U.S. Department of Commerce

Computer Science and Technology

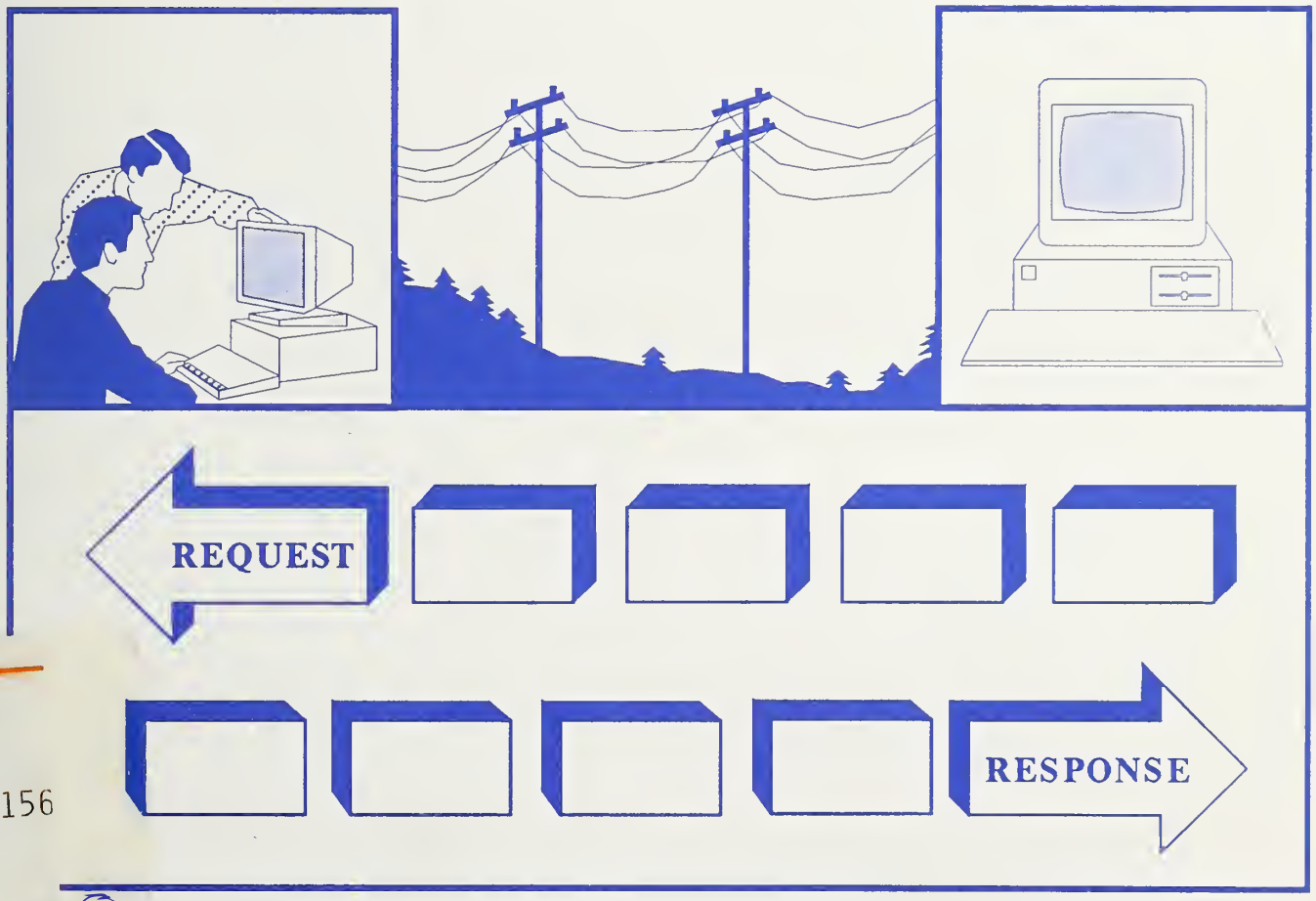
National Bureau of Standards

A11102 833027
NBS PUBLICATIONS
NAT'L INST OF STANDARDS & TECH R.I.C.
A11102833027
/Message authentication code (MAC) valid
QC100 .U57 NO.500-156 1988 V19 C.1 NBS-P

NBS Special Publication 500-156

Message Authentication Code (MAC) Validation System: Requirements and Procedures

Miles Smid, Elaine Barker, David Balenson, and Martha Haykin



QC
100
.U57
#500-156
1988
C.2

AUTHENTICATED
U.S. GOVERNMENT
INFORMATION
GPO

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research to assure international competitiveness and leadership of U.S. industry, science and technology. NBS work involves development and transfer of measurements, standards and related science and technology, in support of continually improving U.S. productivity, product quality and reliability, innovation and underlying science and engineering. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Institute for Materials Science and Engineering.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; provides calibration services; and manages the National Standard Reference Data System. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[3]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following divisions:

- Information Systems Engineering
- Systems and Software Technology
- Computer Security
- System and Network Architecture
- Advanced Systems

The Institute for Materials Science and Engineering

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-cutting scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Institute consists of the following Divisions:

- Ceramics
- Fracture and Deformation[3]
- Polymers
- Metallurgy
- Reactor Radiation

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.
[2]Some divisions within the center are located at Boulder, CO 80303.
[3]Located at Boulder, CO, with some elements at Gaithersburg, MD

Research Information Center
National Bureau of Standards
Gaithersburg, Maryland 20899
QC100
.U57
NO. 500-156
1988
C 2

Computer Science and Technology

NBS Special Publication 500-156

Message Authentication Code (MAC) Validation System: Requirements and Procedures

Miles Smid
Elaine Barker
David Balenson
Martha Haykin

Security Technology Group
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

May 1988

U.S. DEPARTMENT OF COMMERCE
C. William Verity, Secretary

National Bureau of Standards
Ernest Ambler, Director

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 88-600533
National Bureau of Standards Special Publication 500-156
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-156, 50 pages (May 1988)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1988

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington DC 20402

TABLE OF CONTENTS

page

ABSTRACT 1

1. INTRODUCTION 1

2. AUTHENTICATION STANDARDS 2
   - 2.1 Computer Data Authentication (FIPS PUB 113) 2
   - 2.2 ANSI X9.9 4
     - 2.2.1 The Authentication Process 4
     - 2.2.2 The Binary Authentication Option 5
     - 2.2.3 The Coded Character Set Authentication Options 5

3. DESIGN OF THE MAC VALIDATION SYSTEM 7
   - 3.1 Design Philosophy 7
   - 3.2 Basic Configuration and Operation 8

4. BASIC PROTOCOL 9
   - 4.1 Message Flow: Validate Suboptions 10
   - 4.2 Message Flow: Debug Suboptions 12
   - 4.3 Message Formats 14

5. BINARY OPTION 16
   - 5.1 Validate Suboption 16
   - 5.2 Debug Suboption 18

6. CODED CHARACTER SET OPTIONS 18
   - 6.1 Validate Suboptions 19
   - 6.2 Debug Suboptions 22

7. VALIDATION PROCEDURES 23
   - 7.1 Written Application to NBS 23
   - 7.2 Information Supplied to the Applicant by NBS 23
   - 7.3 Testing of the Message Authentication Device 23
   - 7.4 Completion of Testing 24
   - 7.5 Final Status of Validation Testing 24

8. CURRENT STATUS OF THE MVS 24

APPENDIX A: EXAMPLE MVS TEST SESSIONS 25
   - A.1 Binary Validate Suboption 25
     - A.1.1 Example of Successfully Completed Test Session 25
     - A.1.2 Example of Unsuccessfully Completed Test Session Because of Three Consecutive Repeats 27
     - A.1.3 Example of Test Passed After Two Incorrect Responses 29
     - A.1.4 Example of Test Failed Because of Three Incorrect Responses . . . 31
     - A.1.5 Example of Kill Message Sent by DUT 33

page
A.2 Binary Debug Suboption 35
A.2.1 Example of Test Session 35
A.3 Coded Character Set Entire Message/No Editing Validate Suboption 37
A.3.1 Example of Successfully Completed Test Session 37
A.4 Coded Character Set Entire Message/No Editing Debug Suboption 39
A.4.1 Example of Test Session 39
APPENDIX B: SAMPLE VALIDATION CERTIFICATE 41
REFERENCES 42

LIST OF FIGURES

page

Figure 1: The MAC Calculation 3

Figure 2: Sample Financial Message 7

Figure 3: Basic Configuration for Testing 8

Figure 4: Suboptions Provided by the MVS 10

Figure 5: Message Flow for the Validate Suboptions 11

Figure 6: Message Flow for the Debug Suboptions 13

Figure 7: Message Formats for the Binary Validate Suboption 17

Figure 8: Message Formats for the Binary Debug Suboption 19

Figure 9: Message Formats for the Coded Character Set Validate Suboptions . . . 20

Figure 10: Message Formats for the Coded Character Set Debug Suboptions 22

LIST OF ACRONYMS

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
DES	Data Encryption Standard
DPC	Device Protocol Converter
DUT	Device Under Test
EFT	Electronic Funds Transfer
FIPS PUB	Federal Information Processing Standards Publication
IDA	Identity of Authentication Key
MAC	Message Authentication Code
MID	Message Identifier
MVS	MAC Validation System
NBS	National Bureau of Standards
PC	Personal Computer
RBBS	Remote Bulletin Board System

ABSTRACT

The National Bureau of Standards Message Authentication Code (MAC) Validation System (MVS) tests message authentication devices for conformance to two data authentication standards: Federal Information Processing Standard Publication (FIPS PUB) 113, *Computer Data Authentication*, and American National Standards Institute (ANSI) X9.9-1986, *Financial Institution Message Authentication (Wholesale)*. The MVS is designed to perform automated testing on message authentication devices which are remote to NBS. This publication provides brief overviews of the two data authentication standards and introduces the basic design and configuration of the MVS. The requirements and administrative procedures to be followed by those seeking formal NBS validation of a message authentication device are presented. The requirements described include the specific protocols for communication between the message authentication device under test (DUT) and the MVS, the types of tests which the DUT must pass for formal NBS validation, and general instructions for accessing and interfacing to the MVS. An appendix with examples illustrating the MVS testing protocol is provided.

Key Words: Automated testing, computer security, cryptography, Data Encryption Standard (DES), data integrity, message authentication, validation.

1. INTRODUCTION

In automated data processing systems it is often impossible for humans to scan data to determine if it has been modified. Examination may be too time consuming for the vast quantities of data involved in modern data processing applications, or the data may have insufficient redundancy for error detection. Even if human scanning were possible, the data could have been modified in such a manner that it would be very difficult for the human to detect the modification. For example, "do" may have been changed to "do not" or "$1,000" may have been changed to "$10,000". Without additional information, the human scanner could easily accept the altered data as being authentic. This threat may still exist even when data encryption is used. It is therefore desirable to have an automated means of detecting both intentional and unintentional modifications to data. Ordinary error detecting codes are not adequate, because, if the algorithm for generating the code is known, an adversary can generate the correct code after modifying the data. Intentional modification is undetectable with such codes. However, a Message Authentication Code (MAC), a type of cryptographic checksum, can protect against both accidental and intentional, but unauthorized, data modification.

A MAC is initially calculated by applying a cryptographic algorithm and a secret value, called the key, to the data. The initial MAC is retained. The data is later verified by applying the cryptographic algorithm and the same secret key to the data to produce another MAC; this MAC is then compared to the initial MAC. If the two MACs are equal, then the data is considered authentic. Otherwise, an unauthorized modification is assumed. Any party trying to modify the data without knowing the key would not know how to calculate the appropriate MAC corresponding to the altered data.

