SDNS
Secure Data
Network System

Message Security Protocol

Forward:

The SDNS architecture, and its associated specifications, were developed as a cooperative project between government and industry. The project was sponsored by the National Security Agency, and supported by the National Institute of Standards and Technology (formerly the National Bureau of Standards) and the Defense Communications Agency. Twelve leading U.S. companies in computers and telecommunications made significant contributions of their technical talents and development resources. The combined efforts of these organizations produced the basis for improved security technology, interoperable security standards, and cost-effective security for computers and telecommunications.

Introductory note:

This document provides a framework for the SDNS Message Security Protocol. This document is being circulated for comment and approval. It is subject to change during the development phase of SDNS.

Table of Contents

0. Introduction

The requirement for secure Electronic Mail and secure messaging has resulted in the Secure Data Network System (SDNS) architecture developed in support of a secure version of the X.400 Message Handling System. This document describes the additions to the CCITT X.400 Recommendations (either 1984 or 1988) that permit any type of message (including interpersonal messages) to be sent and received securely. ANSI has defined X.400 Message Transfer System conventions for Electronic Data Interchange (EDI). Using the ANSI conventions and the SDNS security additions, EDI messages can be exchanged securely. This protocol is known as the Message Security Protocol (MSP).

1. Scope and Field of Application

This document specifies the Message Security services and protocol that will be implemented in SDNS MHS components. The User Agent provides these services by encapsulating the message content and adding a Message Security Protocol heading before submission to the Message Transfer System. SDNS MSP is transparent to the X.400 MTS.

The Message Security Protocol provides writer to reader confidentiality, integrity, data origin authentication, non-repudiation with proof of origin, access control for message transfer, and request for a signed receipt of the received message.

2. References

X.200|ISO 7498	Information Processing Systems - Open Systems Interconnection - Basic Reference Model.
DIS 7498/2	Information Processing Systems - Open Systems Interconnection - Security Architecture.
X.208|ISO 8824	Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).
X.209|ISO 8825	Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).
X.400|ISO 8505-1	Information Processing Systems - Text Communications - Message Handling: Service and System Overview.
X.411|ISO 8883-1	Information Processing Systems - Text communications - Message Handling: Message Transfer System [part 1] Abstract Service Definition and Procedures.
X.419|ISO 8505-2	Information Processing Systems - Text Communications - Message Handling: Protocol Specification.

X.420|ISO 9065	Information Processing Systems - Text Communications - Message Handling: Interpersonal Messaging System.
X.501/DIS 9594	Information Processing Systems - Open Systems Interconnect - The Directory - Models.
SDN.702	SDNS Directory Specifications for Utilization with the SDNS Message Security Protocol.
SDN.801	SDNS Access Control Concept Document.
SDN.802	SDNS Access Control Specification

3. Definitions

3.1 Open System Interconnection

This document uses the following terms contained in the Basic Reference Model for Open Systems Interconnection (IS 7498), including the Security Architecture (IS 7498/2):

Access control
Connectionless confidentiality
Connectionless integrity
Data origin authentication
Non-repudiation with proof of delivery
Non-repudiation with proof of origin

3.2 Message Handling System

This document uses the following terms contained X.400|ISO 8505-1 Information Processing Systems - Text Communications - Message Handling: Service and System Overview:

Content type
Distribution List (DL)
Interpersonal Messages (IPM)
Message Transfer Agent (MTA)
Message Transfer System (MTS)
O/R Name
P2
P3
SUBMIT
User Agent (UA)

3.3 The Directory

This document uses the following term contained in X.501/DIS 9594 Information Processing Systems - Open Systems Interconnect - The Directory - Models:

Directory Information Base (DIB)

3.4 Secure Data Network System

This document uses the following terms from the SDNS Access Control System Specification:

Key Material IDentifier (KMID)
Key Management System (KMS)

3.4 Message Security

For the purposes of this document the following definitions apply:

Directory service (DS): A directory server containing information (e.g., certificates, auxiliary vectors, and user keying material) corresponding to recipient UAs.

Message Security Protocol (MSP): The SDNS protocol for X.400 message security. MSP is a content protocol, and is implemented within the originator and recipient UAs. MSP processing occurs prior to submitting a message to the MTS and after accepting delivery of a message from the MTS. MSP provides security for a content protocol (e.g. IPM, EDI), but is independent of the content protocol.

