Common Security Protocol (CSP)

ACP 120

June 1998

Foreword

1. ACP120, COMMON SECURITY PROTOCOL, is an UNCLASSIFIED publication. Periodic accounting is not required.

2. ACP120 will be effective for National, Service, or Allied use when directed by the appropriate Implementing Agency; refer to the National Letter of Promulgation (LOP).

3. This publication contains Allied military information and is furnished for official purposes only.

4. This publication is not releasable without prior approval from the United States Military Communications-Electronic Board (USMCEB).

5. It is permitted to copy or make extracts from this publication without consent of the Authorizing Agency.

Letter of Promulgation

RECORD OF CHANGES AND CORRECTIONS

Enter Change or Correction in Appropriate Column

Identification of Change or Correction; Reg. No. (if any) and date of same		Date Entered	By whom entered (Signature; rank, grade, or rate; name of command)
Change	Correction		

RECORD OF CHANGES AND CORRECTIONS

Enter Change or Correction in Appropriate Column

Identification of Change or Correction; Reg. No. (if any) and date of same		Date Entered	By whom entered (Signature; rank, grade, or rate; name of command)
Change	Correction		

Table of Contents

Chapter 1

General

101. Introduction

a. The requirement for secure electronic mail and secure messaging resulted in the development of a security protocol to be used with the CCITT X.400 Message Handling System. This protocol was called the Message Security Protocol (MSP). MSP was initially considered for Defense messaging applications only. In order to commercialize MSP as a Common Security Protocol (CSP) for wider use, the original document (SDN.701) has been re written to form this document.

b. While this specification is oriented around use within an X.400 Message Handling System, CSP may also be used as a secure message/protocol encapsulation facility with other distributed computing environments, such as directory access and key material distribution.

c. This document describes additional functionality to the CCITT X.400 Recommendations (either 1984, 1988, or 1992) that permit any type of message (including interpersonal messages) to be sent and received securely. For example, the ANSI defined X.400 Message Transfer System conventions for Electronic Data Interchange (EDI) can be exchanged securely or application level operations between distributed directory functions.

102. Scope and Field of Application

a. The CSP provides originator to recipient security services. These security services include content confidentiality and integrity, proof of content origin, and proof of content delivery.

b. This document specifies the services and protocol implemented in a CSP User Agent (CSP UA). The CSP UA provides these services by encapsulating the message content and adding a CSP heading before submission to the MTS. CSP is used to protect data regardless of the format of the encapsulated content. However, to enable the definition of CSP in context, its use within X.400 is chosen. CSP is a content that the X.400 MTS transports, and thus is transparent to the X.400 MTS.

c. While this specification is oriented around use within an X.400 Message Handling System, CSP may also be used as a secure message encapsulation facility with other messaging environments. The examples provided in this specification for integrating CSP services and mechanisms with the X.400 MHS model can be applied to other models. This specification refers to the X.400 MHS as an example, but it does not limit the use of CSP to only secure the X.400 MHS.

103. Structure of This Document

a. The structure of this publication is as follows:

Chapter 1 General - includes references, definition, and abbreviations.

Chapter 2 Elements of Service - is a description of the security services provided by CSP.

Chapter 3 Procedures For The Common Security Protocol - contains the processing rules for CSP.

103. (Continued)

Chapter 4 Definitive CSP ASN.1 Module - defines the structures of a CSP protected message.

Chapter 5 Error Conditions - describes CSP error conditions.

104. **References**

a. CCITT X.200 Open Systems Interconnection - Basic Reference Model for CCITT Applications.

b. ISO 7498/2 Information Processing Systems - Open Systems Interconnection - Security Architecture.

c. CCITT X.208 Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).

d. CCITT X.209 Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).

e. CCITT X.400 Message Handling: Service and System Overview.

f. CCITT X.411 Message Handling: Message Transfer System [part 1] Abstract Service Definition and Procedures.

g. CCITT X.419 Message Handling: Protocol Specification.

h. CCITT X.420 Message Handling: Interpersonal Messaging System.

i. CCITT X.501 The Directory - Models.

j. ITU-T X.509 (1997)|ISO/IEC 9594-8: 1997, Information Technology -- Open System Interconnection - The Directory: Authentication Framework

k. CCITT X.518 The Directory - Procedures for Distributed Operation.

l. CCITT X.519 The Directory - Protocol Specification.

m. CCITT X.525 The Directory - Replication.

n. SDN.701 SDNS Message Security Protocol.

o. ACP 133 Common Directory Services and Procedures, draft, Version 1.3, August 1996.

105. **External Definitions**

105.a **Open System Interconnection**

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

(1) Access control
(2) Connectionless confidentiality
(3) Connectionless integrity
(4) Data origin authentication
(5) Non-repudiation with proof of delivery
(6) Non-repudiation with proof of origin

105.b **Message Handling System**

This document uses the following terms contained in CCITT X.400, Message Handling: Service and System Overview:

(1) Content
(2) Content type
(3) Distribution List (DL)
(4) Interpersonal Messages (IPM)
(5) Message Transfer Agent (MTA)
(6) Message Transfer System (MTS)
(7) O/R Address
(8) O/R Name
(9) P2
(10)P3
(11)P7
(12)SUBMIT
(13)User Agent (UA)

105.c **The Directory**

This document uses the following terms contained in CCITT X.500 series, The Directory:

(1) Directory User Agent (DUA)
(2) Directory System Agent (DSA)

105.d **The Directory Authentication Framework**

This document uses the following terms contained in CCITT X.509, The Directory Authentication Framework:

(1) Certificate
(2) Certification Authority (CA)
(3) Certificate Revocation List (CRL)
(4) Compromised Key List (CKL)

105.e **Secure Data Network Systems**

This document use the following terms from the SDNS specifications:

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

106. **Message Security Definitions**

For the purposes of this document the following definitions apply:

a. Common Security Protocol (CSP): The commercially applied common security protocol for X.400 message security, X.500 directory access security and distributed computing environments based on MSP (see MSP below).

b. Common Security Protocol User Agent (CSP UA): A messaging user agent which includes an active implementation of the CSP.

c. Directory Service (DS): A database server containing information (e.g., certificates and user keying material) corresponding to recipients.

d. Local Management Authority (LMA): The LMA controls a single local access control domain by signing attribute certificates for that domain.

e. Mail List (ML): A list or group of recipients which are addressed with a single O/R Address, i.e. an Address List.

f. Mail List Agent (MLA): An agent which a message originator addresses and which represents a group of recipients. The MLA provides message security processing and distribution to that group on behalf of the message originator.

g. Mail List Key (MLK): A token protection key held by all members of a mail list or addressable group.

h. Message Security Protocol (MSP): The SDNS protocol for X.400 message security. MSP for X.400 messaging is a content protocol and is implemented within the originator and recipient MSP 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 the MSP user's content protocol (e.g. IPM, EDI), but MSP is independent of the MSP user's content protocol.

i. Message Security Protocol User Agent (MSP UA): A user agent which includes an active implementation of the SDNS MSP.

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

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

l. Travelling User (TU): A CSP user visiting a CSP equipped facility other than the one at which the user normally processes messages or receives directory access and who wishes to read, process, and/or originate CSP protected messages. A travelling user does not transport CSP equipment, but makes use of CSP equipment located at the site being visited.