American National Standards Institute (ANSI) X9.9-1986, *Financial Institution Message Authentication (Wholesale)* [1], makes use of the Data Encryption Standard (DES) cryptographic algorithm to calculate a MAC. The DES cryptographic algorithm was published by the National Bureau of Standards (NBS) as Federal Information Processing Standard Publication (FIPS PUB) 46, *Data Encryption Standard* [2], in 1977. The specific method for using the DES to calculate a MAC was published by NBS as FIPS PUB 113, *Computer Data Authentication* [3], in 1985.

In 1984, the U.S. Department of Treasury issued a policy directive, Directive 81-80,* *Electronic Funds and Securities Transfer Policy* [4], which established the requirement that the department's electronic funds transfer (EFT) messages be authenticated using ANSI X9.9 on all systems by 1988. In addition, the Department of Treasury instituted a program to certify message authentication devices and published *Criteria and Procedures for Testing, Evaluating, and Certifying Message Authentication Devices for Federal E.F.T. Use* [6]. To assist the Department of Treasury with its certification program, and to fulfull an ongoing goal of providing testing methodologies to support the development and implementation of Federal and voluntary commercial standards, NBS has developed the MAC Validation System (MVS) to test message authentication devices for conformance to FIPS PUB 113 and ANSI X9.9-1986. The development of the MAC Validation System was partially funded by the Security Programs Office of the Department of Treasury.

The following section summarizes FIPS PUB 113 and ANSI X9.9. Subsequent sections describe the design of the MVS (Section 3), the protocols used and tests performed by the MVS (Sections 4, 5, and 6), the procedures to be followed by those seeking formal NBS validation of a message authentication device (Section 7), and the current status of the MVS (Section 8). Examples illustrating the MVS testing protocol are provided in an appendix.

2. AUTHENTICATION STANDARDS

2.1 Computer Data Authentication (FIPS PUB 113)

FIPS PUB 113 defines an algorithm for calculating the MAC that is consistent with ANSI X9.9 and the Department of Treasury's Electronic Funds Transfer Policy. The MAC

* In 1986, Directive 81-80 was superceded by Directive 16-02 [5].

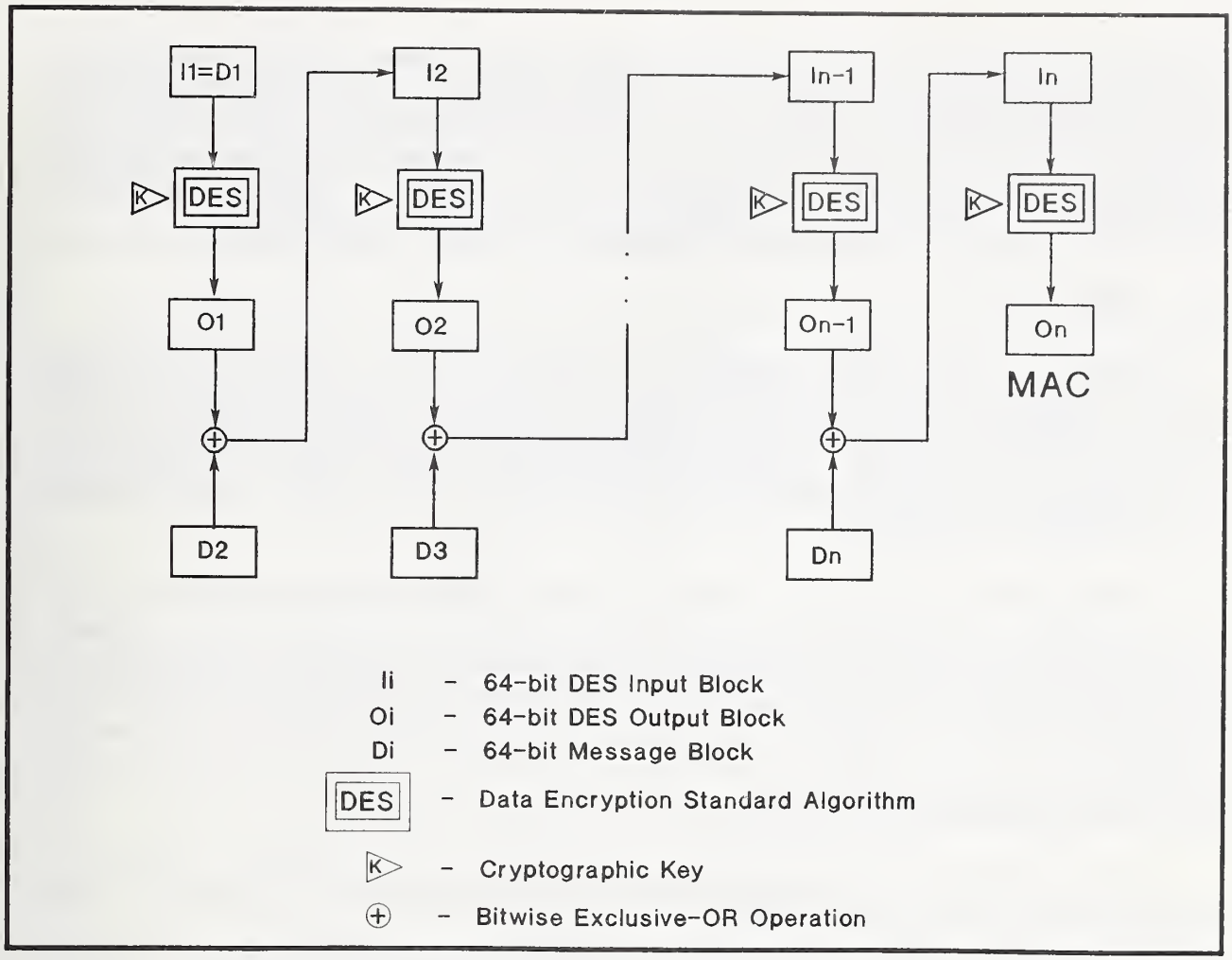


Figure 1. The MAC Calculation

calculation (see Figure 1) is based on the DES cryptographic algorithm which transforms 64-bit input blocks (Ii) to 64-bit output blocks (Oi) using a cryptographic key (K). The data to be authenticated is grouped into contiguous 64-bit blocks (D1,D2,...,Dn). If the number of data bits is not a multiple of 64, then the final data block (Dn) will be a partial block of data, left justified, with zeros appended to form a full 64-bit block. After the first data block (D1) is encrypted using DES, the output (O1) is exclusive-ORed to the second data block (D2) to form the next input (I2) to the DES. This process continues until the final data block (Dn) is exclusive-ORed to a DES output block (On-1), and the result is used as the last input (In) to the DES. The left-most M bits of the final DES output (On) are taken as the MAC, where $16 \leq M \leq 64$, and M is a multiple of 8; typically $M = 32$ bits.

Since the outputs of each DES transformation are chained to the inputs of the next DES transformation, the final MAC is a function of each bit of data and the secret cryptographic key. When the key is unknown, the alteration of a single bit of data will cause an unpredictable alteration of the MAC. Therefore, any intruder who intercepts authen-

ticated data and attempts to make an alteration will not know what the corresponding MAC for the altered data should be. The message authentication algorithm may be used to protect any data (transmitted or stored) which is exposed to possible modification between the initial generation of the MAC and the subsequent regeneration of a MAC. It does not detect errors which occur before the initial MAC is generated.

2.2 ANSI X9.9

Using the same message authentication algorithm as FIPS PUB 113, the ANSI X9.9 standard defines a uniform process for protecting wholesale financial messages with 32-bit, 48-bit, or 64-bit MACs. The process is independent of the transmission media, can be implemented in both automated and manual systems, and is usable by both large and small financial institutions.

Authentication of an entire message provides the best protection against unauthorized modification of data. However, in some applications, it is more desirable to authenticate only selected contents of a message. ANSI X9.9 defines procedures for authenticating an entire message or only certain essential selected contents of a message, called message elements.

2.2.1 The Authentication Process

Given a message to be transmitted from the originator to the recipient, the authentication process involves three steps.

(1) The originator of a message computes a MAC from the contents (or selected contents) of the message using a secret key and one of five authentication options provided by the standard. The five authentication options, which are described in more detail below, include one option for binary data and four options for ASCII messages. The choices of the authentication option and the method of exchanging keys are the responsibility of the originator and the recipient, and should be specified using procedures that are part of a bilateral agreement between the originator and the recipient.

(2) The originator transmits both the unencrypted message and its MAC to the recipient of the message.

(3) The recipient computes a MAC from the contents (or selected contents) of the received message (excluding the received MAC itself, and its delimiters, if any) using the same authentication option and key used by the originator. The computed MAC is then compared with the MAC received with the message.

The authentication process provides verification that the contents (or selected contents) of a message have not been accidentally or deliberately modified during transmission between the originator and the recipient. In addition, the identity of the originator

of a message is implicitly verified by the proper use of the correct secret key. By including the date and a unique message identifier in a message, the authentication process also provides verification of the uniqueness of a message (i.e., that the message is not a duplicate). The message identifier, which must be authenticated, is a value that does not repeat (typically a sequence number), such that there is not more than one message with the same message identifier that has the same date and uses the same key.

The authentication process alone does not guarantee absolute security. The protection provided applies only to the parts of a message that are actually authenticated. Other parts of a message may be subject to undetectable alterations. Physical, personnel, and procedural security controls are necessary for the secure implementation, use, and protection of the authentication process and devices. Keys must be protected in accordance with ANSI X9.17, *Financial Institution Key Management (Wholesale)* [7].

2.2.2 The Binary Authentication Option

The binary authentication option of ANSI X9.9 applies the authentication algorithm to the entire body of a message represented as a sequence of bits. The MAC is placed in the message in a predetermined location according to a bilateral agreement between the originator and the recipient. The binary authentication option provides compatibility with FIPS PUB 113 and is the recommended option for the authentication of bulk data.

2.2.3 The Coded Character Set Authentication Options

ANSI X9.9 provides four coded character set options which apply the authentication algorithm to either the entire contents or selected contents (i.e., message elements) of ASCII messages. The four authentication options differ in the parts of a message which are actually authenticated, i.e., which parts are input to the authentication algorithm in order to compute a MAC for the message.

(1) The *Entire Message with No Editing* Option applies the authentication algorithm to the entire message.

(2) The *Extracted Message Elements with No Editing* Option applies the authentication algorithm only to the message elements and any delimiters present.

The two non-editing options are recommended for the authentication of data whenever the transmission medium provides transparency.

(3) The *Entire Message with Editing* Option applies the authentication algorithm to the entire message, but first edits the contents according to several editing rules which substitute spaces for carriage returns and line feeds, convert all alphabetic characters to upper-case, delete all but certain acceptable characters, eliminate leading spaces, and compress sequences of consecutive spaces.

(4) The *Extracted Message Elements with Editing* Option applies the authentication algorithm only to the message elements and any delimiters present after editing the contents as in (3) above.

The two editing options are recommended for the authentication of ASCII data whenever the transmission medium (e.g.,a BAUDOT network, Telex) is not transparent to the character set being used (i.e., some of the characters are used for control or formatting purposes).

All characters included in the MAC computation must be represented as 8-bit ASCII characters with the leftmost bit set to zero and the right-most seven bits set as defined by ANSI X3.4, *Code for Information Interchange* [8]. If the message is represented by a different character set (e.g., EBCDIC), then the portions of the message to be authenticated must be transformed into ASCII before computing the MAC.

In all four coded character set options, an ASCII message contains fields (i.e., message elements), which are contiguous strings of characters designated for a specific purpose. Examples of fields that may appear in a financial message include the identities of the credit, debit, and beneficiary parties, the transaction value and currency types, the value date, and the identity of the key used for authentication (IDA). These fields may or may not appear in a message, but they must be authenticated if they do appear.