Originator UA: The user agent process which originates a message.

Recipient UA: The user agent process which receives a message.

4. Symbols and Abbreviations

DIB	Directory Information Base
DS	Directory Service
DL	Distribution List
IPM	Interpersonal Messages
KMID	Key Material Identifier
KMS	Key Management System
MSP	Message Security Protocol
MS	Message Store
MTA	Message Transfer Agent
MTAE	Message Transfer Agent Entity
MTS	Message Transfer System
UA	User Agent
UKM	User Keying Material

5. Overview of the Protocol

The Message Security Protocol (MSP) operates on messages of any type, including, but not restricted to, Interpersonal Messages (IPMs) as defined in X.420. Thus, the term "message" is used throughout this document to refer to any content type defined for transfer by the Message Handling System, reflecting the wide applicability of this protocol. The set of security services provided by this protocol are grouped as indicated below:

- message confidentiality, integrity, data origin authentication and access control
- message non-repudiation with proof of origin
- request for a signed receipt of the received message (only available if non-repudiation with proof of origin is selected)

The protocol operates by encapsulating a message content in a security heading; these together form a new content type designated ProtectedContent. This new content carries the original message content (encrypted if message confidentiality is requested), plus various security parameters required by recipients to decrypt and/or validate the message upon receipt. Also included are parameters which specify the algorithms employed to perform encryption, integrity checking, and signature generation/validation, as appropriate for each service. Optionally, selected fields from the original message can be included in the security heading for perusal by the recipient prior to decryption. This facility is initially defined only for the P2 content type, but other content types can be added as needed. (Invocation of this facility may be restricted based on a particular security doctrine.) The resulting encapsulated content is submitted to the Message Transfer System (MTS) for further processing.

5.1 Access Control

The access control function is integrated within the SDNS MSP implementation. MSP operates in conjunction with the user agent (UA) process responsible for originating and receiving messages. It is technically feasible to reflect information resulting from the access control decision process to the user associated with the UA, whether the user is a human or another process. Access control implementation details are a local matter. It should be noted that access control applies only to messages for which confidentiality, integrity, and data origin authentication services are selected.

5.2 Multiple Recipients

The MHS is a store-and-forward message system where messages can be addressed to multiple recipients. MSP accommodates these characteristics in the distribution of keys for use by cipher functions, in the use of several distinct encryption and integrity algorithms, and in the structure of the MSP heading.

The originator of a message assigns the key to be used to protect the message content. The originator obtains, (e.g., from a directory server), the certificate and the UKM for each intended recipient. In preparing a message for submission, the originator collects a key for each recipient. The originator takes the message key, sensitivity

label, a one-way hash on the message content and other security control information and encrypts it under this pairwise key to form a protected token. The originator then places the protected token for each recipient in the MSP heading. During this process, the originator performs access control checks to ensure that each recipient is authorized to receive the message. Each token is tagged by the originator for identification by the proper recipient. The originator's certificate and UKM are placed in the MSP heading to provide each recipient with the necessary data to process the message.

Upon receipt of the message, each recipient selects its token from the security heading using the tag as a search key. Each recipient then collects his pairwise key and decrypts his token to provide access to the message key, one way hash, and other security control information as described above. The security label is checked to enforce access control policy, the message key is used to decrypt the message, and the one way hash is used to verify the integrity of the message.

Figure 1 depicts the structure of a secure message containing an envelope, a security