107. **X.500 Security Definitions**

For the purposes of this document the following definitions apply:

a. *authentication token (token)*: Information conveyed during a strong authentication exchange, which can be used to authenticate its sender;

b. *user certificate (certificate)* : The public keys of a user, together with some other information, rendered not forgeable by encipherment with the private key of the certification authority which issued it;

c. *certification authority* : An authority trusted by one or more users to create and assign certificates. Optionally the certification authority may create the user's keys;

d. *certification path* : An ordered sequence of certificates of objects (certificate users and issuers) in the DIT which, together with the public key of the initial certificate in the path, can be processed to obtain that of the named object in the path;

e. *cryptographic system*, *cryptosystem* : A collection of transformations from plain text into ciphertext and vice versa, the particular transformation(s) to be used being selected by keys. The transformations are normally defined by a mathematical algorithm.

f. *hash function* : A (mathematical) function which maps values from a large (possibly very large) domain (information set) into a smaller range. A “good” hash function is such that the results of applying the function to a (large) set of values in the domain will be evenly distributed (and apparently at random) over the range;

g. *one-way function* : A (mathematical) function f which is easy to compute, but which for a general value y in the range, it is computationally difficult to find and a value x in the domain such that f(x) = y. There may be a few values y for which finding x is not computationally difficult;

h. *public key* : (In a public key cryptosystem) that key of a user’s key pair which is publicly known;

i. *private key* (*secret key* — deprecated): (In a public key cryptosystem) that key of a user's key pair which is known only by that user;

j. *simple authentication* : Authentication by means of simple password arrangements;

k. *security policy* : The set of rules laid down by the security authority governing the use and provision of security services and facilities;

l. *strong authentication* : Authentication by means of cryptographically derived credentials;

m. *trust*:: Generally, an entity can be said to “trust” a second entity when it (the first entity) makes the assumption that the second entity will behave exactly as the first entity expects. This trust may apply only for some specific function. The key role of trust in the authentication framework is to describe the relationship between an authenticating entity and a certification authority; an authenticating entity shall be certain that it can trust the certification authority to create only valid and reliable certificates;

n. *certificate serial number*: An integer value, unique within the issuing CA, which is unambiguously associated with a certificate issued by that CA.

108. **Symbols and Abbreviations**

ACP	Allied Communications Publication
CA	Certification Authority
CMIP	Common Management Information Protocol
CMISE	Common Management Information Service Elements
CSP	Common Security Protocol
CSP DSA	Common Security Protocol Directory System Agent
CSP DUA	Common Security Protocol Directory User Agent
CSP UA	Common Security Protocol User Agent
DAP	Directory Access Protocol
DIB	Directory Information Base
DISP	Directory Information Shadowing Protocol
DL	Distribution List
DOP	Directory Operational Protocol
DSA	Directory System Agent
DSP	Directory System Protocol
DUA	Directory User Agent
IPM	Interpersonal Message
KMID	Key Material Identifier
KMS	Key Management System
LMA	Local Management Authority
LRBAC	Local Rule-Based Access Control
ML	Mail List
MLA	Mail List Agent
MLK	Mail List Key
Ms	Message Store
MSP	Message Security Protocol
MSP UA	Message Security Protocol User Agent
MTA	Message Transfer Agent
MTS	Message Transfer System
PRBAC	Partition Rule-Based Access Control
RA	Release Authority
UA	User Agent
UKM	User Keying Material

Chapter 2

Elements of Service

201. Elements of Service

a. CSP offers security services that are applied directly to the content. This content can be any type of information object, but is described here within the context of a message system. The security services (with the mechanism listed within parenthesis) provided by CSP include:

(1) Content Confidentiality and Integrity (encryption and key distribution) with authenticated key distribution and support for rule-based access control.
(2) Proof of Content Origin (message signature).
(3) Proof of Content Delivery After Security Processing (signed receipts).
(4) Integrity of CSP Encapsulation (sequence signature).
(5) Plain Text Content Description (content description).

b. A system that offers security for messages requires several additional supporting processes, such as group addressing or a time stamping service. CSP conveys information that can support these processes, and this information includes:

(1) Mail List Expansion History (ML Expansion History)
(2) Additional Signatures
(3) Extensions

c. The addition of CSP security services has an effect on some of the other capabilities offered by a message system. Not all message systems offer these capabilities, but when they do, special considerations need to me made with respect to CSP processing. These capabilities are:

(1) Forwarding
(2) Group Addressing
(3) Content Type and Message Identification
(4) Blind Copies

d. CSP is designed to provide these services for messages sent to one or more recipients. The facilities in X.400 which support multiple recipients remain essentially unchanged with the addition of CSP. The CSP security services to be applied to a particular message are selected by the user within the constraints of a site policy. As a consequence of the staged delivery nature of electronic messaging, there is no direct, real-time association formed between an originator and a recipient. Therefore, all of the security functions must be performed independently by the originator and by the recipient, based on both originator and recipient information.

e. The CSP operates by performing security operations on X.400 messages at the originator and recipient UAs. These functions are performed in an independent but consistent fashion at each end of the message exchange based on user security information. This security information includes the user's identity, authorizations, and cryptographic material. CSP processing includes both per-message operations and information and per-recipient operations and information. These operations involve the parsing and generation of elements of the CSP heading based on the services requested by the originator, and the encryption, when requested, of the message content. Per-recipient operations involve the generation of information (known as per-recipient tokens) needed by each recipient to decrypt the message.

202. **Confidentiality of Content and Integrity**

a. This element of service enables an originator to ensure the confidentiality of the content of the message, and assures the recipient that the message content he received is the same as that which the originator sent. Also, this element of service assures the originator that the message he sent cannot be modified without the recipient detecting the modification.

b. Confidentiality and integrity are provided through the encryption of the message content, the calculation of a message hash, and the encryption of the security label, if it is present. CSP offers content confidentiality and integrity to limit disclosure of the message to only the intended recipients. CSP makes no assumptions about the security services provided by lower layers.

c. The associated key management mechanism assures that only the intended recipients can decrypt the message. The authenticated key distribution provides a form of data origin authentication, because it assures a recipient that the message was originated by the user indicated as the originator. In addition the authenticated key distribution preserves, through the staged delivery of a message, a form of peer entity authentication, because it assures the originator that only his intended recipients can receive the message.

d. The CSP UA performs per-recipient processing only for encrypted messages by creating tokens. A token is created for each recipient of the message and for the message originator. The token contains information needed to decrypt and validate the integrity of the message. It also contains information regarding whether the message is also signed, and optionally information in support of rule-based access control.

e. A message key is used to encrypt the message, and also a message hash is calculated. To form a token, this message key and message hash, along with additional control information, is encrypted using one of three key techniques. The techniques allow the use of public key based key agreement technique, public key based key transfer technique, or previously distributed symmetric keys. Both of the public key cryptographic techniques require use of both the originator's and recipient's cryptographic material. When using either of these techniques, this publication uses the term pairwise to describe the cryptographic protection of the token. This assures that one recipient can not modify the message unbeknown to another recipient. Also, this publication focuses on the public key based key agreement technique, because this is the current CCEB agreed key distribution technique.

f. When access control is used within CSP, it involves rule based access control. Based on the sensitivity of the message and the authorizations of the originator, recipient, and workstation, CSP makes the access control decision. Identity-based access controls are the responsibility of the originator, supported by the authenticated key distribution provided by CSP.

g. In support of rule-based access control, this element of services permits an originator to apply a security label to a message and securely bind the label to the content of the message by including an integrity value (hash calculated using the security label) for the security label. Additionally, this security label is encrypted using the same key used to encrypt the content. On receipt, this element of service enables a recipient to verify the binding of the security label to the CSP encapsulated content. CSP processing uses the security label at both origination and reception to make an access control decision, which limits disclosure to authorized parties.

203. **Proof of Content Origination**

a. This element of service provides the recipient of a message with proof that the message came from the user who originated the message. Proof of content origination is provided through a digital signature. The public key needed to validate the digital signature and identity of the message originator are included in the originator’s certificate. The certificate validation process ensures that the originator is a valid user.

b. This service supports the non-repudiation service, which provides the recipient with evidence that demonstrates, to a third-party, who originated the message, and protects against any attempt by the message originator to falsely deny having sent the message. This evidence, provided by the proof of content origin service, is the digital signature along with the certificates necessary to verify it. However, to preclude a subsequent denial by the originator of having sent the message, the digital signature for this particular message must not be effected by subsequent revocations of the originator's certificates. To preserve this evidence, additional records management procedures must be applied. These include trusted timestamps with another signature, and audit logs.