Fields that must always appear in a message and must also be authenticated include the date of message origination (Date) and a message identifier (MID). A MAC must also appear in a message, but is not included in the MAC computation. The formats of the IDA, Date, MID, and MAC fields are fixed by the standard, and each of these fields may appear only once in a message.

In order to locate and identify the fields in a message they must be either implicitly or explicitly delimited. A field is implicitly delimited if its placement in a message is either fixed or unambiguously specified by format rules. A field is explicitly delimited if its placement in a message is identified by a complementary pair of opening and closing explicit delimiters without any intervening delimiters. The standard establishes the following opening and closing explicit delimiters:

	OPEN	CLOSE
Date :	QD-	-DQ
IDA :	QK-	-KQ
MAC :	QM-	-MQ
MID :	QX-	-XQ
Text :	QT-	-TQ

Figure 2 depicts a sample financial message which uses these explicit delimiters. The use of implicit delimiters versus explicit delimiters and the formats of fields that are not fixed by the standard should be specified in the bilateral agreement between the originator and the recipient.

```
TO YOUR BANK

FROM OUR BANK

QD-80 07 14-DQ ///// 1056/  QX-127-XQ

QT-

TRNSFR USD $1234567,89 FRM ACCNT 48020-166
                 /////        TO ACCNT 40210-178

-TQ

KEEP ON QT EXPECT VISIT ON FRIDAY OF
NEW DIV VP ON PROJECT QT-QWERT-TQ BE

Careful

REGARDS

QUIRTO
QK-1357BANKATOBANKB-KQ
QM-D21F 3879-MQ
```

Figure 2. Sample Financial Message

3. DESIGN OF THE MAC VALIDATION SYSTEM

3.1 Design Philosophy

The MVS is designed to allow testing of message authentication devices from locations remote to the MVS. The automated MVS tests can be performed on a message authentication device via telephone lines. Shipping and set-up expenses are thus eliminated.

A message authentication device is *validated* for an option when it passes the appropriate MVS tests. Since there are an infinite number of possible tests and the number of tests that the device will be required to complete is limited, validation of an option does not guarantee 100% conformance with the corresponding option in the standards.

The intent of the validation process is to provide a rigorous conformance test which can be performed at a modest cost. NBS does not try to prevent a dishonest message authentication device vendor from purchasing a validated device and remotely validating the device as the vendor's own product. However, customers who wish to protect themselves against a dishonest vendor could require that the vendor revalidate the device in the customers' presence.

3.2 Basic Configuration and Operation

Figure 3 depicts the basic configuration for testing a device using the MVS. The MVS is implemented on a Personal Computer (PC) that is equipped with a DES encryption board. A public domain Remote Bulletin Board System (RBBS) and a 300/1200 baud modem are used to provide controlled dial-up access to the MVS.

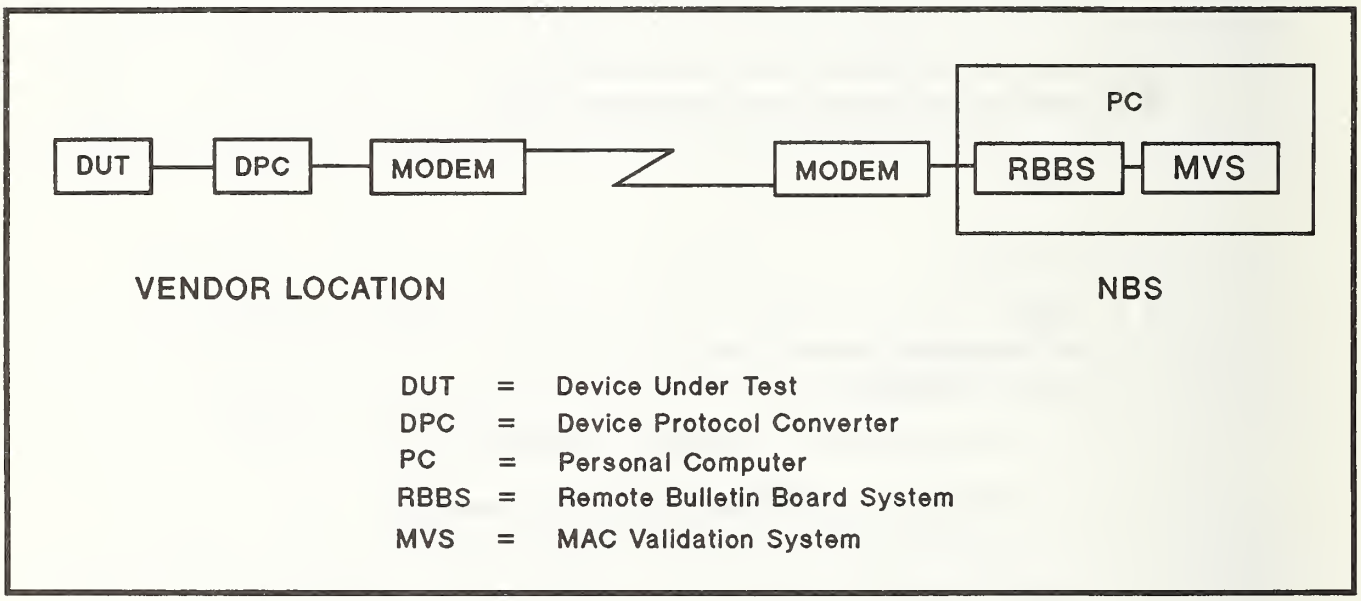


Figure 3. Basic Configuration for Testing

To conduct testing of a device under test (DUT) from a remote location, a user must have a 300/1200 baud modem and communications software (8 data bits, no parity, 1 stop bit, full duplex operation). In addition, the user must provide a device protocol converter (DPC) to interface the DUT with the MVS. The DPC is typically implemented in software (sometimes firmware) which is independent of the DUT and is used to convert the protocols of the DUT to and from the protocols of the MVS.

To access the MVS, the user must first log in to the RBBS with an assigned identifier and password. Various features of the RBBS will permit the user to change the assigned

password and to read bulletins which describe the operation of the system, list currently validated products, and provide any other important information. Other features of the RBBS, including personal mail and file uploading/downloading, are disabled.

Having successfully logged in to the RBBS, the user gains access to the MVS via the "WINDOW" feature of the RBBS, which allows the running of programs external to the RBBS. After selecting the MVS program, the user must log in to the MVS with an assigned identifier and password. The MVS will then display a menu, from which the user can choose to change his assigned password, run debug tests, run validation tests, or check testing status.

All RBBS, MVS, and testing activity is logged to permit monitoring of the systems to aid in resolving any discrepancies with expected test results.

4. BASIC PROTOCOL

The MVS has the capability for testing for conformance to any of the five ANSI X9.9 authentication options. The test for the binary authentication option of ANSI X9.9 will also test for conformance to FIPS PUB 113. ANSI X9.9 specifies the use of 32-bit, 48-bit, or 64-bit MACs, and FIPS PUB 113 specifies the use of MACs whose lengths are a multiple of 8 bits, ranging from 16 bits to 64 bits. However, in testing with the MVS, only 32-bit MACs are used.

For each of the five authentication options, the MVS has both a Validate and a Debug Suboption, as illustrated in Figure 4. The Validate Suboptions provide the actual tests which the DUT must pass in order to earn formal NBS validation. The Debug Suboptions are provided solely as debugging tools for the user and are not required for formal NBS validation.

The Debug and Validate Suboptions may be run on a trial basis in any order and as many times as desired. However, for formal NBS validation, the user shall select the *final validation* process and specify which of the five options are to be tested. Since the Binary Option is a prerequisite for the Coded Character Set Options, the Binary Validate Suboption will always be performed in the final validation process, along with any of the Coded Character Set Validate Suboptions which are specified. After the desired options have been specified, the corresponding Validate Suboptions will be performed automatically, without interruption, in the following sequence:

(1) Binary Validate Suboption
(2) Coded Character Set/Entire Message/No Editing Validate Suboption
(3) Coded Character Set/Extracted Message Elements/No Editing Validate Suboption
(4) Coded Character Set/Entire Message/Editing Validate Suboption
(5) Coded Character Set/Extracted Message Elements/Editing Validate Suboption

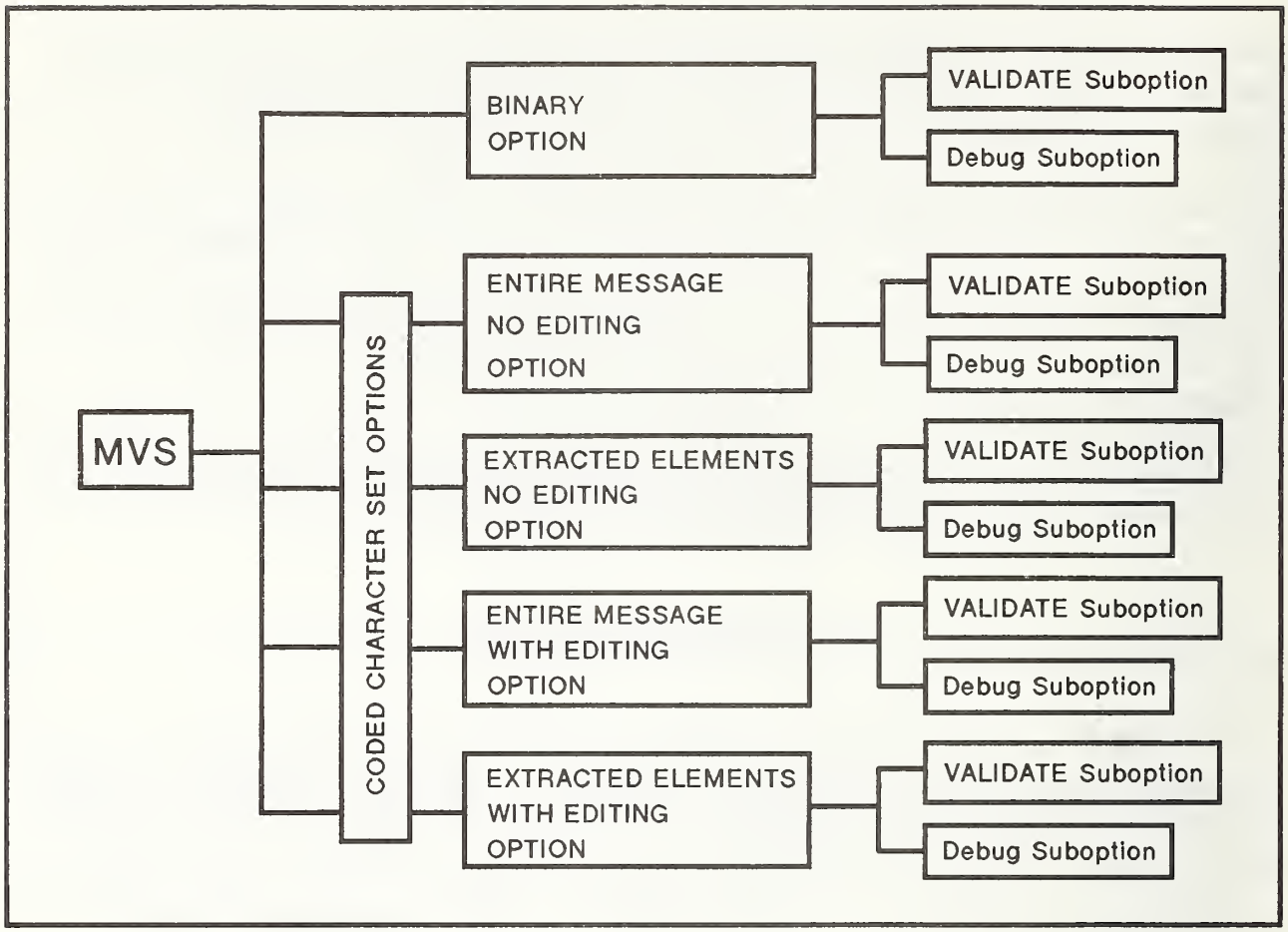


Figure 4. Suboptions Provided by the MVS

Examples of test sessions between the MVS and a DUT are shown for several suboptions in Appendix A.

