change, and detection of denial of message service protocols, as these messages should constitute a very small fraction of the total message traffic. The reduction of bandwidth over the connection is a result of dedicating a portion of each message block to authentication and message type information. In a 64-bit block, this information would occupy about 25% to 35% of the block. (4) Thus only 65% to 75% of the connection bandwidth is available for user data transmission. On input bandwidth utilization is usually not a problem, as the user rarely is capable of taking advantage of the available bandwidth on the

---

(4) In a 64-bit block, five or six characters can be accomodated with space for a four-bit message type field and an authenticator that provides a probability of erroneous authentication on the order of 10^{-6}. The number of characters varies depending on character size, seven or eight bits per character, and desired authenticator size.

connection. (5) On output, however, this is a disadvantage as the host system usually is capable of using the maximum channel capacity in short bursts.

With respect to delays, the control structure of the protection modules and the discussion of host system overhead in message block processing from above indicates that the overhead for preparation, encryption, decryption , and authentication of a single message block should result in a negligible delay. Assuming a terminal module implemented using a microprocessor and a special hardware encryption chip, the total time required to process one message block should be about 100 microseconds. This indicates that the speed of the encryption protection module is not a bandwidth limiting factor for data rates associated with user-computation connections. Relative to the other processing delays encountered by interactive terminal users in their communication with a host system, the delay introduced by the use of the protection protocols is negligible.

Summary

The test implementation tested the completeness of the protocols and permitted evaluation of the impact of the protection protocols on the human interface of a computer utility. The protocols performed as expected and generally were transparent to the user. Even in situations where the intruder actively engaged in connection disruption, the impact on the user was mitigated by the automatic resynchronization protocol. With the addition of further enhancements noted above, the user interface could become quite robust

---

(5) If input to the host is via a multiplexed connection, e.g., an ARPANET connection, this reduction of bandwidth may be of concern.

in the face of dedicated intruder line disruption. The delays experienced in the test implementation were unacceptably long, but with the use of hardware encryption at both ends of the connection and the use of a microprocessor to implement the terminal protection module, it appears that the delays would become negligible.

Chapter Nine

Conclusions

The goal of this thesis was to develop a set of protocols to organize the use of encryption to provide a secure path between a user at a terminal and his interactive computation in a remote host computer. We have proposed a set of protocols to accomplish this goal, and performed a first demonstration of their feasibility. These protocols are designed for use with a block cipher such as the proposed NBS Data Encryption Standard or IBM's Lucifer, taking advantage of the fixed-length blocks to delimit data and control messages. In producing these protocols, every effort has been made to be complete and general. Provision is made for all common aspects of interactive user-computer communication -- from authentication at login, to high priority messages, to character echoing. The protocols are designed to function in a wide variety of communication system configurations.

The level of description in the thesis should be sufficient to allow an implementation to be engineered for most existing and foreseeable systems. We hope that this work will contribute to future widespread use of encryption-based protection measures to reduce the vulnerability of computer systems to release and modification of the data they contain through intrusion on their largely unprotected communication facilities. In order to achieve such widespread use of encryption-based measures, both an encryption algorithm and a set of protocols must be standardized to permit development of low cost terminal protection modules that can be used with any host that employs such

measures. We hope that this research will stimulate work on a standard set of protocols.

Future Work

Although the level of description provided in this thesis should be adequate for one attempting to engineer a terminal protection module, there are areas deserving of further study with respect to implementation of this module. It appears that the use of a general purpose microprocessor and a special purpose encryption chip should provide an adequate hardware base for the terminal module, but questions remain as to what other functions could/should be taken on by the microprocessor, e.g., remote controlled echoing and communication line interfacing. There is also the question of using different arrangements of one or more NBS encryption chips to provide a more secure cipher scheme. Hellman and Diffie have speculated [DH1] that a cipher constructed by cascading two NBS encryption chips and using independent keys would be more secure than the use of a single NBS chip. Such a modification to the protection modules is easily accomplished within the context of the protocols employed in this thesis. It would be a simple matter to extend the key-change protocol to use four messages to transmit the keys for the two cipher chips.