204. **Proof of Content Delivery**

a. This element of service provides the originator of a message with proof that the recipient CSP UA successfully performed CSP processing of the message. Proof of content delivery is provided when the recipient CSP UA returns a digitally signed receipt. Recipient CSP UAs may automatically return signed receipts without interaction with the human user. The public key needed to validate the digital signature on the receipt and identity of the message recipipient are included in the recipient’s certificate. The certificate validation process ensures that the recipient is a valid user.

b. This service supports the non-repudiation service, which provides the originator with evidence that demonstrates, to a third-party, who received the message, and protects against any attempt by the message recipient to falsely deny having received the message. However, to preclude a subsequent denial by the recipient of having received the message, the signed receipt for this particular message must not be effected by subsequent revocations of the recipient's certificates. To preserve this evidence, additional records management procedures must be applied. These include trusted timestamps with another signature, and audit logs.

c. The originator may specify the recipient from which he requests proof of content delivery, via a signed receipt. After successfully validating the signature of the original message, the recipient calculates the signature value in the signed receipt by including the signature value from the original message. This binds the receipt to the original message. Consequently, this service may be requested only if a signature value was generated for the original message (proof of content delivery).

d. To validate a signed receipt, the originator of the message should store information about the message and its original signature. Then upon receiving a signed receipt, it may be validated by using the stored information to verify the signature contained in the receipt.

205. **Integrity of CSP Encapsulation**

This element of service provides a CSP entity which handles a message with proof of the identity of the CSP entity that last processed the message and that an unauthorized party has not modified the CSP heading. CSP offers this service by the calculation of a digital signature over the entire CSP content, including CSP heading fields and protected content. (Note, if the message content is encrypted, this signature is calculated over the encrypted content and the per-recipient tokens used for key management.)

206. **Content Description**

The content description is information available prior to CSP processing. The content description is a string of characters (not encrypted), which may offer some processing information to the recipient or support delivery to the appropriate recipient.

207. **Mail List Expansion History**

This service offers the recipient of a message information about the intermediate processes that have processed the message to expand address lists. This service also provides information to the process that expanded a message to assure it that the message has not been previously processed. This prevents messages from “looping” through these processes indefinitely. See paragraph 211 below. A process that expands an address list must use this service and record its identity in the CSP heading. This process must also apply integrity of CSP encapsulation.

208. **Additional Signatures and Time Stamping**

a. Additional signatures may be used to demonstrate that the message or receipt was prepared or received by a particular time, where the time stamp is provided by a trusted third party. Likewise, additional signatures of third parties on messages can be used to convey support or concurrence to the content. Also, this service provides a means to support the conveyance of digital signatures into non-CSP environments; e.g. X.400 S1c environments. When used in this manner, the additional signature is not necessarily generated by a third party, but by the message originator.

b. To obtain an additional signature on a message content, the message content or the message hash must be transferred to the third party, then the third party generates a signature. The signature may optionally cover the originator's signature, and optionally a time stamp in addition to the message content.

c. To obtain an additional signature on a receipt, the receipt or the receipt hash must be transferred to the third party, then the third party generates a signature. The signature may optionally cover a time stamp in addition to the receipt.

209. **Extensions**

a. This element of service enables a user to apply a registered extension to the CSP. This extension must be understood by both the originator and recipient for it to convey any meaning.

b. Extension may be used to support routing decisions while a message is in transit within a protected environment. However, when the message leaves the protected environment, the extension should be removed. A plain text security label used to support routing decisions would be conveyed in an extension.

210. **Forwarding**

a. One of the characteristics of the proof of content origination service provided by message signatures is the ability to establish the identity of the originator to a third party. Since the forwarding of messages is a standard part of electronic message systems, forwarding signed messages should be a part of these systems also. Likewise, since replying to a message and including the original message in the response is a standard part of electronic message systems, inclusion of a signed message in a reply should be a part of these systems also.

b. Messages are forwarded or included in a reply by placing them within the content of a new message. An CSP protected message can be forwarded **only** if the protected content supports forwarding, because the CSP message is forwarded by including it within the new content. For example, Recommendation X.420 defines a Forwarded Interpersonal Message as a message body part, which is used to forward a previously received IPM. This same body part may contain the original message associated with a reply. Similarly, this document defines a body part, which allows a previously received CSP message or any CSP message, along with its CSP heading which may contain its signature, to be included as part of a new content. This body part is a Forwarded-CSP-Message-Body-Part, which is an extended X.420 body part, and is assigned a registered Object Identifier.

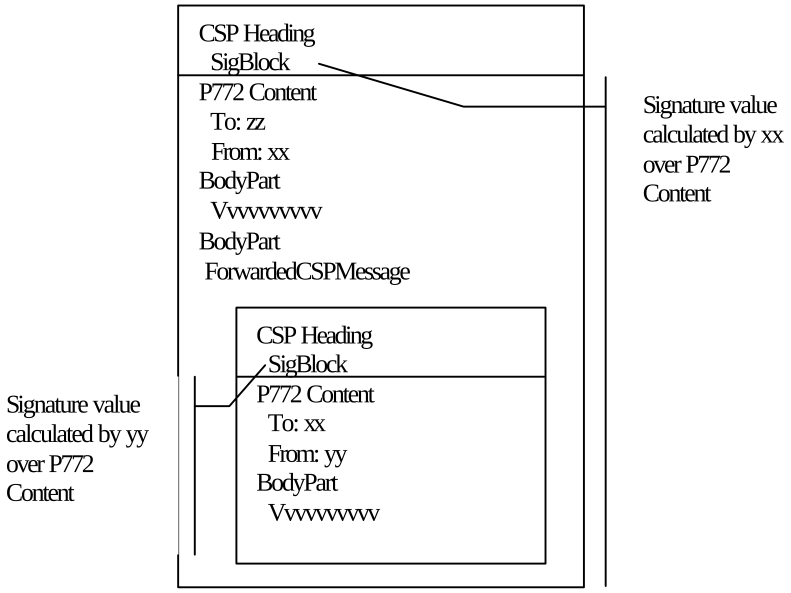

Figure 2-1 Forwarded Signed Message

210. (Continued)

c. Any number of forwarded CSP messages may be conveyed within a new message. Also, forwarded CSP messages may be nested within one another, and signed receipts may be forwarded. The forwarded CSP message may contain only the original unencrypted content and the original CSP signature; however, forwarded messages must be adequately protected. Encrypted CSP messages may also be forwarded if the recipient possesses the cryptographic material required to decrypt the forwarded CSP message. The forwarded CSP message may also contain other CSP heading fields such as the extensions field element, subject to the local security policy.

211. **<u>Group Addressing</u>**

Group addressing allows the message originator to submit a protected message to multiple recipients without selecting each individual recipient or the burden of per-recipient processing for each recipient. The description of originator processing indicates that the CSP UA generates a token for each recipient of an encrypted message. This process can impair performance for messages sent to a large number of recipients. Consequently, CSP includes support of address lists by a process called a Mail List Agent (MLA). An MLA appears to the message originator as a normal message recipient, but the MLA acts as a message expansion point for a Mail List (ML). The administrator of an ML is responsible for establishing rules governing the submission of messages to the list and for ensuring that all members of the ML have appropriate access control authorizations. The originator of a message directs the message to the MLA which then redistributes the message to the members of the ML. This process off loads the per-recipient processing from individual user agents and allows for more efficient management of large MLs.

212. **<u>Content Type and Message Identification</u>**

a. On origination, this element of service enables a user to indicate and on receipt for a user to verify the content type and message identification of a message protected by CSP. Also, the content type permits subtyping in support of fully automated processes.

b. Message Identification allows the originator to relate a signed receipt to a previously originated message. The message identifier (e.g. the IPMIdentifier in an ACP123 message) of the original message is used if it is present. If it is not present an identifier is assigned my the CSP UA.

213. **<u>Blind Copies</u>**

a. If a message system supports Blind Copy (BC), where a recipient is unaware of additional recipients of the message, then at least two messages must created. The first message has the primary and copy recipients. The other messages have the primary, copy and one blind copy recipient. These messages differ in the heading fields, but may contain the same message identifier. Because the messages differ, they have different signature values if they are signed.

b. For proof of content delivery (signed receipts), the correlation of receipts with message to provide validation is done with the Signed Content IDentifier (SCID) which is defined for P772 message to be the IPMIdentifier. If the originator requests proof of content delivery from all of the recipients, then the originator must store all of the signature values associated with the message, and the originator's CSP UA must assign message identifiers for each message and message signature to relate the signed receipt to the original message signature.