4.1 Message Flow: Validate Suboptions

Figure 5 depicts the message flow for the Validate Suboptions. When a Validate Suboption is selected, the DUT shall send a *ready message* to the MVS to indicate that the DUT is ready to proceed with testing.

In each of the Coded Character Set Validate Suboptions, the MVS will send ten *key messages* to the DUT. Each key message will contain a key and corresponding identifier to be used during testing. After receiving the ten key messages, the DUT shall respond with a *continue message* to indicate that it is ready to continue with the testing. The key messages and continue message are not used in the Binary Validate Suboption.

Next, a series of validation tests will be performed. Each validation test will consist of a *request message*, a *response message*, and a *confirm message*. The request message is sent from the MVS to the DUT to request that the DUT perform one of two operations:

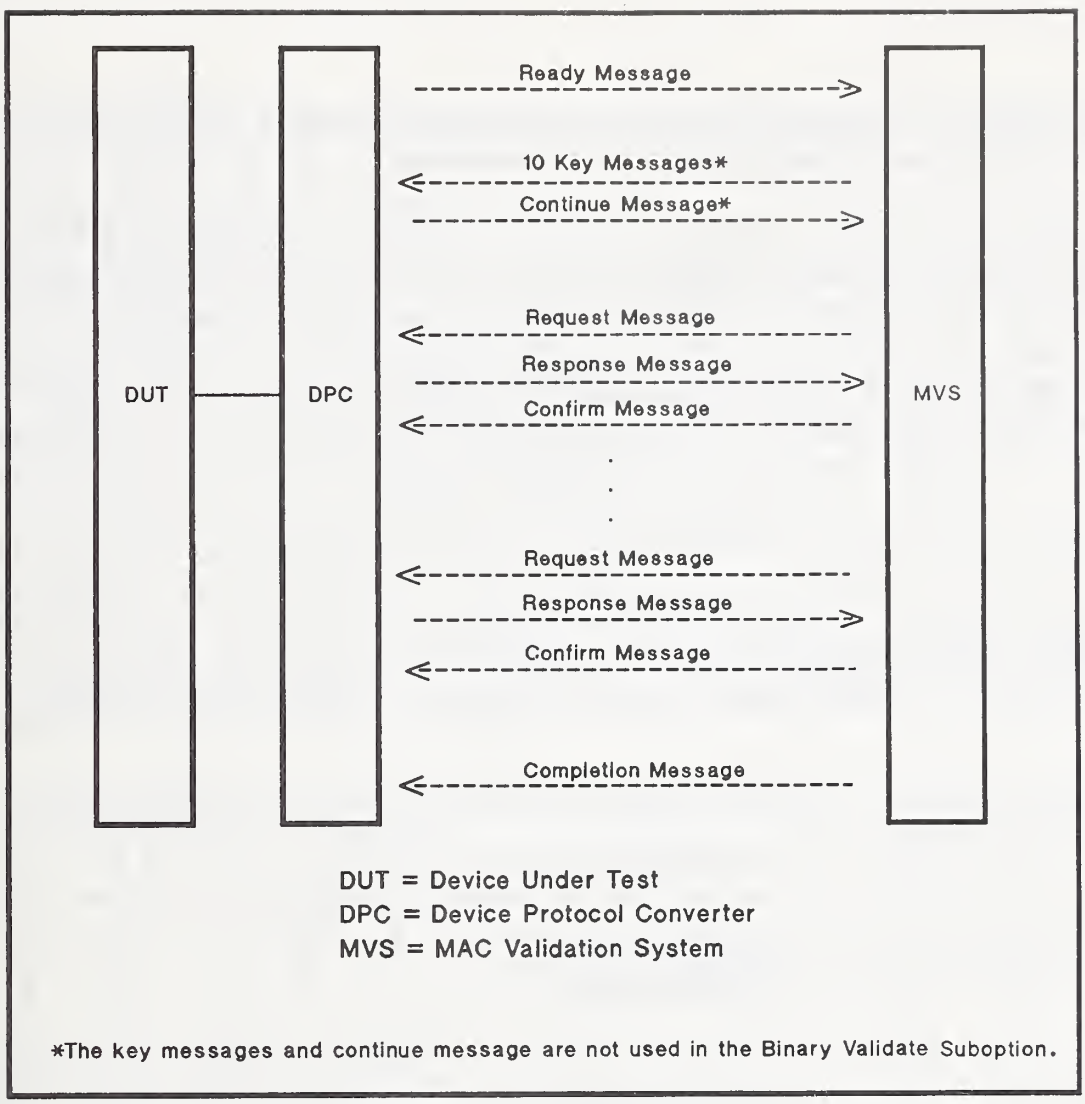


Figure 5. Message Flow for the Validate Suboptions

1) compute a MAC, or 2) compute a MAC and compare it to a MAC included in the request message. The response message is sent from the DUT to the MVS with the appropriate response for the operation performed. The confirm message is sent from the MVS to the DUT to indicate whether or not the DUT returned the correct response.

During validation testing the MVS will maintain a *retest count*, consisting of four decimal digits, which will be initialized to four zeros. The retest count is divided into two subfields. The first subfield, consisting of the left-most digit of the retest count, will indicate the final status of a Validate Suboption. The second subfield, consisting of the three right-most digits of the retest count, indicates the number of tests that are retested during a Validate Suboption.

Whenever the DUT receives a key message or request message from the MVS, the DUT may respond with a *repeat message*, in which case the MVS will resend the message. Normally, a repeat message is not counted against the DUT. However, if the MVS receives three consecutive repeat messages, then the first subfield of the retest count will be

incremented by 1. Testing will terminate (as described below), and the DUT will not receive credit for successfully completing the suboption.

Whenever the DUT provides an incorrect response to a request, the MVS will automatically repeat the same request in the next test. If the DUT provides the correct response within two retests using the same request, then the second subfield of the retest count will be incremented by 1 and testing will continue with the next test using a new request. However, if the DUT provides an incorrect response for three tests using the same request, then the first subfield of the retest count will be incremented by 2. Testing will continue but the DUT will not receive credit for successfully completing the suboption.

At any point during a Validate Suboption, either the MVS or the DUT may optionally terminate the testing by sending a *kill message* to the other device. If a kill message is sent or received by the MVS or if the connection between the MVS and DUT is disconnected for any reason, then the first subfield of the retest count will be incremented by 4. The DUT will not receive credit for successfully completing the suboption.

Upon completion or termination of a Validate Suboption, the MVS will send a *completion message* to the DUT containing the retest count which indicates the final status of the validation testing for that suboption. If the retest count is less than 0006 (fewer than 5 tests were retried, and the first subfield has not been incremented because of a kill message, three consecutive repeats received, or a test failure), then the suboption will be considered to be successfully completed.

In the final validation process, the Validate Suboptions which have been selected will be performed in succession, without interruption. After the DUT completes one Validate Suboption and the MVS sends a completion message indicating *successful* completion of that suboption, the next selected Validate Suboption will begin, and the MVS will wait for the DUT to send a ready message, as described above. In the event that the DUT fails a Validate Suboption during the final validation process, the MVS will send a completion message indicating that the suboption has been failed, and no further testing will be performed.

4.2 Message Flow: Debug Suboptions

The debug suboptions provide simulated testing situations in which the roles of the MVS and the DUT are reversed. During a debug suboption, the DUT may send request messages to the MVS, and the MVS will return correct response messages. The debug suboptions are provided solely to aid the user in debugging the DUT and are not required for formal NBS validation.

Figure 6 depicts the message flow for the Debug Suboptions. The message flow for the Debug Suboptions is similar to the message flow of the Validate Suboptions, except that:

(1) The MVS (not the DUT) will send a ready message after a Debug Suboption has been selected by the user. (The MVS will resend the ready message every three seconds until the DUT responds.)

(2) The DUT (not the MVS) will send request messages, and the MVS (not the DUT) will send response messages.

(3) The DUT may send repeat messages to request retransmission of response messages (as opposed to retransmission of request messages or key messages) sent by the MVS.

(4) There will be no confirm messages sent by either the MVS or the DUT.

(5) There will be no retest count.

(6) There will be no completion message sent by either the MVS or the DUT. The DUT may terminate the testing at any point during a Debug Suboption by sending a kill message to the MVS.

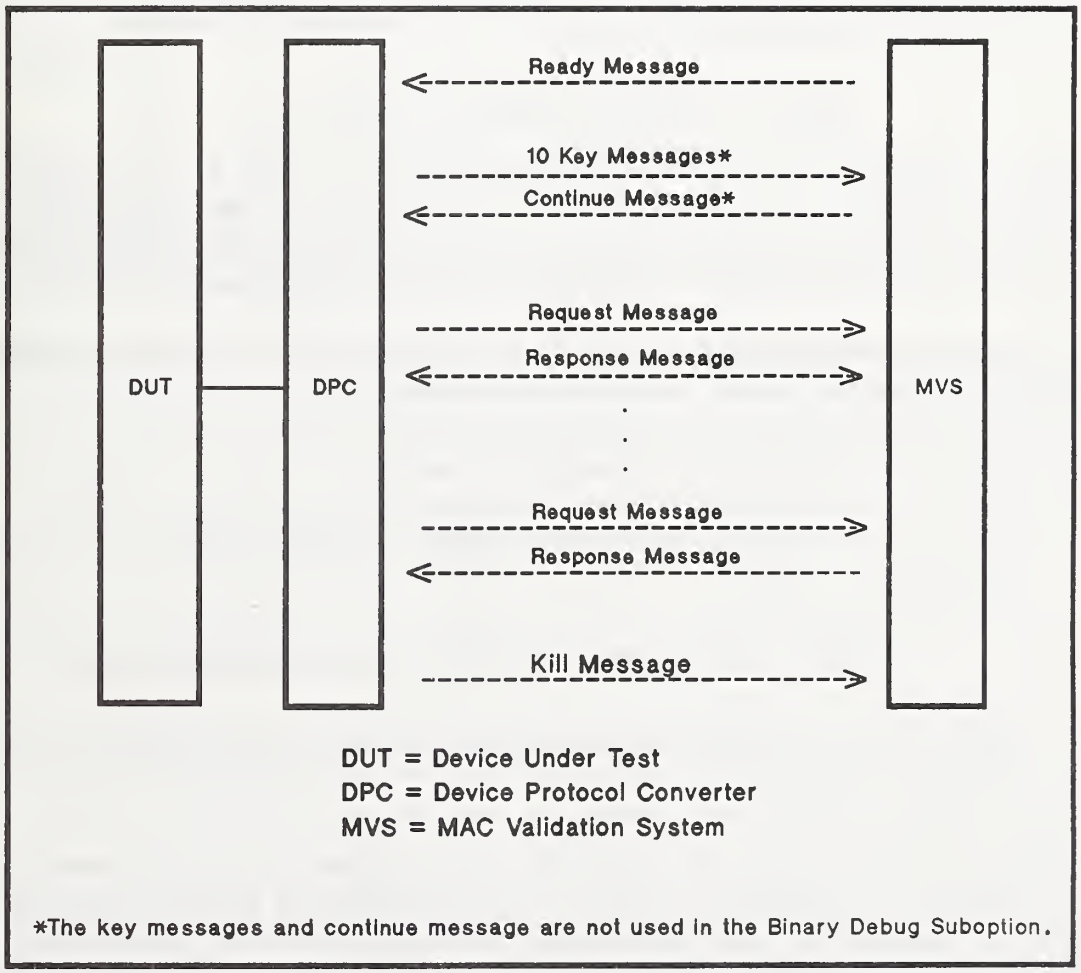


Figure 6. Message Flow for the Debug Suboptions

4.3 Message Formats

Each message sent between the MVS and the DUT shall consist of ASCII characters and shall be terminated with an ASCII ETX character (control-C). In addition, the ETX character will be used within some messages to separate keys, key identifiers, and data within the message. In this document the ETX character will be designated by "(ˆ*C*)". The ASCII hexadecimal characters which are used in the following message formats are the ASCII characters 0–9 and A–F, which represent 4-bit binary values. These ASCII hexadecimal characters must be translated into their equivalent 4-bit binary values and packed into 8-bit bytes prior to processing. The notation < 16 hex > in the following message formats represents exactly 16 ASCII hexadecimal characters that specify the value of a 64-bit cryptographic key with odd parity. The notation "ƀ" represents an ASCII space character, and the notation "h" represents an ASCII hexadecimal character.