Another topic for future study lies in the development of production versions of the host protection module. The protocols have been designed so that the host module can be implemented in existing systems using the wide variety of host communication system configurations that may be encountered, although the task of implementing the host module probably will vary in

difficulty from one host to the next. The exact form of the host encryption/decryption instructions and the problem of managing the primary keys at the host both require careful study. Another important consideration in host implementations will be the overhead encountered by the host and the delays introduced into user interactions. Empirical analysis of the cost of supporting the protection modules and measurement of their performance should be conducted. In a similar vein, studies of the psychological impact of using the protocols should be carried out to determine how the human interface could be further improved.

It would be encouraging to see a proof of correctness of an implementation of the protection protocols developed in this thesis. The area of logical verification of protocols has received little attention so far [Boc], but will certainly be critical to the acceptance of the protocols in the construction of secure systems. Part of the difficulty of proving the correct operation of the protocols lies in establishing the formal assumptions that correspond to informal goals.

There is need to develop suitable algorithms for generating primary and secondary keys at the host. Algorithms used for this purpose should have the properties that the keys they generate are statistically well distributed yet the sequence of keys should not be predictable by someone observing successive members of the sequence and knowing the algorithm being employed. Certainly much research into this area must have been performed by agencies of the Department of Defense in conjunction with existing needs to generate keys, but it seems unlikely that many of the results of this research will become publicly available. In the public domain, Hellman [Hel] has suggested the use

of two random number generators, one generating statistically random numbers using a conventional technique, as described by Knuth, and the other generating numbers in a "non-deterministic" fashion, e.g., using the value of the real time clock as part of its functional input. Encryption keys could be fashioned by combining the output from these two random number generators using an exclusive-or operation.

Finally, it would be very interesting to see if similar protocols can be developed based on stream ciphers. The use of stream ciphers holds the promise of overcoming bandwidth utilization problems by employing variable-length messages. However, it is not clear whether protection modules and protocols developed for use with stream ciphers can be as simple as the ones illustrated in this thesis. The tradeoffs between bandwidth utilization and complexity must be carefully examined.

Appendix

Cryptanalysis

The conversion of ciphertext to cleartext by analytic techniques without knowledge of the key is a topic beyond the scope of this thesis. As noted in chapter two, it is assumed that both Lucifer and the NBS algorithm are resistant to such cryptanalytic attacks. (1) In the case of the NBS algorithm, as noted by Diffie and Hellman [DH1], the potential availability of very fast, inexpensive encryption chips, and the size of the key space for the NBS algorithm make breaking the cipher by exhaustive searching of the key space not entirely infeasible. It is ironic that the potential availability of an NBS encryption chip may make practical both the inclusion of encryption devices in terminals and the breaking of the cipher system by means formerly considered impractical. As the possibility of practical exhaustive search is of importance in assessing the level of security provided by encryption, we

---

(1) It is very hard to establish the resistance level of an encryption algorithm to cryptanalysis. If a method of analyzing the cipher is discovered then it provides an upper bound on the amount of work that may be needed to break the cipher. But if no method is found, then one has no guarantee that the cipher is unbreakable or even very hard to break, since some fresh analysis might discover a simple means of drastically reducing the work needed to break the cipher. Whenever the cipher in question is not theoretically secure, one is faced with this problem. During the development of Lucifer, IBM made efforts to determine how susceptible the cipher was to various cryptanalytic techniques. Although these efforts did not reveal any weaknesses that could be exploited by a cryptanalyst, this does not provide one with a firm basis for concluding that the cipher is practically unbreakable.

now present an analysis of the effort required to break the NBS and Lucifer ciphers by key space search.

The goal of an exhaustive key search is to determine the key used to encipher some set of message blocks. It is presumed that the analyst has available some number of blocks of ciphertext and that for some of these blocks he knows portions of the corresponding cleartext block. The key search is to be performed by a large system equipped with an array of computing elements, each capable of deciphering (or enciphering) a single block of text and comparing the result (with masking) to another block in parallel. Each element in the array can signal the result of a successful operation to a central controller. We will refer to the amount of time required to perform a single deciphering and comparison as the basic cycle time of this system.

A single element could be used to search the entire key space. By employing large numbers of the elements all operating in parallel under the supervision of some central unit, however, the amount of time required to search the key space can be reduced by a factor equal to the number of elements employed.

Now that we have a model for the key search process, some discussion of the size of the key space and the expected duration of the search is possible. For the 128-bit Lucifer, the key space contains approximately 3.4×10^{38} keys, while the NBS algorithm, using a 56-bit key, has a key space containing only approximately 7.2×10^{16} keys. Note that, on the average, only half of the key space need be searched if the correct key can be recognized when encountered. The conditions under which an analyst can know he has the correct key will be discussed later.