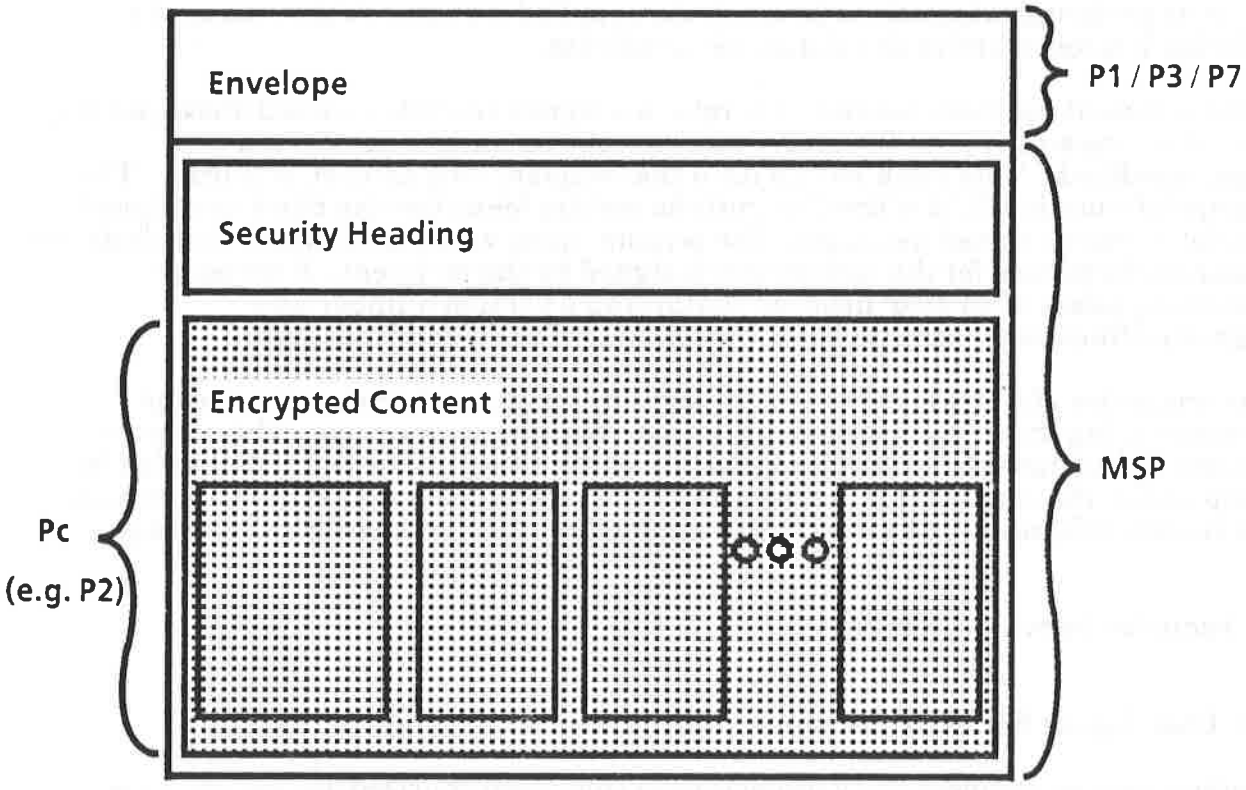

Figure 1. Secure Message Structure

heading, and a message content.

5.3 Message Processing for Signatures and Signed Receipts

If the originator selects the non-repudiation with proof of origin service, a SignatureBlock is included in the MSP heading. This data item includes an identifier

for the signature algorithm, the originator's signatureCertificate, controlInformation, and a signatureValue. To sign a message, the originator calculates a one-way hash on the (original) message content, as above, then continues the calculation to include the SignatureInformation. This field includes an encapsulatedContentType and a contentIdentifier. The resulting hash value is signed and included as the signatureValue in the SignatureBlock.

If the originator elects to request signed receipts from any or all of the recipients, this is indicated in the receiptsIndicator field of the SignatureInformation. If all recipients are requested to return such receipts, a flag is set; if a subset of the recipients are requested to return receipts, a list of their O/RNames is included in the field. An originator might maintain a database, (e.g., indexed by contentIdentifier), indicating the recipients from whom signed receipts for a specified message (and a copy of the message plus the hash value) are expected. This data will be required in support of later receipt processing. The database is a local matter and thus outside the scope of this protocol specification.

A recipient validates a message signature in a two-step process. First, the one-way hash on the received message content and the SignatureInformation is calculated. Then he performs calculations to verify the supplied signatureValue using the information contained in the signatureCertificate.

If the recipient has been requested to return a signed receipt, as noted above, he may generate a receipt or not. To return a receipt, the recipient constructs a new SignatureBlock. This block will contain the recipient's signatureCertificate. The receiptInformation field of the ControlInformation identifies the block as a signed receipt (versus a signed message). The original hash value is extended to include the ControlInformation for the receipt and is signed by the recipient. A message consisting solely of an MSP heading containing a version number and SignatureBlock is transmitted to the originator of the signed message.

The originator of a message requesting signed receipts will process the receipt message using the stored message hash from the original message. This value is extended to include the ControlInformation of the receipt. The computed value is compared to the verified signature value of the receipt message. If the values match, the returned SignatureBlock serves as a signed receipt for the original message.