The formats for the messages described in Sections 4.1 and 4.2 are as follows:

- o A *ready message* shall have the format READY(ˆ*C*) .

- o A *key message* shall have the format

  KEY= < 16 hex > (ˆ*C*)IDA= < 0–16 chrs > (ˆ*C*)

  where < 0–16 chrs > represents from 0 to 16 characters taken from a restricted set of ASCII characters* that specify the identifier of the corresponding cryptographic key. Key messages shall only be used in the Coded Character Set Options.

- o A *continue message* shall have the format CONTINUE(ˆ*C*). Continue messages shall only be used in the Coded Character Set Options.

- o A *request message* shall have one of four formats.

  (a) For the Binary Option, a request message for which a MAC is to be computed shall have the format

  KEY= < 16 hex > (ˆ*C*)DATA= < up to 1000 hex > (ˆ*C*)

  where < up to 1000 hex > represents up to 1000 ASCII hexadecimal characters that specify binary data.

---

* ANSI X9.9-1986 requires that "the IDA shall conform to the requirements for key identifiers of ANSI X9.17-1985." Thus, the format of the IDA field for ANSI X9.9-1986 shall be 0 to 16 characters from the following 42-character subset of the ASCII characters: the digits (0–9), the uppercase letters (A–Z), comma (,), solidus (/), dash or hyphen (-), asterisk (*), and open and close parentheses (() and ()).

(b) For the Binary Option, a request message for which a MAC is to be computed and compared with a MAC contained in the message shall have the format

KEY= < 16 hex > (^C)DATA=QM- < 9 chrs > -MQ < up to 985 hex > (^C)

where < 9 chrs > represents 9 ASCII characters that specify a MAC in the format hhhhþhhhh, and where < up to 985 hex > represents up to 985 ASCII hexadecimal characters that specify binary data.

(c) For the Coded Character Set Options, a request message for which a MAC is to be computed shall have the format

DATA= < up to 1000 chrs > (^C)

where < up to 1000 chrs > represents up to 1000 ASCII characters, excluding the ASCII NUL (control-@) and ETX (control-C) characters.

(d) For the Coded Character Set Options, a request message for which a MAC is to be computed and compared with a MAC contained in the message shall have the format

DATAM= < up to 1000 chrs > (^C)

where < up to 1000 chrs > represents up to 1000 ASCII characters as in request message format (c) above.

o A *response message* shall have one of four formats.

(a) In the Binary Debug Suboption, if a request message contains a protocol format error (i.e. does not correspond to one of the four request message formats above) or in the Coded Character Set Options, if the corresponding request message contains data that violates an ANSI X9.9 formatting rule (e.g., missing field, incorrect field format), then the error shall be indicated in a response message which shall have the format QM-ÞÞÞÞ*ÞÞÞÞ-MQ(^C).

(b) If a computed MAC is to be returned, then it shall be returned in a response message which shall have the format QM-hhhhþhhhh-MQ(^C).

(c) If a MAC is to be computed and compared against a MAC received in the request message, and the two MACs are equal, then the MAC shall be returned in a response message which shall have the format QM-hhhh+hhhh-MQ(^C).

(d) If a MAC is to be computed and compared against a MAC received in the request message, and the two MACs are not equal, then the received MAC shall be returned in a response message which shall have the format QM-hhhh*hhhh-MQ(^C).

- A *confirm message* shall have either the format PASS(^C) or the format FAIL(^C).
- A *kill message* shall have the format KILL(^C).
- A *repeat message* shall have the format REPEAT(^C).
- A *completion message* shall have either the format

  OPTION COMPLETED BUT FAILED, RETEST COUNT = < xyyy > (^C)

  which indicates that the corresponding Validate Suboption was failed and where < xyyy > represents 4 decimal digits which indicate the retest count, or the format

  OPTION COMPLETED SUCCESSFULLY(^C)

  which indicates that the corresponding Validate Suboption was completed successfully.

5. BINARY OPTION

In the Binary Option a request message shall consist of a key and data, formatted according to request message format (a) or (b) of Section 4.3. The given key shall be used to compute a MAC for the given data, which shall be converted to binary data before it is passed to the MAC algorithm.

A request message may or may not contain a MAC. If a request message does not contain a MAC, then request message format (a) is used, and up to 4000 bits of binary data which are represented by up to 1000 ASCII hexadecimal characters may be specified. The corresponding response message shall indicate the computed MAC using response message format (b).

If a request message contains a MAC, then request message format (b) is used, and the MAC field shall appear as the first 15 characters of data in the format QM-hhhh$\not$hhhh-MQ and shall not be included in the MAC computation. Up to 3940 bits of binary data which are represented by up to 1000 − 15 = 985 ASCII hexadecimal characters may be specified. The corresponding response message shall indicate the appropriate MAC using either response message format (c) or (d) depending on whether or not the computed MAC matches the MAC contained in the request message.

5.1 Validate Suboption

The Validate Suboption of the Binary Option permits the MVS to test the DUT by sending request messages to the DUT consisting of specially selected key and data values.

Ready Message:	READY(^C)
Request Message:	KEY=<16 hex>(^C)DATA=<up to 1000 hex>(^C) -or- KEY=<16 hex>(^C)DATA=QM-<9 chrs>-MQ<up to 985 hex>(^C)
Response Message:	QM-<9 chrs>-MQ(^C)
Confirm Message:	PASS(^C) -or- FAIL(^C)
Completion Message:	OPTION COMPLETED BUT FAILED, RETEST COUNT=<xyyy>(^C) -or- OPTION COMPLETED SUCCESSFULLY(^C)

(^C) = ASCII ETX Character

<hex> = ASCII Hexadecimal Character

<chrs> = ASCII Characters (some of which may be hexadecimal)

<xyyy> = 4 ASCII Decimal Digits

Figure 7. Message Formats for the Binary Validate Suboption

Figure 7 summarizes the messages and their formats for this suboption. The Binary Validate Suboption is a prerequisite to the Coded Character Set Validate Suboptions.

After this suboption is selected and the DUT sends a ready message to the MVS, the MVS will send a request message to the DUT consisting of a key and data. If the DUT recognizes a MVS protocol error in the request message, then the DUT may respond with a repeat message, in which case the MVS will resend the same request message. Otherwise, the DUT shall respond with a response message that specifies the appropriate MAC for the given request message. Upon receiving the response message, the MVS will send a confirm message to the DUT to indicate whether or not the MAC specified in the response message was correct. Upon completion of this suboption the MVS will report the status of the validation testing to the DUT by sending a completion message.

Tests involving the following types of key and data combinations are performed in the Binary Validate Suboption:

(1) Selected key and data combinations (without a MAC) which are related to those given in Appendix B of NBS Special Publication 500-20, *Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard* [9], except that the data consists of the given data with one to eight ASCII hexadecimal ones appended. 235 tests of this type are performed.

(2) Selected key and data combinations (without a MAC) which are related to those generated by the DES Maintenance Test as specified in NBS Special Publication 500-61, *Maintenance Testing for the Data Encryption Standard* [10], except that the data consists of the generated data with one to eight ASCII hexadecimal ones appended. 192 tests of this type are performed.

(3) Key and data combinations (without a MAC) which are randomly generated. At least 100 tests of this type are performed. Some of these combinations consist of data whose length is not a multiple of 64 bits so that the DUT must correctly pad the data in the MAC computation.

(4) Key and data combinations (with a MAC) which are randomly generated. At least 100 tests of this type are performed. Approximately half of the MACs are randomly chosen to be incorrect.

5.2 Debug Suboption

The Binary Debug Suboption permits the DUT to send the MVS request messages consisting of key and data combinations. The MVS will either return correct response messages for these request messages or indicate that the DUT's request message is not properly formatted (see Section 4.3). This suboption is provided for the user's benefit and is not required for formal NBS validation. Figure 8 summarizes the messages and their formats for the Binary Debug Suboption.

6. CODED CHARACTER SET OPTIONS

Each of the Coded Character Set Suboptions begins with an initial exchange of ten keys and corresponding key identifiers using key messages formatted as described in Section 4.3. If subsequent request messages do not contain an IDA field as part of the data, then the first of the ten transmitted keys is used to compute the MAC for the corresponding data.

In the Coded Character Set Options a request message uses either request message format (c) or (d) of Section 4.3, and contains a message formatted according to the rules of ANSI X9.9. In both cases, the request message may or may not contain a MAC. Explicit delimiters specified in ANSI X9.9 are used to locate and identify needed information such as the IDA and MAC fields. Implicit delimiters are not used.

Ready Message:	READY(^C)
Request Message:	KEY=<16 hex>(^C)DATA=<up to 1000 hex>(^C) -or- KEY=<16 hex>(^C)DATA=QM-<9 chrs>-MQ<up to 985 hex>(^C)
Response Message:	QM-<9 chrs>-MQ(^C)
Kill Message:	KILL(^C)

(^C) = ASCII ETX Character
<hex> = ASCII Hexadecimal Character
<chrs> = ASCII Characters (some of which may be hexadecimal)

Figure 8. Message Formats for the Binary Debug Suboption

If the request message uses request message format (c), then a MAC shall be computed. The computed MAC shall not be compared with a MAC which may appear in the request message. If a MAC does not appear in the request message, then this fact shall be ignored (i.e., the response message shall not indicate a format error.) If 1) a MAC does appear in the message and it is not properly delimited or formatted, or 2) any other ANSI X9.9 formatting rules are violated, response message format (a) shall be used. If the request message is formatted correctly according to ANSI X9.9, the response message shall indicate the computed MAC using response message format (b).

If the request message uses request message format (d), then a MAC shall be computed and compared with a MAC which must appear in the message. If the computed MAC compares successfully with the received MAC, response message format (c) shall be used. If the MACs do not compare, response message format (d) shall be used. If a MAC does not appear in the request message, or any other ANSI X9.9 formatting rules are violated, response message format (a) shall be used.

6.1 Validate Suboptions

The Coded Character Set Validate Suboptions permit the MVS to test the DUT by sending request messages to the DUT. Figure 9 summarizes the messages and their formats for these suboptions.