Chapter 3

Procedures For The Common Security Protocol

301. Overview

a. This Chapter specifies the structure of CSP and the procedures for processing CSP for each of the security services described in Chapter 2.

b. The CSP heading, i.e., all the CSP except for the encapsulated content, shall be encoded using the Distinguished Encoding Rules. The algorithms parameters shall use EXPLICIT tagging.

SECTION II

Procedures for Content Confidentiality and Integrity with Authenticated Key Distribution and Access Control (encryption and key distribution)

302. CSP with Content Confidentiality and Integrity

The structure of CSP is as follows when only content confidentiality and integrity are selected.

```
Csp   ::=    SEQUENCE {
          originatorSecurityData        [0]  OriginatorSecurityData OPTIONAL,
          recipientSecurityData         [2]  SEQUENCE  (SIZE (1..ub-recipients)) OF PerRecipientToken
                                                      OPTIONAL,
          encapsulatedContent           [8]  OCTET STRING (SIZE (1..ub-content-length))
                                                      OPTIONAL  }
```

303. OriginatorSecurityData

303.a Procedures

(1) **OriginatorSecurityData** is a sequence of security information pertaining to the originator of the message. It contains the **keyManagementCertificate**, **messageSecurityData**, **confidentialityAlgorithm**, **integrityAlgorithm**, **msdProtectionAlgorithm**, and **tokenProtectionAlgorithm**. If access control is required, then **messageSecurityData** and **msdProtectionAlgorithm** must both be present in **orginatorSecurityData**. If access control is not required, then **messageSecurityData** may be absent. If **messageSecurityData** is absent, then **msdProtectionAlgorithm** must also be absent.

(2) The **keyManagementCertificate** is an X.509 certification path which establishes the binding between the originator's Subject Name and the originator's key management public cryptographic material, and KMID. The **keyManagementCertificate** provides each recipient with information necessary to process his **PerRecipientToken**.

```
OriginatorSecurityData  ::=  SEQUENCE  {
          confidentialityAlgorithm                      AlgorithmIdentifier,
          integrityAlgorithm                            AlgorithmIdentifier,
          tokenProtectionAlgorithm                      AlgorithmIdentifier,
          msdProtectionAlgorithm                        AlgorithmIdentifier OPTIONAL,
          keyManagementCertificate        [0]           CertificationPath OPTIONAL,
          messageSecurityData             [4]           ProtectedMessageSecurityData OPTIONAL}
```

303.b **ProtectedMessageSecurityData**

(1) **ProtectedMessageSecurityData** contains the **SecurityLabel** of the message, and optionally the structure **AttributeCertificationPath**, which includes the originator's **AttributeCertificate.** The CSP UA encrypts **MessageSecurityData** to form **ProtectedMessageSecurityData.** The CSP UA uses the **msgKey** and the **msdProtectionAlgorithm** to encrypt **MessageSecurityData.**

(2) When the originator's **AttributeCertificates** are associated with the user's X.509 key management certificate, then these attribute certificates are included in **kmAttrCerts.**

```
ProtectedMessageSecurityData  ::=  OCTET STRING
        -- Protected form of MessageSecurityData, encrypted using msgKey

MessageSecurityData  ::=  SEQUENCE  {
        label           SecurityLabel,
        kmAttrCerts     SEQUENCE OF AttributeCertificationPath OPTIONAL  }
```

303.c **SecurityLabel**

The **SecurityLabel** contains information regarding the classification of the original content that is protected by the CSP encapsulation. This definition is imported 1988 CCITT Recommendation X.411.

303.d **AttributeCertificationPath**

AttributeCertificationPath contains the **AttributeCertificate** of the originator, and a sequence of **AttributeCertificates** and **Certificates** from the authorities that issued the **AttributeCertificate** to the originator.

```
-- Imported from the Draft 1997 X.509 Recommendation Definitions

        AttributeCertificate, AttributeCertificationPath, ACPathData
                FROM AuthenticationFramework {joint-iso-ccitt ds(5) module(1) authentication-
                        framework(7) 3}
```

304. **PerRecipientToken**

304.a **Procedure**

Each **PerRecipientToken** contains information for an intended recipient of the message. There is one of these for each recipient, and it has two parts, the **Tag** and the **ProtectedRecipientKeyToken.** The **tag** is clear text and contains identifying information each recipient uses to select and process his **PerRecipientToken.** The **ProtectedRecipientKeyToken** is an **OCTET STRING**, the value of which the originator encrypted in a key that only the intended recipient can obtain. If an MLA is processing a message then the **ProtectedRecipientKeyToken** value is the result of the MLA encrypting the **RecipientKeyToken** with a pairwise key or ML Key

```
PerRecipientToken  ::=  SEQUENCE  {
        tag                             Tag,
        protectedRecipientKeyToken      ProtectedRecipientKeyToken  }
```

304.b Tag

The Tag is clear text and contains identifying information each recipient uses to select and process his **PerRecipientToken.** The Tag may either be a **PairwiseTag**, **MLTag** or **KeyTransferTag.** The recipient uses **PairwiseTag** to identify a token protected with a pairwise key, and uses **MLTag** to identify a token protected with a **MLKey**. The recipient uses the **KeyTransferTag** to identify another form of pariwise key, where a symmetric key is conveyed from the originator to each recipient by using reversible public key algorithms.

```
Tag ::= CHOICE {
        pairwise        PairwiseTag,
        mlTag           [0]     MLTag,
        keyTransfer     [1]     KeyTransferTag}
```

304.c PairwiseTag

The **PairwiseTag** is used when a key agreement technique of key distribution is used. This technique allows both the originator and each recipient to calculate a secret symmetric key. The **PairwiseTag** is generated by the Originator CSP process and contains information that the recipient uses to identify which of its posted certificates and UKMs the Originator CSP UA used to protect the token. The **kmid** and the **edition** identify the certificate, and the **date**, if present, identifies the recipient's UKM.

```
PairwiseTag ::= SEQUENCE {
        kmid                    Kmid,
        edition                 INTEGER (1..ub-edition-size),
        date                    UTCTime OPTIONAL }
```

304.d Kmid

The **Kmid** is a unique identifier associated with the user's key management public cryptographic material. The KMID is the subject key identifier extension contained in the Version 3 X.509 Certificate. If this extension is not present, then a hash of the subject public key may be used instead. The hash algorithm used for the creation of the KMID is implied by the **tokenProtectionAlgorithm.**

```
Kmid ::= OCTET STRING
```

304.e MLTag

(1) If an MLA is not using pairwise keys, then it protects the originator's **RecipientKeyToken** using a single Mail List Key (MLK). After protecting it with the MLK, the MLA includes this token in the **ProtectedRecipientKeyToken.** For the remainder of this document, when the token is protected with an MLK, the token is referred to as the **mlKeyToken.** The Tag associated with the **mlKeyToken** is an **mlTag.**

(2) **MLTag** is generated by an MLA and contains an **mlid** that is the **Kmid** of MLA, and an **mlKeyDate** is the date the MLA performed redistribution.

```
MLTag ::= SEQUENCE {
        mlid                    Kmid,
        mlKeyDate               UTCTime }
```

304.f **KeyTransferTag**

The **KeyTransferTag** is used when a symmetric key is conveyed from the originator to each recipient by using reversible public key algorithms. The originator CSP UA generates the **KeyTransferTag** and it contains the information that the recipient uses to identify which of its posted certificates the originator CSP UA used to protect the token. The recipient CSP UA identifies the certificate used by the values of **issuer** and **serialNumber.** The **pkEncryptedTokenPrototectionKey** contains a symmetric key that the originator CSP UA has encrypted with the recipient's public key. When this format of tag is used, the CSP UA *must* apply CSP Encapsulation Integrity.

```
KeyTransferTag              ::= SEQUENCE {
        issuer                                  GeneralNames,
        serialNumber                            CertificateSerialNumber,
        pkEncryptedTokenProtectionKey           OCTET STRING  }

CertificateSerialNumber                         INTEGER
```