6. Security Service Elements

6.1 User Agent Security Services

Connectionless confidentiality protects data from unauthorized disclosure. This service is provided by an encryption mechanism that is applied to the message content.

Connectionless integrity protects data from modification. This service is provided by a one-way hash applied to the plaintext message content.

Data origin authentication provides corroboration to the application process that the source of the message is the claimed originator. The key used to encrypt the message content is separately encrypted for each recipient using a token key. Successful

decryption of the message key and successful verification of the one-way hash provides data origin authentication.

Access control for message transfer is a service to both originator and recipient. The originator is prohibited from submitting, and the recipient is prohibited from receiving messages that violate the security policy.

Non-repudiation with proof of origin is a service to originator and recipient. The originator is assured that the signed message did not admit tampering. The recipient of a signed message is assured that the message sent cannot be denied. A digital signature placed in the security heading of the message binds the identity of the originator to the message content. The recipient of the message can subsequently prove to a third party that only the originator could have produced the signature.

Request For Signed Receipt of the received message asks the recipient to digitally sign a hash of the message and return the signature. If the recipient performs this action and the verified receipt hash value supplied by the recipient matches that computed by the originator using the signature verification process, then the originator is provided a form of non-repudiation with proof-of-delivery service.

The Request For Signed Receipt may be used only in conjunction with the non-repudiation with proof-of-origin service. The originator of a message selects which recipients are to return a signed receipt. This list is contained within the signature block in the MSP security heading. The originator's MSP process must include the list of recipients requested to return a signed receipt in the calculation of the hash used to produce the digital signature for the non-repudiation with proof of origin service.

6.2 Required Services from the Message Transfer Service

The basic Message Transfer Service (MTS) enables a user agent (UA) to submit and receive messages. If a message cannot be delivered, the originating UA is informed.

The following Message Transfer Service (MTS) user facilities must be selected when MSP is invoked:

- Content return requested.
- Conversion prohibited.
- Physical delivery prohibited.
- Recipient redirection prohibited.
- DL expansion prohibited.

A site or community specific security doctrine may further restrict possible use of the following optional user facilities:

- Content identifier.
- Priority.
- Disclose recipients.
- Deferred delivery time.
- Latest delivery time.
- Originator report request.

6.3 Required Services from the Directory User Agent

In order to support key distribution for SDNS message security, the Directory Information Base (DIB) must store for each MSP user: certificate, a signature certificate, a set of UKMs with associated tags, and any auxiliary vectors associated with the user. The certificate, signature certificate, and any auxiliary vector(s) contain the O/R name of the user and additional identifying information. The UKMs are submitted by the user and employed in the keying process. The tags identify each UKM and a user-generated signature binds each UKM to its tag. The originator of a message accesses the directory service to obtain address information, the certificate, auxiliary vector, and tagged UKM for each recipient. {Details of these and other key management support functions provided by the directory service are contained in SDN.702 SDNS Directory Specifications for Utilization with the SDNS Message Security Protocol.}

7. Message Security Abstract Service Primitive

Secure-message is the service primitive for use with SDNS X.400 message security. This primitive permits a UA to select one or more security services:

a. Confidentiality, integrity, data origin authentication, and access control;
b. Non-repudiation with proof of origin;
c. Request for signed receipt .

Service c, Request for signed receipt, may be selected only in combination with service b, non-repudiation with proof of origin.

Secure-message service is provided by the UA. The originator of a message requests the UA to process a message content. This request is based on the arguments supplied to invoke the submit operation. The recipient of a message containing a ProtectedContent type invokes the UA to process the message content.

7.1 Secure-message-request parameters

The Secure-message-request primitive requires all of the parameters required by the SUBMIT operation (as defined in CCITT X.411). In addition, the following arguments are required:

ProtectionFlags
conf-integ-doa-ac
nonrepud
requestForSignedReceipt

If the conf-integ-doa-ac flag is set then the following arguments are required:

originatorCertificate
originatorAuxVector
originatorUKM
confidentialityAlgorithm
integrityAlgorithm
tokenConfidentialityAlgorithm
tokenIntegrityAlgorithm
Sensitivity

If the nonrepud flag is set the following arguments are required:

signatureCertificate
signatureAlgorithm

If the requestForSignedReceipt flag is set, the following arguments are required:

contentIdentifier
receiptsIndicator

The ProtectionFlags are BOOLEANS that indicate which of the sets of security services the originator desires. Note that if the requestForSignedReceipt flag is set, then the nonrepud flag must be set.