Ready Message:	READY(^C)
Key Message:	KEY=<16 hex>(^C)IDA=<0-16 chrs>(^C)
Continue Message:	CONTINUE(^C)
Request Message:	DATA=<up to 1000 chrs>(^C) -or- DATAM=<up to 1000 chrs>(^C)
Response Message:	QM-<9 chrs>-MQ(^C)
Confirm Message:	PASS(^C) -or- FAIL(^C)
Completion Message:	OPTION COMPLETED BUT FAILED, RETEST COUNT=<xyyy>(^C) -or- OPTION COMPLETED SUCCESSFULLY(^C)

(^C) = ASCII ETX Character
<hex> = ASCII Hexadecimal Character
<chrs> = ASCII Characters (some of which may be hexadecimal)
<xyyy> = 4 ASCII Decimal Digits

Figure 9. Message Formats for the Coded Character Set Validate Suboptions

After one of the suboptions is selected and the DUT sends a ready message to the MVS, the MVS will send ten key messages to the DUT. When the DUT has received the ten key messages, it shall send a continue message to the MVS. The MVS will then send a request message to the DUT consisting of an ASCII message with explicit delimiters. If the DUT recognizes a MVS protocol error in the request message, then the DUT may respond with a repeat message, in which case the MVS will resend the same request message. Otherwise, the DUT shall respond with the appropriate response message for the given request message. Upon receiving the response message, the MVS will send a confirm message to the DUT to indicate whether or not the response message was correct.

Upon completion of this suboption the MVS will report the status of the validation testing to the DUT by sending a completion message.

In each of the Coded Character Set Validate Suboptions the following will be tested:

(1) Messages requiring only the computation of a MAC. Approximately three fourths of the messages will require that a MAC be computed, but not compared to a MAC which may or may not appear in the message. These messages will use request message format (c).

(2) Messages requiring computation of a MAC and comparison with the MAC contained in the message. Approximately one fourth of the messages will contain MACs that shall be compared with the computed MAC. These messages will use request message format (d). Approximately one half of these messages will contain incorrect MACs.

(3) The example message given in Appendix B of ANSI X9.9.

(4) Messages that shall be modified by deleting characters, inserting characters, modifying characters, and transposing characters.

(5) Messages of varying lengths, and hence, requiring varying amounts of padding during the MAC computation.

(6) Messages with the entire ASCII character set (excluding the NULL and ETX characters), and messages with the parity bits set for each character.

(7) Processing of explicit delimiters, including messages that contain incomplete explicit delimiters, lowercase explicit delimiters, unexpected opening explicit delimiters, missing closing explicit delimiters, unexpected closing explicit delimiters, mismatched closing explicit delimiters, and pairs of explicit delimiters that are transposed.

(8) Processing of fixed message element formats, including the Date, IDA, MID, and MAC fields.

(9) Processing of required message elements, including the Date, MID, and MAC fields, and processing of optional message elements, including the IDA and Text fields.

(10) Handling of message elements that should appear only once in a message, including the Date, IDA, MID, and MAC fields, and handling of message elements that may appear more than once in a message, including the Text field.

(11) Messages with the message elements arranged in different orders.

(12) Application of the editing rules in the proper order, including processing of messages with carriage returns and line feeds, processing of lowercase alphabetics, processing of characters other than the uppercase letters, digits, space, comma, period, dash, solidus, asterisk, and open and close parenthesis, processing of messages (or message

elements) containing leading spaces, and processing of messages containing sequences of consecutive spaces (internal and trailing).

6.2 Debug Suboptions

The Coded Character Set Debug Suboptions permit the DUT to send the MVS request messages. These suboptions are provided for the user's benefit and are not required for formal NBS validation. Figure 10 summarizes the messages and their formats for these suboptions.

Ready Message:	READY(^C)
Key Message:	KEY=<16 hex>(^C)IDA=<0-16 chrs>(^C)
Continue Message:	CONTINUE(^C)
Request Message:	DATA=<up to 1000 chrs>(^C) -or- DATAM=<up to 1000 chrs>(^C)
Response Message:	QM-<9 chrs>-MQ(^C)
Kill Message:	KILL(^C)

(^C) = ASCII ETX Character
<hex> = ASCII Hexadecimal Character
<chrs> = ASCII Characters (some of which may be hexadecimal)
<xyyy> = 4 ASCII Decimal Digits

Figure 10. Message Formats for the Coded Character Set Debug Suboptions

Once a Coded Character Set Suboption is selected and a ready message is sent from the MVS to the DUT. the DUT shall send ten key messages to the MVS. After the MVS has received the keys and is ready for testing it will send a continue message to the DUT. The DUT shall then send a request message consisting of data, using either request message format (c) or (d) of Section 4.3. The data may contain explicit delimiters as specified in ANSI X9.9. The MVS will respond with the appropriate response message. The DUT may optionally terminate testing by sending a kill message to the MVS. In the event that the DUT sends a request message which is not properly formatted (i.e., does

not conform to request message format (c) or (d)), the MVS will terminate testing by sending a kill message to the DUT.

7. VALIDATION PROCEDURES

This section outlines the administrative procedures to be followed by a vendor or other organization seeking formal NBS validation of a message authentication device.

7.1 Written Application to NBS

A written application containing the following information shall be sent to NBS.

- o The name, address, and telephone number of the applicant and of the person within the applicant's organization who will be responsible for the validation testing.
- o A brief functional description of the message authentication device to be tested, including the name, model number, serial number, and version number of each major hardware/software component of the device.
- o A list indicating the authentication options against which the device is to be validated.

Written applications for message authentication device validation should be addressed to

Manager, Security Technology Group
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

7.2 Information Supplied to the Applicant by NBS

Upon receipt of a written application, NBS will provide the applicant with the telephone number of the RBBS/MVS, the identifiers and passwords required to access the RBBS/MVS, and specific instructions concerning use of the RBBS/MVS.

7.3 Testing of the Message Authentication Device

Validation testing of the message authentication device shall be conducted using the RBBS/MVS via remote dialup. If the device fails the validation, then either 1) the deficiency within the device may be corrected and testing repeated, 2) the options against which the device is to be validated may be changed and testing repeated, or 3) the device may be withdrawn from the validation process.

7.4 Completion of Testing

If and when the message authentication device successfully completes the desired set of options, or the device is withdrawn from the validation process, then the applicant shall contact NBS. Validation of the a message authentication device is subject to written confirmation by NBS.

7.5 Final Status of Validation Testing

NBS will report the final status of the validation testing to the applicant and issue a *validation certificate* indicating the options that were successfully completed. (See Appendix B for a sample Validation Certificate.) Note that formal NBS validation of the a message authentication device using the MVS is not equivalent to Department of Treasury certification. In addition to the NBS validation testing, Department of Treasury certification also requires that the device pass all of the other certification criteria specified in *Criteria and Procedures for Testing, Evaluating, and Certifying Message Authentication Devices for Federal E.F.T. Use* [6].

8. CURRENT STATUS OF THE MVS

The NBS MAC Validation System has been fully functional since May, 1986. Vendors, banks, and other organizations both in the United States and in Europe have used the MVS to test message authentication devices. NBS has awarded several message authentication device validation certificates, some for devices which conformed to all five authentication options. The MVS has detected flaws in some device implementations, which were later corrected. The time required for testing, using a 1200 baud modem (within the U.S.), has averaged 20-30 minutes for the Binary Option and 15-20 minutes for each of the Coded Character Set Options. A list including the names of vendors and organizations who have validated message authentication devices and the options against which the devices have been validated can be obtained by writing to the manager of the Security Technology Group at the address listed in Section 7.1.

APPENDIX A: EXAMPLE MVS TEST SESSIONS

Example test sessions between the MVS and a DUT are shown for several suboptions. The complete protocol, including the message flow and the message contents, is depicted, e.g.,

```
                                READY (^C)
---------------------- READY MESSAGE ---------------------->
```

For each message, the type of message and its direction is shown by a dashed arrow. The message contents are depicted above the dashed arrow. All non-printable control characters are displayed preceeded by an up arrow (carat) and shown within parentheses. For example, the character control-C is shown as (^*C*).

A.1 Binary Validate Suboption

A.1.1 Example of Successfully Completed Test Session

This example depicts a successfully completed test session. The test session starts with a READY MESSAGE and proceeds with a series of n tests (i.e., *test* 1, *test* 2, ..., *test* n). Each test consists of a REQUEST MESSAGE, a RESPONSE MESSAGE, and a CONFIRM MESSAGE. Notice that *test* 1, for example, includes a REQUEST MESSAGE which contains data for which a MAC must be computed, but NOT compared to a MAC which may appear in the message (in this example, there is no MAC in the data to be compared against). However, *test* 2, for example, includes a REQUEST MESSAGE which contains data for which a MAC must be computed and compared to a MAC which appears in the data (the MAC appears at the beginning of the data and is delimited by "QM-" and "-MQ"). In this example, the DUT passes every test, sends no REPEAT MESSAGES, and does not send a KILL MESSAGE. Thus, the test session ends with a COMPLETION MESSAGE which indicates successful completion of the suboption.

```
DUT                                                                  MVS

                          READY(^C)
---------------------- READY MESSAGE ---------------------->

   KEY=1C587F1C13924FEF(^C)DATA=F32927EAC4339C6E111(^C)
<-------------------- REQUEST MESSAGE ---------------------
                                                                  )
                     QM-D7E5 A7D6-MQ(^C)                          }
------------------- RESPONSE MESSAGE -------------------->        } TEST 1
                                                                  }
                          PASS(^C)                                }
<-------------------- CONFIRM MESSAGE ---------------------      )

  KEY=1F2F1C6BCEE5EA07(^C)DATA=QM-AEB1 791F-MQ4349319(^C)
<-------------------- REQUEST MESSAGE ---------------------
                                                                  )
                     QM-AEB1*791F-MQ(^C)                          }
------------------- RESPONSE MESSAGE -------------------->        } TEST 2
                                                                  }
                          PASS(^C)                                }
<-------------------- CONFIRM MESSAGE ---------------------      )

                              .
                              .
                              .

   KEY=E9323864518302D0(^C)DATA=QM-9F29 C755-MQ968C44BD
   6EC528635F201BC3557E32AFAD2D61C15FA255CE8B4416258F80
                  C2B2EF608275E4(^C)
<-------------------- REQUEST MESSAGE ---------------------
                                                                  )
                    QM-9F29+C755-MQ(^C)                           }
------------------- RESPONSE MESSAGE -------------------->        } TEST n
                                                                  }
                          PASS(^C)                                }
<-------------------- CONFIRM MESSAGE ---------------------      )

              OPTION COMPLETED SUCCESSFULLY(^C)
<----------------- COMPLETION MESSAGE --------------------
```

Example A.1.1

A.1.2 Example of Unsuccessfully Completed Test Session Because of Three Consecutive Repeats

This example depicts an unsuccessfully completed test session. The test session starts with a READY MESSAGE and proceeds normally with a series of tests. However, after several tests, the DUT encounters a test in which, for some reason, the DUT responds to the REQUEST MESSAGE with a REPEAT MESSAGE (to request retransmission of the REQUEST MESSAGE) three (3) consecutive times. The MVS retransmits the REQUEST MESSAGE for each REPEAT MESSAGE it receives, and, after repeating the REQUEST MESSAGE the third time, the MVS terminates the test session by sending a KILL MESSAGE. The test session ends with a COMPLETION MESSAGE which indicates that the suboption was unsuccessfully completed. The retest count of 1000 indicates that the test session was terminated by the MVS after receiving three (3) consecutive REPEAT MESSAGES.

```
DUT                                                                  MVS

                                  READY(^C)
---------------------- READY MESSAGE ---------------------->

   KEY=100791159819010B(^C)DATA=0000000000000000111(^C)
<-------------------- REQUEST MESSAGE ---------------------
                                                                  )
                     QM-9CAC F7C9-MQ(^C)                          }
-------------------- RESPONSE MESSAGE --------------------->      } TEST 1
                                                                  }
                                  PASS(^C)                        }
<-------------------- CONFIRM MESSAGE ---------------------      )

                                  .
                                  .
                                  .

   KEY=07A7137045DA2A16(^C)DATA=3BDD1190493728021(^C)
<-------------------- REQUEST MESSAGE ---------------------
                                                                  )
                                 REPEAT(^C)                       }
-------------------- REPEAT MESSAGE ---------------------->       }
                                  .                               }
   KEY=07A7137045DA2A16(^C)DATA=3BDD1190493728021(^C)             }
<-------------------- REQUEST MESSAGE ---------------------       }
                                                                  }
                                 REPEAT(^C)                       }
-------------------- REPEAT MESSAGE ---------------------->       }  TEST i
                                                                  }
   KEY=07A7137045DA2A16(^C)DATA=3BDD1190493728021(^C)             }
<-------------------- REQUEST MESSAGE ---------------------       }
                                                                  }
                                 REPEAT(^C)                       }
-------------------- REPEAT MESSAGE ---------------------->       }
                                                                  }
   KEY=07A7137045DA2A16(^C)DATA=3BDD1190493728021(^C)             }
<-------------------- REQUEST MESSAGE ---------------------       )

                                  KILL(^C)
<--------------------- KILL MESSAGE -----------------------

    OPTION COMPLETED BUT FAILED, RETEST COUNT=1000(^C)
<----------------- COMPLETION MESSAGE --------------------
```