304.g **ProtectedRecipientKeyToken and RecipientKeyToken**

The **ProtectedRecipientKeyToken** is the protected form of **RecipientKeyToken.** The **RecipientKeyToken** contains the **msgKey** that is used to protect the message. The **msgHash** is calculated over the (original unprotected text) message content. The **signatureBlockIndicator** is set to true if a **signatureBlock** is present. The **encapsulatedContentType** is the Content Type of the message that is protected by CSP. If the **msdHash** is present, then the **OriginatorSecurityData MessageSecurityData** component must also be present and the **msdHash** is calculated over the **MessageSecurityData.** If the **msdHash** is absent, then the **OriginatorSecurityData MessageSecurityData** component must also be absent. If this check fails, then a security error has occurred and the CSP UA shall reject the CSP message before decrypting the encapsulated content.

```
ProtectedRecipientKeyToken     ::= OCTET STRING

        -- Protected form of RecipientKeyToken

RecipientKeyToken  ::=  SEQUENCE  {
        msgKey                          OCTET STRING,
        msgHash                         OCTET STRING,
        signatureBlockIndicator         BOOLEAN,
        encapsulatedContentType         CspContentType,
        msdHash                         OCTET STRING OPTIONAL}
```

305. **Generating a CSP Message**

305.a **Security Service Selection**

(1) The CSP UA processes a message content that is accompanied, implicitly or explicitly, by submission envelope information. In some UA-MTA configurations 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, the CSP UA uses the following values:

- originator-name
- recipient-names
- content-type
- content

(2) The CSP UA determines message-security-label information in one of two ways: implicitly, based on local processing context, or explicitly, based on a message-security-label argument.

305.b **Access Control**

(1) Rule-based Access Control is optional in CSP. Access control is a determination made concerning the authorization of the originator to send and the recipient to receive the message. These access control decisions are based on the authorization information of the originator, the recipient, and the system on which the originator's user agent resides. The system on which the recipient's user agent resides cannot be included in this access control decision. The recipient may use more than one user agent, each located on a different system, each with different authorizations.

(2) User authorization information is determined from the users' key management certificates and **kmAttrCerts** while the system authorization is determined from local configuration data. Additional authorization information may be contained in the originator's (**kmAttrCerts**) and recipient's attribute certificates. If the originator's attribute certification path is required, then the CSP UA places the originator's attribute certificates in **AttributeCertificationPath.** Additional X.509 certificates and attribute certificates needed to verify the originator's attribute certificate may also be included in the **AttributeCertificationPath.** If applicable, the CSP UA includes the **security-categories** within the **SecurityLabel** to specify the sensitivity of the encapsulated content.

(3) Within **SecurityLabel**, the **PrivacyMark** is carried as provided by the originating UA, but it is not used in access control decisions by the CSP UA. A **security-policy-identifier** may be used to identify the security policy in force to which the **SecurityLabel** relates. If present, a **security-classification** may have one of a list of values. The basic **security-classification** hierarchy is defined in the X.411 specification, but the use of these values is defined by the **security-policy** in force.

305.c **Per Message Processing**

(1) The CSP UA calculates a hash value over the (original unprotected text) message content, and stores this value in **msgHash.** The CSP UA obtains a **msgKey**[1], and then uses it to encrypt the message content forming **encapsulatedContent**[2]. The message content is encrypted as a single string. The CSP UA constructs the **RecipientKeyToken.**

(2) Optionally, the CSP UA builds the **MessageSecurityData** structure including the **securityLabel.** The CSP UA calculates a cryptographic hash function over **MessageSecurityData**, which it places in **msdHash.** The CSP UA then encrypts the structure **MessageSecurityData**, using **msgKey**, to form **ProtectedMessageSecurityData.** The CSP UA indicates the integrity and confidentiality algorithms it used in **msdProtectionAlgorithm**, and places any parameters required by the algorithms there as well.

305.d **Per-recipient Processing**

(1) Establish a symmetric key to be used with the algorithm specified in the **tokenProtectionAlgorithm.** The symmetric key may be establised through key agreement techniques, key transfer techniques, or out-of-band distribution eechniques. If the algorithm requires, the CSP UA obtains the recipient's SIGNED UKM, which the recipient has posted to the directory. After validating the authenticity of the signature associated with the UKM, the CSP UA uses the UKM to form a pairwise key that only this intended recipient can derive. When forming the tag, the Date identifies the posted UKM that the CSP UA used to protect the **RecipientKeyToken.**

(2) The CSP UA encrypts the **RecipientKeyToken**, forming the **protectedRecipientKeyToken.** This along with the tag, which identifies the recipient's certificate through the KMID, forms the **PerRecipientToken.** The CSP UA forms one of these for each recipient.

(3) If the originator is not already an explicit recipient, then the CSP UA generates a **RecipientKeyToken** for the originator. This allows the originator to decrypt a message, in case it is returned for non-Delivery.

306. **Processing a CSP Message**

306.a **Security Service Determination**

The recipient requests the UA to accept delivery of a message from the MTA, or the UA retrieves a message from an MS. If the **ContentType,** an OBJECT IDENTIFIER, indicates CSP, then CSP processing commences. Next the CSP UA determines, by processing the CSP structure, that **OriginatorSecurityData** is present, and thus the originator invoked content confidentiality and integrity with authenticated key distribution services.

[1] How **msgKey** is obtained is out of the scope of this document.

[2] If confidentiality is not invoked then the message content becomes **encapsulatedContent**.

306.b **Token Selection**

(1) The CSP UA selects the correct **PerRecipientToken** by first examining each **PerRecipientToken** with a **PairwiseTag** until it identifies a **PairwiseTag** that contains the CSP UA's **Kmid**. When **date** is present in the **PairwiseTag**, the recipient may use it to discover which previously posted UKM was used by the originator.

(2) If no **PerRecipientToken** is identified, the CSP UA then first examines each **KeyTransferTag** based on the **IssuerName** and **CertificateSerialNumber** of the CSP UA. If no **KeyTransferTag** is identified, the CSP UA then examines the **PerRecipientToken** with an **MLTag** until it identifies an **MLTag** that contains the **Kmid** of an ML of which it is a member.

(3) If **OriginatorSecurityData** is present and a correct **PerRecipientToken** is not found then a security error has occurred.

(4) Once the correct token is identified, the CSP UA decrypts the token. The CSP UA then checks if **signatureBlockIndicator** is true. If it is true and the **SignatureBlock** is not present, then a security error has occurred.

(5) If **msdHash** is present, then the CSP UA uses the **msgKey** to decrypt **ProtectedMessageSecurityData.** The CSP UA then calculates a hash using **MessageSecurityData** and compares the result of the hash calculation to **msdHash**. If the comparison fails, then a security error has occurred. If either **msdHash** is present, and **ProtectedMessageSecurityData** is not present or **msdHash** is not present and **ProtectedMessageSecurityData** is present, then a security error has occurred.

306.c **Rule-Based Access Control**

If **MessageSecurityData** is present, then the CSP UA enforces rule based-access control, which is a determination made concerning the authorization of the originator and recipient to process and receive the message. These authorizations are based on the authorization information of the originator (obtained from **keyManagementCertificate** and **kmAttrCert**), the recipient and the recipient's end system. User authorizations represent a maximum clearance, while the end system authorization represents a range. The CSP UA checks that both the originator and recipient have the required authorizations and the **securityLabel** is dominated by the high end of the range of the recipient's end system. If the message security level is below the minimum of the range of the recipient's end system, the CSP UA upgrades the message to this minimum security level prior to delivering it to the user.