The originatorCertificate is the unique identification phrase of the originator. The originatorAuxVector is additional access control information required by the security policy in force. The originatorUKM is a portion of the originator's contribution to the token key used to cover the ProtectedToken. The signatureCertificate is used to verify the signature applied to this message.

The confidentialityAlgorithm, integrityAlgorithm,tokenConfidentialityAlgorithm, tokenIntegrityAlgorithm and signatureAlgorithm identify the algorithms used by encryption functions and include any parameters necessary for the operation of the function.The Sensitivity indicates the security level of the message.

The Secure-message-request primitive causes the UA to form a ProtectedContent. This is then input as a Content to the Message Transfer SUBMIT operation.

If an MSP implementation permits the use of blind carbon copies (bcc) and conf-integ-doa-ac is selected, then the MSP process must generate separate copies of the ProtectedContent to input to the SUBMIT operation. One copy should contain the protected tokens for all recipients not on the bcc list and should include these recipients in the SUBMIT argument list. If the identity of each bcc recipient is to be concealed from each other recipient (including other bcc recipients), then a SUBMIT operation should be invoked for each bcc recipient and the protected content should contain a protected token for only the designated recipient. This restriction is not needed if the message is only signed.

The requestForSignedReceipt flag indicates that the originator has requested one or more recipients to return a signed copy of the received message. The contentIdentifier is an octet string used to uniquely identify the message among all those submitted by the originator. The ReceiptsIndicator is either an integer with

value 1, indicating that ALL recipients should be requested to return a signed receipt, a sequence of O/R names identifying the recipients who are to be explicitly requested to respond with a signed receipt, or an integer with value 0, indicating that no receipts are requested. As above, in order to provide maximum privacy in the context of blind carbon copies, multiple submissions will be required to prevent a ReceiptsIndicator sequence from disclosing identities of bcc recipients.

7.2 Secure-message-indicate parameters

The Secure-message-indicate primitive provides all of the arguments provided by the DELIVER operation as defined in (CCITT X.411). In addition, the following arguments are provided:

a. SignatureBlock
b. msgIntegrityValidity
c. sensitivity
d. tokenIntegrityValidity

The SignatureBlock is present if the originator requested the message to be signed. The SignatureBlock contains the signatureAlgorithm, the signatureCertificate, the signatureControlInformation and the signatureValue. The signatureAlgorithm identifies the algorithm. The signatureCertificate is used in the signature verification process. The signatureValue is the result of the signature function calculated for this message. The controlInformation contains the signatureType, the encapsulatedContentType, the signatureContentIdentifier, and the receiptsIndicator.

The signatureValidity, msgIntegrityValidity, and tokenIntegrityValidity arguments are BOOLEANS that indicate the success or failure of the signature, message integrity, and token integrity calculations respecitvely. The Sensitivity argument specifies the security labeling of this message.

8. Message Security Protocol

The confidentiality, integrity, data origin authentication, access control, non-repudiation with proof of origin, and the request for signed receipt services are provided within the User Agent (UA) through the addition of a security heading. The content provided by the UA and the security heading constitute a new content type.

8.1 Message Security Heading

The ProtectedContent is a sequence of the version, OriginatorSecurityData, SignatureBlock, recipient-security-data, bypassed-content-data, and encapsulatedContent. The bypassed-content-data is content dependent data that is unprotected and intended for the recipient to use prior to the invocation of security. processing. The encapsulatedContent is the original message content, after the UA applies the security services, (e.g. the encrypted content if confidentiality has been requested).

OriginatorSecurityData comprises the originator's originatorCertificate, originatorUKM, originatorAuxVector, confidentialityAlgorithm,

integrityAlgorithm, tokenConfidentialityAlgorithm, and tokenIntegrityAlgorithm. The originatorCertificate is the unique identification phrase of the originator. The originatorUKM is the originator key material. The originatorUKM and originatorCertificate are combined with the recipient's posted (to a directory service) UKM and certificate to form the key used to protect the ProtectedToken. The confidentialityAlgorithm, integrityAlgorithm, tokenConfidentialityAlgorithm, and tokenIntegrityAlgorithm identify the algorithms used by encryption functions and include any parameters necessary for the operation of the function.