Example A.1.2

A.1.3 Example of Test Passed After Two Incorrect Responses

This example depicts a successfully completed test session in which a particular test requires three attempts before it is passed. The test session starts with a READY MESSAGE and proceeds normally with a series of tests. However, after several tests, the DUT encounters a test to which, for some reason, the DUT provides an incorrect response. Since the response is incorrect, the test is automatically retried; that is, the MVS gives the DUT another attempt at the same test. Again the DUT provides an incorrect response, and again, the test is automatically retried. On the third attempt at the test, the DUT provides the correct response, and hence, passes the test. Testing continues with the next test. In this example, the DUT passes every test. no additional tests are retried, and the DUT does not send a KILL MESSAGE. Thus, the test session ends with a COMPLETION MESSAGE which indicates successful completion of the suboption. Note that, since a single test was retried, the final retest count (which is not included in the COMPLETION MESSAGE) would be 0001.

```
DUT                                                    MVS

                          READY(^C)
--------------------- READY MESSAGE --------------------->

   KEY=1C587F1C13924FEF(^C)DATA=892479ED40A35BB211(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                          )
                     QM-E0AE B421-MQ(^C)                  }
------------------- RESPONSE MESSAGE --------------------> } TEST 1
                                                          }
                          PASS(^C)                        }
<------------------- CONFIRM MESSAGE ---------------------
                                                          )
                              .
                              .
                              .
   KEY=2946A1C1AEA86780(^C)DATA=4DED55C0FACA30801111(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                          )
                     QM-E719 8C7B-MQ(^C)                  }
------------------- RESPONSE MESSAGE --------------------> } TEST i1
                                                          }
                          FAIL(^C)                        }
<------------------- CONFIRM MESSAGE ---------------------
                                                          )
   KEY=2946A1C1AEA86780(^C)DATA=4DED55C0FACA30801111(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                          )
                     QM-F82A 9D8C-MQ(^C)                  }
------------------- RESPONSE MESSAGE --------------------> } TEST i2
                                                          }
                          FAIL(^C)                        }
<------------------- CONFIRM MESSAGE ---------------------
                                                          )
   KEY=2946A1C1AEA86780(^C)DATA=4DED55C0FACA30801111(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                          )
                     QM-D608 7B6A-MQ(^C)                  }
------------------- RESPONSE MESSAGE --------------------> } TEST i3
                                                          }
                          PASS(^C)                        }
<------------------- CONFIRM MESSAGE ---------------------
                                                          )

   KEY=A2F8625E9E737679(^C)DATA=QM-8F1B 7655-MQ311CFA098
   0BAC345F913524FB4F6769960D894E4B383633E44C48C53999DEF
                        0E794E(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                          )
                     QM-8F1B*7655-MQ(^C)                  }
------------------- RESPONSE MESSAGE --------------------> } TEST i+1
                                                          }
                          PASS(^C)                        }
<------------------- CONFIRM MESSAGE ---------------------
                                                          )
                              .
                              .
                              .
             OPTION COMPLETED SUCCESSFULLY(^C)
<----------------- COMPLETION MESSAGE --------------------
```

Example A.1.3

A.1.4 Example of Test Failed Because of Three Incorrect Responses

This example depicts an unsuccessfully completed test session in which a particular test is failed. The test session starts with a READY MESSAGE and proceeds normally with a series of tests. However, after several tests, the DUT encounters a test to which, for some reason, the DUT provides an incorrect response. Since the response is incorrect, the test is automatically retried; that is, the MVS gives the DUT another attempt at the same test. Again the DUT provides an incorrect response, and again, the test is automatically retried. On the third attempt at the test, the DUT once again, provides an incorrect response, and the test is considered failed. Testing continues with the next test. In this example, no additional tests are retried or failed, and the DUT does not send a KILL MESSAGE. Thus, the test session ends with a COMPLETION MESSAGE which indicates that the suboption was unsuccessfully completed. The retest count of 2001 indicates that the test session was unsuccessfully completed due to a failed test and that a single test was retried at least one time.

```
DUT                                                    MVS

                          READY(^C)
--------------------- READY MESSAGE --------------------->

   KEY=1C587F1C13924FEF(^C)DATA=892479ED40A35BB211(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                            )
                     QM-E0AE B421-MQ(^C)                    }
------------------- RESPONSE MESSAGE -------------------->  } TEST 1
                                                            }
                          PASS(^C)                          }
<------------------- CONFIRM MESSAGE ---------------------  )

                              .
                              .
                              .

   KEY=2946A1C1AEA86780(^C)DATA=4DED55C0FACA30801111(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                            )
                     QM-E719 8C7B-MQ(^C)                    }
------------------- RESPONSE MESSAGE -------------------->  } TEST i1
                                                            }
                          FAIL(^C)                          }
<------------------- CONFIRM MESSAGE ---------------------  )

   KEY=2946A1C1AEA86780(^C)DATA=4DED55C0FACA30801111(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                            )
                     QM-F82A 9D8C-MQ(^C)                    }
------------------- RESPONSE MESSAGE -------------------->  } TEST i2
                                                            }
                          FAIL(^C)                          }
<------------------- CONFIRM MESSAGE ---------------------  )

   KEY=2946A1C1AEA86780(^C)DATA=4DED55C0FACA30801111(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                            )
                     QM-F82A 9D8C-MQ(^C)                    }
------------------- RESPONSE MESSAGE -------------------->  } TEST i3
                                                            }
                          FAIL(^C)                          }
<------------------- CONFIRM MESSAGE ---------------------  )

   KEY=A2F8625E9E737679(^C)DATA=QM-8F1B 7655-MQ311CFA098
   0BAC345F913524FB4F6769960D894E4B383633E44C48C53999DEF
                        0E794E(^C)
<------------------- REQUEST MESSAGE ---------------------
                                                            )
                     QM-8F1B*7655-MQ(^C)                    }
------------------- RESPONSE MESSAGE -------------------->  } TEST i+1
                                                            }
                          PASS(^C)                          }
<------------------- CONFIRM MESSAGE ---------------------  )

                              .
                              .
                              .

    OPTION COMPLETED BUT FAILED, RETEST COUNT=2001(^C)
<----------------- COMPLETION MESSAGE --------------------
```

Example A.1.4

A.1.5 Example of Kill Message Sent by DUT

This example depicts an unsuccessfully completed test session in which the DUT terminates the test session by sending a KILL MESSAGE. The test session starts with a READY MESSAGE and proceeds normally with a series of tests. However, after several tests, the DUT encounters a test to which, for some reason, the DUT provides an incorrect response. Since the response is incorrect, the test is automatically retried, that is, the MVS gives the DUT another attempt at the same test. However, the DUT decides, for some reason (e.g., it knows why it provided an incorrect response and wants to correct the problem rather than continue with testing), to terminate the test session by sending a KILL MESSAGE to the MVS. The test session ends with a COMPLETION MESSAGE which indicates that the suboption was unsuccessfully completed. The retest count of 4000 indicates that the test session was unsuccessfully completed due to a KILL message sent by the DUT.

```
DUT                                                    MVS

                          READY(^C)
---------------------- READY MESSAGE ---------------------->

         KEY=201A434545D51901(^C)DATA=22F4904(^C)
<-------------------- REQUEST MESSAGE ---------------------
                     QM-588D 42A7-MQ(^C)
-------------------- RESPONSE MESSAGE --------------------->   TEST 1
                          PASS(^C)
<-------------------- CONFIRM MESSAGE ---------------------

                              .
                              .
                              .

   KEY=1C587F1C13924FEF(^C)DATA=F32927EAC4339C6E111(^C)
<-------------------- REQUEST MESSAGE ---------------------
                    QM-E8F6 B8E7-MQ(^C)
-------------------- RESPONSE MESSAGE --------------------->   TEST i
                          FAIL(^C)
<-------------------- CONFIRM MESSAGE ---------------------

   KEY=1C587F1C13924FEF(^C)DATA=F32927EAC4339C6E111(^C)
<-------------------- REQUEST MESSAGE ---------------------
                          KILL(^C)
---------------------- KILL MESSAGE ----------------------->

    OPTION COMPLETED BUT FAILED, RETEST COUNT=4000(^C)
<----------------- COMPLETION MESSAGE --------------------
```

Example A.1.5

A.2 Binary Debug Suboption

A.2.1 Example of Test Session

This example depicts a complete test session. The test session starts with a READY MESSAGE and proceeds with a series of n tests (i.e., *test* 1, *test* 2, ..., *test* n). Each test consists of a REQUEST MESSAGE and a RESPONSE MESSAGE. There is no CONFIRM MESSAGE. Notice that *test* 1, for example, includes a REQUEST MESSAGE which contains data for which a MAC must be computed, but NOT compared to a MAC which may appear in the message (in this example, there is no MAC in the data to be compared against). However, *test* 2, for example, includes a REQUEST MESSAGE which contains an error in the testing protocol (ie., "DAXX:" instead of "DATA=") so the MVS provides a RESPONSE MESSAGE which indicates that it found an error in the REQUEST MESSAGE. The test session ends when the DUT sends a KILL MESSAGE to the MVS.

```
DUT                                                    MVS

                          READY(^C)
<-------------------- READY MESSAGE ----------------------

  KEY=9845F14929EA862A(^C)DATA=59236D8E9239BC60111(^C)
-------------------- REQUEST MESSAGE -------------------->
                                                                TEST 1
                    QM-7BDA 3ED0-MQ(^C)
<------------------ RESPONSE MESSAGE --------------------

  KEY=A946025A3AFB973B(^C)DAXX:6A347E9FA24ACD71111(^C)
-------------------- REQUEST MESSAGE -------------------->
                                                                TEST 2
                    QM-    *    -MQ(^C)
<------------------ RESPONSE MESSAGE --------------------

  KEY=1C587F1C13924FEF(^C)DATA=2FE31E8B6FBC5E39111(^C)
-------------------- REQUEST MESSAGE -------------------->
                                                                TEST 3
                    QM-F23D D241-MQ(^C)
<------------------ RESPONSE MESSAGE --------------------

                              .
                              .
                              .

  KEY=7529D32C58D02F1A(^C)DATA=0351A6C1A9552E8211111(^C)
-------------------- REQUEST MESSAGE -------------------->
                                                                TEST n
                    QM-B987 A362-MQ(^C)
<------------------ RESPONSE MESSAGE --------------------

                          KILL(^C)
--------------------- KILL MESSAGE ---------------------->
```