We will now make a simplifying assumption about the nature of the cipher that is being analyzed. The assumption is that the cipher is approximately perfect. A perfect cipher has the property that no two distinct keys will transform two distinct cleartext blocks into the same ciphertext block [Sha]. (2) Although Lucifer and the NBS algorithm are not necessarily perfect, it is believed that they probably do not deviate significantly from this property [FH3]. In the case of a perfect cipher, an analyst who possesses one ciphertext block and the complete matching cleartext can now determine, by exhaustive searching of the key space, which key was used to encipher the block, because only one key will transform a specific cleartext block into a specific ciphertext block. If an analyst knows all but k bits of the cleartext in an intercepted ciphertext block, there are 2^k keys that will correctly decipher the known portion of the block while the unknown bits range over all the possible values that k bits may take on.

When an analyst has several blocks and portions of the cleartext associated with each, it is reasonable to ask how many keys will be in the set that results from intersecting the results of the key searches for each of the incompletely known blocks. Let K be the size of the key space, N be the number of unknown bits in each intercepted block, and J be the number of such intercepted blocks. Then the expected size of the set that results from the intersection of the "possible" key sets for each intercepted block, E(I), is given by the following expression.

---

(2) For purposes of exhaustive key searching, "perfection" constitutes a worst case assumption. In the case of a non-perfect cipher, an intruder may discover several keys that correctly decipher a known intercepted block of ciphertext and he must further test to determine which one is the key used to encipher the collection of messages in which he is interested.

$$E(I) = (K-1)^{1-J} * (2^{N} - 1)^{J} + 1$$

The meaning of this result is that the possession of only a few blocks and the knowledge of only a modest fraction of the bits in each block reduces the expected size of the intersection key set to less than two. In the case of the NBS algorithm, with only two intercepted blocks and 36 bits known in each block (56% of the block) an analyst can discover the key used to encipher the blocks in a two-phase operation. All but a few of the array elements can be put to work simultaneously deciphering one of the two blocks with a number of different keys. Whenever one of the elements finds a key that correctly deciphers the known portion of the first block, one of the otherwise idle elements will decipher the second block with the same key. Despite the incomplete information available to the analyst, this procedure will usually produce only one key that correctly deciphers the known portions of both blocks.

Despite the arguments presented above, there is still an overriding question that has not been considered: How long will it take to search the key space? We have noted that the time involved in the key space search is inversely proportional to the number of elements in the array; adding more elements reduces the time required to perform the search. Let us examine a concrete example to put the question into perspective.

Diffie and Hellman have proposed a scenario in which a deciphering device similar to the one described above is constructed [DH1]. They suggest that the special purpose chips can be made with a cycle time of one microsecond at a cost of about $10 per chip, and they propose the construction of an array of

1,000,000 chips and associated controlling and power supply hardware, costing again as much as the array of special purpose chips, for a total system cost of $20,000,000. They point out that such a system could search half the key space of the proposed NBS algorithm in about one day, given a matching clear and ciphertext block. On the other hand, the time required for a similar search of the key space of the Lucifer algorithm is about 10^{19} years.

Our earlier results on exhaustive searching of the key space given only partial matching blocks of clear and ciphertext indicate that more time would be required to successfully determine the key under such circumstances, but the extra time involved should not be substantial enough to change the general nature of figures put forth by Diffie and Hellman.

Thus, while it is not feasible to consider exhaustive key searching as a means of discovering the key used in a Lucifer based system, it is not unreasonable to consider such an attack on a system based on the NBS cipher. As Diffie and Hellman point out, these calculations are especially disturbing when the projected improvements in hardware speed and reduced hardware costs of the next decade are taken into consideration. Similar calculations can be performed assuming different system cycle times, numbers of array elements and costs. Basically, though, it is apparent that a determined analyst with adequate resources can determine the key used to encipher potentially large volumes of data under the NBS cipher within a reasonable time period, given some knowledge of the contents of intercepted ciphertext blocks.

Bibliography

[ARP] Advanced Research Projects Agency, "ARPA Current Network Protocols," NIC 7104, Network Info. Center, Stanford. Res. Inst., Menlo Park, CA., Dec. 1974.

[Bar] Baran, P., "On Distributed Communications: Vol. IX, Security, Secrecy, and Tamper-Free Considerations," Rand Memo. RM-3765-PR, August 1964.