The SignatureBlock contains the digital signature used for non-repudiation with proof of origin. The SignatureBlock consists of the user's signatureAlgorithm, signatureCertificate (as presented in the Directory), the signatureValue, and the signatureControlInformation.

The PerRecipientToken comprises the recipient's Tag and ProtectedToken.

SignatureControlInformation comprises the signatureType, the encapsulatedContentType, the signedContentIdentifier, and the ReceiptsIndicator. The signatureType contains a zero (0) if the ProtectedContent is a signed message (signedMsg), a one (1) if the ProtectedContent is a signed message with signed receipts requested (sMsgReqRecp), and a two (2) if the ProtectedContent is a signed receipt (signedReceipt). The encapsulatedContentType is the ContentType of the original message being protected. The signedContentIdentifier identifies the message among all those submitted by the originator.

The ReceiptsIndicator is either an integer or a sequence of O/RNames. If ReceiptsIndicator is an integer, then a value of zero (0) indicates no receipts are requested (AllOrNone), and a value of one (1) indicates all recipients must return a receipt (ReceiptList). If ReceiptsIndicator is a sequence of O/RNames then the originator requests a signed receipt from each recipient listed.

The Tag is the identifier associated with the recipient's posted certificate and UKM, and includes the edition and effective period of the certificate and UKM. The Tag is not encrypted.

The ProtectedToken comprises the msgKey, msgHash, Sensitivity, encapsulatedContentType, signatureBlockIndicator and token-integrity-check. The msgKey is the key used by the encryption function identified by the confidentialityAlgorithm to encrypt the message body. The msgHash is the result of the checkfunction applied to the entire Content. The checkfunction is identified by the integrityAlgorithm. The Sensitivity indicates the security level of the encrypted portions of the message. The encapsulatedContentType identifies the type of the original content before MSP processing. The signatureBlockIndicator is a flag indicating to the recipient that the originator has requested a signature block. The token-integrity-check is the result of an integrity function applied to the concatenation of the msgKey, msgHash, sensitivity, encapsulatedContentType, and signatureBlockIndicator. The integrity function is identified by the tokenIntegrityAlgorithm. The Protected Token is encrypted using the token key.

```
MSP DEFINITIONS ::=
    BEGIN
    -- the following types must be imported from other modules
    -- Content
    -- AlgorithmIdentifier
    -- ContentType
    -- ORName

ProtectedContent ::= [48] SEQUENCE OF {
      version                    [0] IMPLICIT INTEGER,
      originatorSecurityData     [1] IMPLICIT OriginatorSecurityData
                                         OPTIONAL,
      signatureBlock             [2] IMPLICIT SignatureBlock
                                         OPTIONAL,
      recipient-security-data    [3] IMPLICIT SET PerRecipientToken
                                         OPTIONAL,
      bypassed-content-data      [4] IMPLICIT ANY OPTIONAL,
      encapsulatedContent        [5] IMPLICIT Content OCTETSTRING  }

OriginatorSecurityData ::= SET {
      originatorCertificate      [0]  IMPLICIT OCTETSTRING,
      originatorUKM              [1]  IMPLICIT OCTETSTRING,
      originatorAuxVector        [2]  IMPLICIT OCTETSTRING
                                         OPTIONAL,
      confidentialityAlgorithm   [3]  IMPLICIT AlgorithmIdentifier,
      integrityAlgorithm         [4]  IMPLICIT AlgorithmIdentifier,
      tokenConfidentialityAlgorithm [5] IMPLICIT AlgorithmIdentifier,
      tokenIntegrityAlgorithm    [6]  IMPLICIT AlgorithmIdentifier  }

SignatureBlock ::= SET {
      signatureAlgorithm         [0]  IMPLICIT AlorithmIdentifer,
      signatureCertificate       [1]  IMPLICIT OCTETSTRING,
      signatureValue             [2]  IMPLICIT OCTETSTRING,
      controlInformation         [3]  IMPLICIT ControlInformation }

ControlInformation::= CHOICE {
      signatureInformation       [0]  IMPLICIT SignatureInformation,
      receiptInformation         [1]  IMPLICIT ReceiptInformation }

PerRecipientToken::= SET {
      tag                        [0]  IMPLICIT Tag,
      protectedToken             [1]  IMPLICIT ProtectedToken }

SignatureInformation ::= SEQUENCE OF {
      encapsulatedContentType    [0]  IMPLICIT ContentType OPTIONAL,
      signedContentIdentifier    [1]  IMPLICIT OCTET STRING,
      receiptsIndicator          [2]        ReceiptsIndicator OPTIONAL }
```

```
ReceiptsIndicator ::= CHOICE {
      allOrNone                    [0]  IMPLICIT AllOrNone,
      receiptList                  [1]  IMPLICIT ReceiptList}

AllOrNone ::= INTEGER {
                                        noReceipt (0),
                                        allReceipt (1)  }

ReceiptList ::=  SEQUENCE OF ORName

ReceiptInformation ::= SEQUENCE OF  {
      encapsulatedContentType      [0]  IMPLICIT ContentType OPTIONAL,
      signedContentIdentifier      [1]  IMPLICIT OCTET STRING,
      signatureValue               [2]  OCTETSTRING   }

Tag  ::= SEQUENCE {
      kmid                         [0]  OCTETSTRING,
      edition                      [1]  INTEGER
      dateString                        UTCTime   }

ProtectedToken ::= SET {
      msgKey                       [0]  IMPLICIT OCTETSTRING,
      msgHash                      [1]  IMPLICIT OCTETSTRING,
      sensitivity                  [2]       Sensitivity,
      encapsulatedContentType      [3]  IMPLICIT ContentType,
      signatureBlockIndicator      [4]  IMPLICIT BOOLEAN,
      token-integrity-check        [5]  IMPLICIT OCTETSTRING  }

Sensitivity ::= CHOICE {
      ccitt                        [0]  Ccitt,
      dod                          [1]  DoD }

Ccitt ::= INTEGER {
                                        personal  (0),
                                        private   (1),
                                        companyConfidential  (2) }


DoD ::= INTEGER {
                                        unclassified(85),    -- 0101 0101
                                        confidential(122),   -- 0111 1010
                                        secret(173),         -- 1010 1101
                                        top-secret(222)      -- 1101 1110
                                          }

END -- of MSP
```