Example A.2.1

A.3 Coded Character Set Entire Message/No Editing Validate Suboption

A.3.1 Example of Successfully Completed Test Session

This example depicts a successfully completed test session for the Coded Character Set Entire Message/No Editing Validate Suboption. This example is applicable to all four Coded Character Set Validate Suboptions. For the other Coded Character Set Validate Suboptions, the example would be the same, except for the values of the MACs which would change from one suboption to another. The test session starts with a READY MESSAGE which is followed by a series of ten (10) KEY MESSAGES and a CONTINUE MESSAGE. The test session then proceeds with a series of n tests (i.e., *test* 1, *test* 2, ..., *test* n). Each test consists of a REQUEST MESSAGE, a RESPONSE MESSAGE, and a CONFIRM MESSAGE. Notice that *tests* 1 and 2, for example, both include a REQUEST MESSAGE which begins with "DATA=" indicating that a MAC must be computed, but not compared to a MAC which may or may not appear in the message. Also notice that *test* 1, for example, includes a REQUEST MESSAGE which contains a message properly formatted according to the coded character set rules of ANSI X9.9. However, *test* 2, for example, includes a REQUEST MESSAGE which contains a message that is not properly formatted according to the coded character set rules of ANSI X9.9 (in this example, the message identifier field (MID) is missing its closing explicit delimiter "-MX"). Even though the proper response is not a valid MAC, the DUT passes the test for providing the proper response. Notice that *tests* 3 and n, for example, both include a REQUEST MESSAGE which begins with "DATAM=" indicating that a MAC must be computed and compared to a MAC which must appear in the message. The MAC which appears in the REQUEST MESSAGE in *test* 3 turns out to be the correct MAC, hence the proper DUT response is a RESPONSE MESSAGE which includes the proper MAC with a "+" in the middle. However, the MAC which appears in the REQUEST MESSAGE in *test* n turns out to be an incorrect MAC, hence the proper DUT response is a RESPONSE MESSAGE which includes the improper MAC with a "*" in the middle. In this example, the DUT passes every test, sends no REPEAT MESSAGES, and does not send a KILL MESSAGE. Thus, the test session ends with a COMPLETION MESSAGE which indicates successful completion of the suboption.

Examples of unsuccessfully completed Coded Character Set Suboptions are not depicted here, because the protocol is identical to that depicted above in the examples for the Binary Validate Suboption.

```
DUT                                                                  MVS

                          READY(^C)
---------------------- READY MESSAGE ---------------------->

    KEY=E6A12F079D15C437(^C)IDA=1357BANKATOBANKB(^C)
<---------------------- KEY MESSAGE -----------------------
          KEY=2A45AE52C8D68325(^C)IDA=KEY2(^C)
<---------------------- KEY MESSAGE -----------------------
                              .
                              .
                              .                                  KEYS
        KEY=CD98AE912580768C(^C)IDA=KEY ONE(^C)
<---------------------- KEY MESSAGE -----------------------
                         CONTINUE(^C)
-------------------- CONTINUE MESSAGE --------------------->

    DATA=QD-851205-DQQK-KEY2-KQQX-MID--XQQT-TXT--TQ(^C)
<-------------------- REQUEST MESSAGE ---------------------
                   QM-3150 CF93-MQ(^C)
-------------------- RESPONSE MESSAGE --------------------->   TEST 1
                          PASS(^C)
<-------------------- CONFIRM MESSAGE ---------------------

    DATA=QD-851205-DQQK-KEY2-KQQX-MIDQK--XQQT-TXT-TQ(^C)
<-------------------- REQUEST MESSAGE ---------------------
                   QM-    *    -MQ(^C)
-------------------- RESPONSE MESSAGE --------------------->   TEST 2
                          PASS(^C)
<-------------------- CONFIRM MESSAGE ---------------------

    DATAM=QD-851205-DQQK-KEY3-KQQX-MID-XQQT-TXTXXXX-TQ
                   QM-E909 C232-MQ(^C)
<-------------------- REQUEST MESSAGE ---------------------
                   QM-E909+C232-MQ(^C)
-------------------- RESPONSE MESSAGE --------------------->   TEST 3
                          PASS(^C)
<-------------------- CONFIRM MESSAGE ---------------------
                              .
                              .
                              .
      DATAM=XQT-TXT-TQXQD-851211-DQXQM-7469 5FF0-MQX
                 QK-KEY3-KQXQX-MID-XQ(^C)
<-------------------- REQUEST MESSAGE ---------------------
                   QM-7469*5FF0-MQ(^C)
-------------------- RESPONSE MESSAGE --------------------->   TEST n
                          PASS(^C)
<-------------------- CONFIRM MESSAGE ---------------------


              OPTION COMPLETED SUCCESSFULLY(^C)
<----------------- COMPLETION MESSAGE --------------------
```

Example A.3.1

A.4 Coded Character Set Entire Message/No Editing Debug Suboption

A.4.1 Example of Test Session

This example depicts a complete test session. The test session starts with a READY MESSAGE which is followed by a series of ten (10) KEY MESSAGES and a CONTINUE MESSAGE. The test session then proceeds with a series of n tests (i.e., *test* 1, *test* 2, ..., *test* n). Each test consists of a REQUEST MESSAGE and a RESPONSE MESSAGE. There is no CONFIRM MESSAGE. The test session ends when the DUT sends a KILL MESSAGE to the MVS.

```
DUT                                                    MVS

                        READY(^C)
<-------------------- READY MESSAGE ----------------------

          KEY=AD37C7FD10ABEF5D(^C)IDA=KEY2(^C)
---------------------- KEY MESSAGE ---------------------->

          KEY=4C40B30D7583F45B(^C)IDA=KEY3(^C)
---------------------- KEY MESSAGE ---------------------->

                              .
                              .                            KEYS
                              .

     KEY=DF45FB2F3173EC8F(^C)IDA=A1B3N5K7,Q8/WTYZ(^C)
---------------------- KEY MESSAGE ---------------------->

                        CONTINUE(^C)
<------------------ CONTINUE MESSAGE ---------------------

  DATA=XQT-TXT-TQXQD-851211-DQXQK-KEY5-KQXQX-MID-XQX(^C)
-------------------- REQUEST MESSAGE -------------------->
                                                           TEST 1
                  QM-C23D 4FB2-MQ(^C)
<------------------ RESPONSE MESSAGE ---------------------

  DATA=QD-851205QK--DQQK-KEY2-KQQX-MID-XQQT-TXT-TQ(^C)
-------------------- REQUEST MESSAGE -------------------->
                                                           TEST 2
                  QM-    *    -MQ(^C)
<------------------ RESPONSE MESSAGE ---------------------

     DATAM=XQT-TXT-TQXQD-851211-DQYQM-990A 01EC-MQX
                QK-KEY2-KQXQX-MID-XQX(^C)
-------------------- REQUEST MESSAGE -------------------->
                                                           TEST 3
                  QM-990A+01EC-MQ(^C)
<------------------ RESPONSE MESSAGE ---------------------
                              .
                              .
                              .

      DATAM=XQT-TXT-TQXQD-851211-DQYQM-1266 50A6-MQX
                QK-KEY3-KQXQX-MID-XQX(^C)
-------------------- REQUEST MESSAGE -------------------->
                                                           TEST n
                  QM-1266*50A6-MQ(^C)
<------------------ RESPONSE MESSAGE ---------------------

                        KILL(^C)
--------------------- KILL MESSAGE ---------------------->
```

Example A.4.1

APPENDIX B: SAMPLE VALIDATION CERTIFICATE

NATIONAL BUREAU OF STANDARDS MESSAGE AUTHENTICATION DEVICE VALIDATION CERTIFICATE

A message authentication device manufactured by _____ successfully passed the National Bureau of Standards (NBS) Message Authentication Device Validation Tests (Version). The tests were designed to validate conformance to the NBS Federal Information Processing Standards Publication (FIPS PUB) 113, *Computer Data Authentication*, and the American National Standards Institute (ANSI) Revised Standard X9.9-1986, *Financial Institution Message Authentication (Wholesale)*. The following hardware and software components were validated:

Hardware: _____

Model: _____ Serial No: _____

Software: _____

Version: _____

The device passed the NBS validation tests for the following ANSI X9.9 options:

Binary Option (FIPS 113) _____

_____

_____

_____

_____

_____

SAMPLE

Devices bearing the above identification and manufactured to the same design specifications may be labelled as complying with the NBS validation tests for the options listed above. No tests other than those described in the NBS *Message Authentication Code (MAC) Validation System: Requirements and Procedures* were performed and no warranty of the device by the NBS is either expressed or implied.

Dated: _____

Signed: _____

Manager, Security Technology Group
Institute for Computer Sciences and Technology
National Bureau of Standards

REFERENCES

[1] American National Standard X9.9-1986 Draft Revision, *Financial Institution Message Authentication (Wholesale)*, American Bankers Association, Washington, D.C., April 7, 1986.

[2] Federal Information Processing Standard Publication (FIPS PUB) 46, *Data Encryption Standard*, National Bureau of Standards, January 15, 1977.

[3] Federal Information Processing Standard Publication (FIPS PUB) 113, *Computer Data Authentication*, National Bureau of Standards, May 30, 1985.

[4] U.S. Department of Treasury Directive 81-80, *Electronic Funds and Securities Transfer Policy–Message Authentication*, Washington, D.C., August 16, 1984.

[5] U.S. Department of Treasury Directive 16-02, *Electronic Funds and Securities Transfer Policy–Message Authentication and Enhanced Security*, Washington, D.C., October 3, 1986.

[6] *Criteria and Procedures for Testing, Evaluating, and Certifying Message Authentication Devices for Federal E.F.T. Use*, Second Edition, U.S. Department of Treasury, Washington, D.C., September 1, 1986.

[7] American National Standard X9.17-1985, *Financial Institution Key Management (Wholesale)*, American Bankers Association, Washington, D.C., 1985.

[8] American National Standard X3.4-1977, *Code for Information Interchange*, American National Standards Institute, New York, New York, 1977.

[9] Gait, Jason, *Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard*, NBS Special Publication 500-20, Revised September, 1980.

[10] Gait, Jason, *Maintenance Testing for the Data Encryption Standard* , NBS Special Publication 500-61, August, 1980.

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS/SP-500/156	2. Performing Organ. Report No.	3. Publication Date May 1988

4. TITLE AND SUBTITLE

Computer Science and Technology:
Message Authentication Code (MAC) Validation System: Requirements and Procedures

5. AUTHOR(S) Miles Smid, Elaine Barker, David Balenson, Martha Haykin

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as item 6.

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number 88-600533.

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The National Bureau of Standards Message Authentication Code (MAC) Validation System (MVS) tests message authentication devices for conformance to two data authentication standards: Federal Information Processing Standard Publication (FIPS PUB) 113, Computer Data Authentication, and American National Standards Institute (ANSI) X9.9-1986, Financial Institution Message Authentication (Wholesale). The MVS is designed to perform automated testing on message authentication devices which are remote to NBS. This publication provides brief overviews of the two data authentication standards and introduces the basic design and configuration of the MVS. The requirements and administrative procedures to be followed by those seeking formal NBS validation of a message authentication device are presented. The requirements described include the specific protocols for communication between the message authentication device under test (DUT) and the MVS, the types of tests which the DUT must pass for formal NBS validation, and general instructions for accessing and interfacing to the MVS. An appendix with examples illustrating the MVS testing protocol is provided.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Automated testing; computer security; cryptography; Data Encryption Standard (DES); data integrity; message authentication; validation

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES 50

15. Price

USCOMM-DC 6043-P80

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS Technical Publications

Periodical

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Bureau of Standards
Gaithersburg, MD 20899

Official Business
Penalty for Private Use $300

Stimulating America's Progress
1913-1988