[Ben] Benedict,G., "An Encryption Module for Multics," BSc Thesis, M.I.T., Dept. of Electrical Engineering, May 1974. (Also available as M.I.T. Laboratory for Computer Science Tech. Memo. TM-50.)

[Bob] Bobrow, D., et al., "TENEX, a paged time sharing system for the PDP-10," CACM 15, 3 (March 1972), pp. 135-143.

[Boc] Bochman, G., "Logical Verification and Implementation of Protocols," Proc. Fourth Data Communications Symposium, Oct. 1975, pp. 7.15-7.20.

[BBN] Bolt, Beranek, and Newman, "Specifications for the interconnection of a host and an IMP," Bolt Beranek and Newman, Inc., Cambridge, Mass., BBN Rep. 1822, Dec. 1974.

[Bra1] Branstad, D., "Security Aspects of Computer Networks," AIAA Computer Network Systems Conference, April 1973, paper 73-427.

[Bra2] Branstad, D., "Encryption protection in computer data communications," Proc. Fourth Data Communications Symposium, Oct. 1975, pp. 8.1-8.7.

[Bri] Bright, S., "Simulation Confirms Proposed Encryption Algorithm," Computerworld 9, 47 (Nov. 1975), p. 8.

[Cla] Clark, D., "An Input/Output Architecture for Virtual Memory Computer Syetems,"PhD Thesis, M.I.T., Dept. of Electrical Engineering, Jan. 1974. (Also available as M.I.T. Laboratory for Computer Science Tech. Rep. TR-117.)

[DH1] Diffie, W. and Hellman, M., "A Critique of the proposed Data Encryption Standard," CACM 19, 3 (March 1976), pp. 164-165.

[DH2] Diffie, W. and Hellman, M., "Multiuser Cryptographic Techniques," to appear in 1976 NCC AFIPS Conf. Proc., June 1976.

[FH1] Feistel, H., "Cryptographic coding for data bank privacy," IBM Corp. Res. Rep. RC 2827, March 1970.

[FH2] Feistel, H., "Cryptography and computer privacy," Scientific American 228, 5 (May 1973), pp. 15-23.

[FNS] Feistel, H., Notz, W., Smith, J., "Cryptographic techniques for machine to machine data communications," IBM Corp. Res. Rep. RC 3663, December 1971; also in IEEE Proc. 63, 11 (Nov. 1975), pp. 1545-1554.

[Hel] Hellman, M., "The Information Theoretic Approach to Cryptography," Stanford Univ., Center for Sys. Research, April 1974.

[Hon] Honeywell Information Systems, Inc., Honeywell Macro Assembler Program Manual for Series 600/6000, BN86 Rev. 2, March 1973.

[Hub] Huber, A., "A Multi-Process Design of a Paging System," SM and EE Thesis, MIT Dept. of Electrical Engineering and Computer Science, June 1976. (To appear as M.I.T. Laboratory for Computer Science Tech. Rep.)

[IBM1] IBM, "Synchronous Data Link Control General Information," IBM Corp. Sys. Ref. Lib., GA27-3093-0, March 1974.

[IBM2] IBM,"IBM 3600 Finance Communication System 3614 Programmer's Guide," IBM Corp., GC27-0010-0, 1974.

[ISO] International Standards Organization, TC97/SC6, Document 1005.

[KD1] Kahn, D., "The Code Battle," Playboy 22, 12 (Dec. 1975), p. 134.

[KD2] Kahn, D., The Codebreakers, Macmillan, New York, 1967.

[KaR] Kahn, R., "The organization of computer resources into a packet radio network," in 1975 NCC, AFIPS Conf. Proc. 44, pp. 177-186.

[MIT] M.I.T. Laboratory for Computer Science "An Introduction to Multics," Tech. Rep. TR-123, Feb. 1974.

[NBS] National Bureau of Standards, Computer Data Protection, Federal Register 40, 52 (March 1975), pp. 12067-12250.

[RK] Reed, D. and Kanodia, R., "Eventcounts: A New Model for process Synchronization," (to appear), Jan. 1976.

[ReD] Reed, D., "Processor Multiplexing in a Layered Operating System," SM Thesis, MIT Dept. of Electrical Engineering and Computer Science (in progress). (Also to appear as a M.I.T. Laboratory for Computer Science Tech. Rep.)