8.2 Elements of the Procedure

8.2.1 Originating a Secure Message

8.2.1.1 Secure-message-request primitive issued

The originator UA presents, for MSP processing, a message content that is accompanied (implicitly or explicitly) by submission envelope information. If the UA and MTA reside in the same system, an explicit submission envelope may not be employed but equivalent information will be present as the message is transferred between the UA and the MTA. From the submit envelope, MSP processing uses the following data items:

- originator O/R name (if not implicitly provided)
- recipient O/R name list
- sensitivity
- ContentType designation for the message
- message Content

Message security label information is determined in one of two ways: (implicitly) based on local processing context, or (explicitly) based on the Sensitivity parameter. The UA may require some or all of the offered security services to be invoked for every message, or may allow a subset from this list.

8.2.1.2 Calculate Hash values

If the conf-integ-doa-ac flag is set, a msgHash is calculated. If the msgHash algorithm requires a key, the msgKey is used.

8.2.1.3 Sign message

If the nonrepud flag is set and the requestForSignedReceipt is not set, the signatureType is set to signedMsg (0) indicating that a signature is present. If the nonrepud flag is set and the requestForSignedReceipt flag is also set, signatureType is set to sMsgReqRecp (1) indicating that a signed message with a request for signed receipt is present. When the requestForSignedReceipt flag is set, the receiptsIndicator and contentIdentifier are included in the SignatureInformation. In either case, a signature hash is calculated over the message content and the SignatureInformation. This signature hash is called the signatureValue. The signatureAlgorithm, the signatureCertificate, the signatureValue, and the SignatureInformation form the SignatureBlock, which is included in the MSP message.

If the requestForSignedReceipt is set, a local database entry is contructed which includes the contentIdentifier, signatureValue, receiptsIndicator, and the message content. This database is used for later processing of signed receipts.

8.2.1.4 Encipher content

If the conf-integ-doa-ac flag is set, a message encryption key (msgKey) is generated and used to encipher the submitted message content.

8.2.1.5 Calculate token-integrity-check

If the conf-integ-doa-ac flag is set, the msgKey, the msgHash, the Sensitivity, the encapsulatedContentType, and the signatureBlockIndicator fields are concatenated and a token-integrity-check is calculated. The result is then appended to these fields all of which is referred to as the Token for this message.

8.2.1.6 Produce ProtectedToken

For each recipient in turn, the Token is encrypted using the token key. The recipient's key and the UKM associated with the encryption of the Token for a given recipient is the one which the recipient posted to the Directory System and which is designated by the Tag as being valid for the time at which the message processing began.