306.d **Message Processing**

From the **RecipientKeyToken**, the CSP UA obtains the **msgKey** and then decrypts the message. The CSP UA calculates a hash value as a function of the content, and if the value is equal to the **msgHash**, the content integrity is verified. Once the recipient UA has access to the **msgKey**, it can decrypt the security label if it is present. The recipient UA has access to the unencrypted content of the message by virtue of possession of msgKey. If the access control check subsequently fails, the user could then have unauthorized access to the unencrypted message content. For this reason, the access control functionality and the message decryption functionality must remain separate processes.

SECTION III

Procedures for Proof of Content Origin (message signature)

307. CSP with Proof of Content Origin

The structure of CSP is as follows when only Proof of Content Origin is selected.

```
Csp   ::=   SEQUENCE {
        signatureBlock          [1]  SignatureBlock OPTIONAL,
        encapsulatedContent     [8]  OCTET STRING (SIZE (1..ub-content-length))
                                                  OPTIONAL  }
```

308. SignatureBlock

The **SignatureBlock** contains information used to provide proof of content origin. The CSP UA calculates a digital signature, which consists of signing a hash using the originator's private cryptographic signature material. Whenever the CSP UA applies confidentiality to the message (i.e. encrypts the message), it may apply confidentiality to the signature value by using **encSigData** for **signatureValue**. If **AttributeCertificates** are associated with the user's X.509 signature certificate, then these attribute certificates are included in **sigAttrCerts**.

```
SignatureBlock  ::= SEQUENCE  {
        signatureAlgorithm                    AlgorithmIdentifier,
        signatureValue                        SignatureValue,
        controlInformation                    ControlInformation,
        signatureCertificate        [0]       CertificationPath OPTIONAL,
        sigAttrCerts                [1]       SEQUENCE OF AttributeCertificationPath
                                                      OPTIONAL  }

SignatureValue   ::= CHOICE  {
        sigValue                SigValue,
        encSigData              EncSigData  }

EncSigData   ::= SEQUENCE  {
        encSigAlgorithm         AlgorithmIdentifier,
        encSigValue             ProtectedSigValue  }

ProtectedSigValue  :=  OCTET STRING
        --Protected form of SigValue

SigValue  ::=  OCTET STRING
```

308.a ControlInformation

ControlInformation is **SignatureInformation** that contains additional information from the originator including receipt requests.

```
ControlInformation ::= CHOICE  {
        signatureInformation        [0]  SignatureInformation }
```

308.b SignatureInformation

(1) The **encapsulatedContentType** specifies the **CspContentType** of the original message. The **signedContentIdentifier** must be generated by the originator of the message. The original UA should generate the identifier. In the case that the original content is either P22 (Interpersonal Message) or P772 (Military Message Handling Structure) then this value should be IPMIdentifier, and should always be present on signed messages.

306.b (Continued)

(2) If the original UA cannot generate a content identifier, then the **signedContentIdentifier**, which is specified as an OCTET STRING must be globally unique. To assure its global uniqueness, this OCTET STRING must contain the concatenation of the KMID (see paragraph 302.3 with respect to the KMID) from the key management public cryptographic material or the public signature keying material, followed by a thirteen octet **UTCTime** with a value of yymmddhhmmssZ, followed by a random number.

(3) The originator may request a signed receipt by placing the **receiptRequest** in **SignatureInformation.** The **receiptRequests** indicates from which recipients the originator requests signed receipts. The originator retains the **signedContentIdentifier** along with the **receiptRequests**, the hash that was used to compute the **signatureValue**, and the **signatureValue** in a local database. The database is used for later processing of signed receipts.

(4) The originator uses the **receiptsTo** field to identify the users to whom the receiving UA should send signed receipts. It is used only if signed receipts are to be sent to users other than, or in addition to, the originator. If the **receiptsTo** field is used to designate recipients in addition to the originator, then the originator's **GeneralNames** must be included in the **receiptsTo** list. Originators should always use the **newReceiptsTo** form of **receiptsTo.** The **oldReceiptsTo** form of **receiptsTo** is included to allow the CSP UA to validate signatures on messages created with an earlier protocol.

```
SignatureInformation  ::= SEQUENCE  {
        encapsulatedContentType              CspContentType OPTIONAL,
        signedContentIdentifier              OCTET STRING,
        receiptRequests                      ReceiptsIndicator,
        receiptsTo                           ReceiptsTo OPTIONAL  }

ReceiptsTo  ::= CHOICE  {
        newReceiptsTo               [0]      SEQUENCE (SIZE (1..ub-receiptsTo)) OF
                                                     GeneralNames,
        oldReceiptsTo                        SEQUENCE (SIZE (1..ub-receiptsTo)) OF
                                                     ORName }
```

308.c **CspContentType**

(1) **CspContentType** specifies the content type of the original message. Its structure is based on to that contained within X.411 Abstract Services:

```
        MTSAbstractService {joint-iso-ccitt mhs-motis(6) mts(3)
                modules(0) mts-abstract-service(1)}
```

(2) But, this definition includes a third syntax, **externalWithSubtype**, which has been added to support the automatic processing of specialized messages.

```
CspContentType  ::=  CHOICE {
        built-in                   BuiltinContentType,
        external                   ExternalContentType,
        externalWithSubtype        ExternalContentWithSubtype  }

BuiltinContentType  ::=  [APPLICATION 6] INTEGER  {
        unidentified  (0),
        external  (1),
        interpersonal-messaging-1984  (2),
        interpersonal-messaging-1988  (22)  }  (0..ub-built-in-content-type)

ExternalContentType  ::=  OBJECT IDENTIFIER

ExternalContentWithSubtype  ::=  SEQUENCE  {
        external              ExternalContentType,
        subtype               INTEGER }
```

308.d **ReceiptsIndicator**

(1) **ReceiptsIndicator** allows the Originator to specify either a list of recipients that should send signed receipts (**receiptList**), or an enumerated value (**allOrNone**).

(2) The value **receiptList** is a list of **GeneralNames. AllOrNone** can either be **noReceipt** (0), **allReceipts** (1), or **firstTierRecipients** (2).

(3) The value **noReceipt** means no recipient should send a receipt. The value **allReceipts** means all recipients should send receipts. The value **firstTierRecipients** means recipients with no information present in the **mlExpansionHistory** should send receipts; that is the recipient did not receive this message as a result of being a member of an ML.

```
ReceiptsIndicator ::= CHOICE {
        allOrNone               [0] AllOrNone,
        oldReceiptList          [1] SEQUENCE OF ORName,
        receiptList             [2] SEQUENCE OF GeneralNames }

AllOrNone ::= INTEGER {
        noReceipt               (0),
        allReceipts             (1),
        firstTierRecipients     (2) }
```

309. **Signature Generation**

a. Since content proof of origin[3] is invoked, the CSP UA forms a **signedContentIdentifier**, which it retains in a local database along with the hash value used to compute the **signatureValue**, the **signatureValue**, and **receiptRequests** associated with this message.

b. If content proof of delivery[4] is invoked also, the originator must request that either all recipients send receipts (**allOrNone** {1}), only Tier One recipients send receipts (**allOrNone** {2}), or a specific list of recipients should send receipts (**receiptList**).

c. Having completed the construction of **signatureInformation**, the CSP UA calculates a hash value, msgHash. The message hash value is calculated over the original unprotected data included within the **encapsulatedContent** OCTET STRING. The **msgHash** is calculated over the value bytes of the **encapsulatedContent** OCTET STRING; the tag and length bytes of the **encapsulatedContent** OCTET STRING are not included in the hash calculation. If confidentiality is invoked,the CSP UA places the **msgHash** in the RecipientKeyToken,. The CSP UA calculates a second hash value over the concatenation of the **msgHash** and **signatureInformation** (**ControlInformation** CHOICE including the context-specific tag [0]); i.e., a hash calculated over the value of **msgHash** prepended to **ControlInformation.** The CSP UA uses this second hash as the input to the signature algorithm, which the CSP UA uses to calculate the **signatureValue** associated with this message. Whenever the CSP UA applies confidentiality to the message (i.e., encrypts the message), it may apply confidentiality to the signature value (i.e., may encrypt the signature value).

[3] Proof of content delivery requires the originator to select proof of content origin.