[RW] Roberts, L., and Wessler, B., "Computer Network Development to Achieve Resource Sharing," in 1970 SJCC, AFIPS Conf. Proc. vol. 36, pp. 543-549.

[Sal1] Saltzer, J., "Traffic Control in a Multiplexed Computer System," ScD Thesis, MIT Dept. of Electrical Engineering, 1966. (Also available as M.I.T. Laboratory for Computer Science Tech. Rep. TR-30.)

[Sal2] Saltzer, J., "Protection and the Control of Sharing in Multics," CACM 17, 7 (July 1974), pp. 388-402.

[SO] Saltzer, J. and Ossanna, J., "Remote terminal character stream processing in Multics," in 1970 SJCC, AFIPS Conf. Proc. 36, pp. 621-627.

[SS1] Saltzer, J. and Schroeder, M., "The protection of information in computer systems," Proc. IEEE 63, 9 (Sept. 1975), pp. 1287-1308.

[Sav] Savage, J., "Some Simple Self-Synchronizing Data Scramblers," Bell Sys. Tech. Jour., 42, 2 (Feb. 1967), pp. 449-487.

[ScP] Schmid, P., "Review of Ciphering Methods to Achieve Communication Security in Data Transmission Networks," Proc. Int. Zurich Seminar on Digital Comm., March 1976.

[Sch] Schroeder, M., "Engineering a Security Kernel for Multics," Operating Systems Review 9, 5 (Nov. 1975), pp. 25-32.

[SS2] Schroeder, M. and Saltzer, J., "A Hardware Architecture for Implementing Protection Rings," CACM 15, 3 (March 1972), pp. 157-170.

[Sha] Shannon, C., "Communication Theory of Secrecy Systems," Bell Sys. Tech. Jour. 28, 4 (Oct. 1949), pp. 656-715.

[Smi] Smith, J., "The design of Lucifer, a cryptographic device for data communications," IBM Corp. Res. Rep. RC 3326, April 1971.

[SNO] Smith, J., Notz, W., Osseck, P., "An Experimental Application of Cryptography to a Remotely Accessed Data System," IBM Corp. Res. Rep. RC 3508, August 1971.

[TCC] Telenet Communications Corporation, Host Interface Specifications, Washington, D.C., March 1975.

[Tur] Turn, R., "Privacy Transformations for Databank Systems," in 1973 NCC, AFIPS Conf. Proc. 42, PP. 589-601.

CS-TR Scanning Project
Document Control Form

Date : 12 / 11 / 95

Report # LCS-TR-162

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 128 (129 - IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☒ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form(2) ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP:	(1-122) UN#'ED TITLE PAGE, 2-7, UN#BLANK
	8-121
	(123-129) SCANCONTROL, COVER, DOD(2), TRGT'S(3)

Scanning Agent Signoff:

Date Received: 12/11/95 Date Scanned: 1/11/96 Date Returned: 1/18/96

Scanning Agent Signature: Michael N. Cook

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report 162	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: ENCRYPTION-BASED PROTECTION PROTOCOLS FOR INTERACTIVE USER-COMPUTER COMMUNICATION	5. TYPE OF REPORT & PERIOD COVERED: S.M. Thesis, April 1975-May 1976
	6. PERFORMING ORG. REPORT NUMBER: Technical Report 162
7. AUTHOR(s): Stephen Thomas Kent	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0661
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency, Department of Defense, 1400 Wilson Boulevard, Arlington, Virginia 22209	12. REPORT DATE: May 1976 13. NUMBER OF PAGES: 122
14. MONITORING AGENCY NAME & ADDRESS *(If different from Controlling Office)*: Office of Naval Research, Department of the Navy, Information Systems Program, Arlington, Virginia 22217	15. SECURITY CLASS. *(of this report)*: Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

encryption-based protection, computer security, communication security, block ciphers, NBS data encryption standard

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

see reverse side

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Block 20.

This thesis develops a complete set of protocols, which utilize a block cipher, e.g., the NBS data encryption standard, for protection interactive user-computer communication over physicall unsecured channels. The use of the block cipher protects against disclosure of message contents to an intruder, and the protocols provide for the detection of message stream modification and denial of message service by an intruder. The protocols include facilities for key distribution, two-way login authentication, resynchronization following channel disruption, and expedition of high priority messages. The thesis presents designs for modules to implement the protocols, both in a terminal and in a host computer system, and discusses the results of a test implementation of the modules on Multics.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**