8.2.1.7 Produce PerRecipientToken

Each resulting ProtectedToken from step 6 is paired with the corresponding Tag of the intended recipient (the Tag is associated with the recipient's posted certificate and UKM to identify the UKM employed). The resulting set of data items is incorporated into the recipient-security-data contained in the MSP secure heading.

8.2.1.8 Place bypass data in the security heading

If the submitting entity designated any data to be bypassed, and if the implementation permits such bypass, this data is included in the MSP message. The content type of the encapsulated message is also included in the security heading. Since the data to be bypassed may differ as a function of the encapsulated content type, the recipient must examine the field which specifies the encapsulated content type to determine how to interpret this bypassed data. The bypassed data shall be represented in a fashion so that for a specified content type any intended recipient will be capable of interpreting the bypassed data.

8.2.1.9 Submit the secure message

The newly formed message is submitted to the MTA with the (validated) recipient list, SecureMessage ContentType (=48), ConversionProhibited flag, and AlternateRecipientProhibited flag set.

8.2.2 Receiving a Secure Message

8.2.2.1 Recipient UA accepts delivery of message

The recipient requests the UA to accept delivery of a message from an MTA, or the UA retrieves a message from an MS. ContentType is 48, therefore MSP processing commences. If the submitting entity designated any data to be bypassed, and if the implementation permits such bypass, this data is included in the MSP heading. The content type of the encapsulated message is also included in the security heading. Since the data to be bypassed may differ as a function of the encapsulated content type, the recipient must examine the field which specifies the encapsulated content type to determine how to interpret this bypassed data. The bypassed data shall be represented in a fashion so that for a specified content type the recipient can interpret the bypassed data.

8.2.2.2 Select correct PerRecipientToken

MSP process identifies the correct PerRecipientToken for this recipient based on the Tag.

8.2.2.3 Decrypt ProtectedToken

The MSP process uses the token key to decrypt the ProtectedToken.

8.2.2.4 Verify token-integrity-check

A token-integrity-check is calculated and compared to the originator's token-integrity-check value. If the compare is unequal the recipient is notified (through local means) that the message has been modified.

If the signatureBlockIndicator is true, then the signature block should be present. If it should be and isn't then the originator should be notified that the message has been modified.

8.2.2.5 Decrypt content

If the conf-integ-doa-ac flag is set the content is decrypted using the msgKey.

8.2.2.6 Verify content integrity

The msgHash is calculated over the message content. If the calculated value does not equal the originator's msgHash, the recipient is notified through local means that the message has been modified.

8.2.2.7 Verify Signature

If the SignatureBlock is present in the security heading, a recipient validates the message signature in a two-step process. First, he calculates the one-way hash on the received message content and the SignatureControlInformation. Then he performs calculations to verify the supplied signatureValue using the information contained in the signatureCertificate.

If the signatureType indicates that a receipt is requested and the ReceiptsIndicator indicates that a return receipt is requested from all recipients or if it indicates selective receipting and this recipient's O/R name appears on the receiptList, the recipient is notified through local means of the request for signed receipt. Processing to generate a signed receipt is described in section 8.2.3.

If the signatureType indicates that this message is a signed receipt, the signatureValue is calculated using the supplied signatureCertificate. The contentIdentifier is used to search a local database to retrieve the signatureValue associated with the original message. A hash is calculated using this value as the initialization value and continuing over the ControlInformation in this receipt. If this value matches that from the receipt the receipt is deemed valid and the local database is updated accordingly, otherwise, the recipient is notified that receipt validation failed.

8.2.2.8 Deliver Secure Message

The Secure-message indication occurs and the arguments of the primitive, as specified in section 7.2, are supplied to the UA.

8.2.3 Generating a Signed Receipt

If, as described in section 8.2.2.7, a signed receipt is requested by the originator and the recipient elects to comply with the request, a SignatureBlock is constructed. The SignatureBlock contains SignatureCertificate of this recipient and the SignatureInformation specifies a signed receipt as the signatureType. That is, signatureType is set to signedReceipt (2). The signatureValue is computed by signing the signatureValue of the original message and the ControlInformation in the receipt. The resulting SignatureBlock is concatenated with the version to form an MSP message and is submitted to the originator of the received, signed message.