[4] The value **allOrNone** is set to {0} if proof of content delivery is *not* invoked.

310. Signature Validation

If the **SignatureBlock** is present, the CSP UA calculates the hash value over the content. This hash is the **msgHash**, which has been previously compared to **msgHash** contained in the **RecipientKeyToken**, if confidentiality is invoked. A second hash value is calculated over the concatenation of the **msgHash** and **signatureInformation** (**ControlInformation** CHOICE including the context-specific tag [0]); i.e., a hash calculated over the value of **msgHash** prepended to **ControlInformation.** This second hash is used as the input to the signature algorithm, which the CSP UA uses to validate the **signatureValue** associated with this message. If the originator had applied confidentiality to the message then it may have also applied confidentiality to the **signatureValue.** In this case, the CSP UA must decrypt the signature value before it can validate the **signatureValue.**

SECTION IV

Procedures for Proof of Content Delivery (signed receipts)

311. CSP with Proof of Content Delivery

The structure of CSP is as follows when Proof of Content Delivery is selected.

```
Csp  ::=  SEQUENCE  {
          signatureBlock                    [1]  SignatureBlock OPTIONAL  }

SignatureBlock  ::=  SEQUENCE  {
          signatureAlgorithm                AlgorithmIdentifier,
          signatureValue                    SignatureValue,
          controlInformation                ControlInformation,
          signatureCertificate      [0]     CertificationPath OPTIONAL,
          sigAttrCerts              [1]     SEQUENCE OF AttributeCertificationPath
                                                    OPTIONAL  }

SignatureValue  ::= CHOICE  {
          sigValue             SigValue }

SigValue  ::=  OCTET STRING
```

312. ControlInformation

ControlInformation is **ReceiptInformation** that contains information from the recipient used to identify the message for which the recipient has returned the receipt.

```
ControlInformation  ::=  CHOICE  {
          receiptInformation        [1]  ReceiptInformation  }
```

313. ReceiptInformation

ReceiptInformation is included in the **SignatureBlock** generated by a recipient of a message in response to the originator's request for signed receipts. The **encapsulatedContentType** is the content type of the original message. The **signedContentIdentifier** is the identifier of the original message.

```
ReceiptInformation  ::= SEQUENCE  {
          encapsulatedContentType           CspContentType OPTIONAL,
          signedContentIdentifier                   OCTET STRING }
```

314. **Receipt Processing**

This section describes the procedures for the generation and validation of signed receipts. The originator of a message may request recipients to return signed receipts only if the originator has signed the message.

314.a **Hash Calculations for a Signed Receipt**

When the signature for a receipt is either generated by the recipient of a signed message that includes a request for a signed receipt, or validated by the originator of a signed message after receiving the signed receipt, the signature value of the original message is included in the hash calculation. The CSP UA obtains the **encapsulatedContentType**, **signedContentIdentifier**, and the **SigValue** from the original message, and places them into the **LocalReceiptInformation** structure; prepends the hash value used to verify **SigValue** to the **LocalReceiptInformation** structure, and then calculates the hash. **LocalReceiptInformation** is only used by an implementation; it is not contained within **CommonSecurityProtocol**.

```
LocalReceiptInformation ::= SEQUENCE  {
        encapsulatedContentType        CspContentType OPTIONAL,
        signedContentIdentifier        OCTET STRING,
        originatorSignatureValue       OCTET STRING  }
```

314.b **Signed Receipt Generation**

(1) If the recipient is generating a signed receipt, then a hash value is calculated using the hash value used to compute the **SigValue** of the original message prepended to a locally constructed structure (See Paragraph 310.1), and then signed. The **signatureAlgorithm** identifies the algorithm used to generate the digital signature including the hash function. The **signatureCertificate** is an X.509 certification path which establishes the binding between the recipient's Subject Name and the recipient's digital signature public cryptographic material.

(2) When the recipient requests the CSP UA to process a message, the CSP UA determines whether the originator has requested this recipient to return a signed receipt, by examining the structures **ReceiptsIndicator**, and **MlExpansionHistory** if it is present. If **MlExpansionHistory** is present, then the last occurrence of **MlData** is used in the following process. The following is an outline of a procedure for determining whether a signed receipt should be generated. An implementation shall be functionally equivalent to the external behavior resulting from this procedure.

– If **ReceiptsIndicator** contains the value allOrNone and it is equal to **noReceipt** (0), then the originator has not requested a signed receipt and the CSP UA terminates processing of a receipt for this message; i.e. the CSP UA does not generate a signed receipt.

– If the **MlExpansionHistory** is present and if **MlReceiptPolicy** contains the value none (**NULL**), then the receipt policy of the ML supersedes the originator’s request for a signed receipt, and the CSP UA terminates processing of a receipt for this message.

306.b (Continued)

– If **allOrNone** is equal to the **allReceipts** (1), then the CSP UA generates a signed receipt.

– If **ReceiptsIndicator** contains the value **allOrNone**, and it is equal to **firstTierRecipients** (2), and the **MlExpansionHistory** structure is not present, then the CSP UA generates a signed receipt. If **allOrNone** is equal to **firstTierRecipients** (2), and the **MlExpansionHistory** structure is present, then the CSP UA terminates processing of a receipt for this message.

– If **ReceiptsIndicator** contains the value **receiptList**, and the **receiptList** contains the **GeneralNames** of this recipient, then the CSP UA generates a signed receipt. If **ReceiptsIndicator** contains the value **receiptList** and the **receiptList** does not contain the **GeneralNames** of this recipient the CSP UA terminates processing of a receipt for this message.

(3) The CSP UA generates the **ReceiptInformation** structure, which does not include the **signatureValue** of the original message. Rather, when calculating hash for the receipt's **signatureValue**, the CSP UA includes the **SigValue** of the original message. The CSP UA calculates the hash as specified above in Paragraph 310.1. The receipt's **signatureValue** is the result of signature generation, which uses this hash value. When generating the signature for a signed receipt, the CSP UA always uses the unprotected form of **SignatureValue**, i.e. **SigValue.**

(4) The CSP UA constructs the structure **SignatureBlock** using **ReceiptInformation** (**ControlInformation** CHOICE including the context-specific tag [1]). The CSP UA constructs the sequence **CommonSecurityProtocol**, which contains the **SignatureBlock.**

(5) The CSP UA constructs a sequence of recipients for the signed receipts. If the value **receiptsTo** is not present, then the CSP UA places the originator of the original message in the sequence of recipients of the signed receipt. If the value **receiptsTo** is present, then the CSP UA uses the value of **receiptsTo** as the sequence of recipients of the signed receipt. If **MlExpansionHistory** is present and contains the value **insteadOf** the CSP UA replaces the sequence of recipients of the signed receipt with the value **insteadOf.** If **MlExpansionHistory** is present and contains the value **inAdditionTo**, the CSP UA adds the sequence of recipients contained in the value **inAdditionTo** to the sequence of recipients of the signed receipt.

(6) The ACP123 UA associated with the CSP UA creates an X.400 submission envelope for the results of the preceding processing. The ACP123 UA includes in this submission envelope the sequence **CommonSecurityProtocol**, which is either **Csp**, or **SIGNED Csp.** The CSP UA submits this envelope to the MTS for transfer and subsequent delivery.

SECTION V

Procedures for Integrity of CSP Encapsulation (sequence signature)

315. CSP with Integrity of CSP Encapsulation

a. The structure of CSP is as follows when Integrity of CSP Encapsulation is selected. The **signedCsp** choice is selected to provide Integrity of CSP.

```
CommonSecurityProtocol  ::= CHOICE  {
        signedCsp           [1] SIGNED Csp  }

Csp   ::=    SEQUENCE  {
        originatorSecurityData              [0]  OriginatorSecurityData OPTIONAL,
        signatureBlock                      [1]  SignatureBlock OPTIONAL,
        recipientSecurityData               [2]  SEQUENCE (SIZE (1..ub-recipients)) OF
                                                     PerRecipientToken OPTIONAL,
        contentDescription          [3]  TeletexString (SIZE (1..ub-subject-field))
                                                     OPTIONAL,
        mlExpansionHistory                  [10] MlExpansionHistory OPTIONAL,
        additionalSignatures                [13] SEQUENCE of AddSignature OPTIONAL,
        extensions                          [12] SEQUENCE OF Extension OPTIONAL,
        CSPSequenceSignatureCertificate
                                            [7]  CertificationPath OPTIONAL,
        cspSequenceSigAttrCerts             [14] SEQUENCE OF AttributeCertificationPath
                                                     OPTIONAL,
        encapsulatedContent                 [8]  OCTET STRING (SIZE (1..ub-content-length))
                                                     OPTIONAL }
```

b. Optionally, the CSP UA may sign **CommonSecurityProtocol** by calculating a one-way hash on it and then signing the resulting value. **signedCsp** contains the **Csp SEQUENCE**, the **algorithmIdentifier** indicating the signature algorithm used to sign **Csp** and the resulting **signatureValue.** If **Csp** is signed, **Csp** may contain the **cspSequenceSignatureCertificate CertificationPath** and/or **cspSequenceSignatureAttrCerts AttributeCertificationPath.** When **cspSequenceSignatureCertificate** is present, a CSP UA uses it to verify the **Csp SEQUENCE** signature value. When **cspSequenceSignatureCertificate** is absent, the **CertificationPath** contained in **signatureBlock** is used to verify the **Csp** SEQUENCE **signatureValue.**

SECTION VI

Procedures for Content Description (content description)

316. CSP with Content Description

a. The structure of CSP is as follows when Content Description is selected

```
Csp  ::=  SEQUENCE  {
        contentDescription          [3]  TeletexString OPTIONAL
                                                     (SIZE (1..ub-subject-field))  }
```

b. The **contentDescription** is generally used in combination with other componets of the **Csp** SEQUENCE. If local security policy permits, the CSP UA may place unprotected text data into **contentDescription.** This data may include the human readable IPM Subject field or priority and precedence information. There is no standardized mechanism defined in X.400, ACP123, or ACP120 to process such human readable priority and precedence information.

SECTION VII

Procedures for Mail List Expansion History (ML Expansion History)

317. CSP with Mail List Expansion History

The structure of CSP is as follows when Mail List Expansion History is selected. When **MLExpansionHistory** is present, some of the components may be present and others absent depending on the services selected for the message sent to the mail list. For example, the **signatureBlock** will be present only if the message sent to the mail list was signed.

```
SIGNED Csp  ::=  SEQUENCE  {
        originatorSecurityData              [0]  OriginatorSecurityData OPTIONAL,
        signatureBlock                      [1]  SignatureBlock OPTIONAL,
        recipientSecurityData               [2]  SEQUENCE (SIZE (1..ub-recipients)) OF
                                                      PerRecipientToken OPTIONAL,
        mlExpansionHistory                  [10]  MlExpansionHistory OPTIONAL,
        cspSequenceSignatureCertificate     [7]  CertificationPath OPTIONAL,
        cspSequenceSigAttrCerts             [14]  SEQUENCE OF AttributeCertificationPath
                                                      OPTIONAL,
        encapsulatedContent                 [8]  OCTET STRING (SIZE (1..ub-content-length))
                                                      OPTIONAL  }
```

318. MlExpansionHistory

The **mlExpansionHistory** is updated each time a message is distributed to members of an ML by an MLA. The history of MLs enables an MLA to detect a loop in the sequence of ML expansions. If the expansion, **mlExpansionHistory**, is absent, a recipient discovers that he is a first tier recipient.

```
MlExpansionHistory          ::=  SEQUENCE OF MlData
```

319. MlData

MlData contains the expansion history of an MLA that has processed the message. As an MLA distributes a message to members of an ML, the MLA records its **mlid**, date and time of expansion, and its receipt policy in an **MLData** and appends it to the **mlExpansionHistory.**

```
MlData ::=  SEQUENCE  {
        mlid                    Kmid,
        expansionTime           UTCTime,
        mLReceiptPolicy         MLReceiptPolicy OPTIONAL  }
```

320. MLReceiptPolicy

a. The receipt policy of the ML can withdraw the originator’s request for the return of a signed receipt. However, if the originator of the message has not requested a signed receipt, the MLA cannot request a signed receipt.

320. (Continued)

b. When present, the **MLReceiptPolicy** specifies a receipt policy which supersedes the originator's request for signed receipts. The policy can be one of three possibilities. An ML can specify that receipts should not be returned (**none**). An ML can specify that receipts should be returned to an alternate list of recipients, instead of to the originator (**insteadOf**). An ML can specify that receipts should be returned to a list of recipients in addition to the originator (**inAdditionTo**).

```
MLReceiptPolicy   ::=       CHOICE {
        none                        [0] NULL,
        insteadOf                   [1] SEQUENCE (SIZE (1..ub-insteadOf)) OF GeneralNames,
        inAdditionTo                [2] SEQUENCE (SIZE (1..ub-inAdditionTo)) OF GeneralNames  }
```

321. <u>**Mail List Agent Processing**</u>

a. Distributing a message to members of an ML upon receipt of a message, the MLA verifies the CSP sequence signature, if present. If the signature verification fails, then a security error has occurred.

b. The MLA locates its **PerRecipientToken** and removes all other **PerRecipientTokens**. The MLA decrypts this token and obtains the **msgKey**. If **messageSecurityData** is present, the MLA uses the **msgKey** to decrypt the **protectedMessageSecurityData** to obtain the **SecurityLabel** and it determines that the label for this message is within the intersection of originator's and ML's authorizations. The MLA also determines that the message security label is within the intersection of the ML’s and ML recipients’ authorizations. This access check is performed even in the event that an **mlKey** is used to protect the token in the CSP message distributed to the ML.

c. The MLA then determines if **pairwise** keys are required for this ML. If so, the MLA constructs **PerRecipientTokens**. If the ML uses **mlKey**, then the MLA constructs an **mlKeyToken**.

d. If the MLA is using pairwise keys to protect the **PerRecipientToken**, then the MLA replaces the **keyManagementCertificate** with the MLA's certification path. The MLA may also replace **confidentialityAlgorithm** or **integrityAlgorithm**, **tokenProtectionAlgorithm**, or **msdProtectionAlgorithm**, if necessary. Within **MessageSecurityData**, the MLA changes **kmAttrCerts** to the MLA’s value, but leaves the value label unchanged.

e. The MLA updates (or creates if it is the first MLA) **mlExpansionHistory** and conveys its receipt policy. The MLA must always sign the **Csp**. After constructing the sequence **Csp**, the MLA applies a digital signature to **Csp**.

SECTION VIII

CSP Heading Construction

322. CSP Heading Construction

a. The CSP UA now has sufficient information to encode the **Csp** structure including the: **originatorSecurityData**, **signatureBlock**, **recipientSecurityData**, **contentDescription**, **additionalSignatures**, **extensions**, and **encapsulatedContent.** Once the CSP UA has encoded these structures, the CSP UA forms **Csp**. Depending on the site policy, the CSP UA may then sign the complete protected content, **Csp** (CSP heading and protected content). Before the CSP UA signs **Csp**, and if the CSP UA has not placed its signature certificate or attribute certificate in the **signatureBlock**, then the CSP UA places its signature certificate into **cspSequenceSignatureCertificate** and places its attribute certificate into **cspSequenceSignatureAttrCerts.** The CSP UA then signs **Csp** by calculating a hash value over the **Csp** sequence, and uses this result to calculate a signature value that it appends to the **Csp** sequence, forming **SIGNED Csp.**

b. The ACP 123 UA associated with the CSP UA creates an X.400 submission envelope for the results of the preceding processing, and based on information obtained from the original submission envelope, which accompanied the message content. The ACP 123 UA includes in this submission envelope the sequence **CommonSecurityProtocol**, which is either **Csp** or **SIGNED Csp.** The ACP 123 UA submits this envelope to the MTS for transfer and delivery.

SECTION IX

Procedures for Additional Signature Processing

323. AddSignature

a. **AddSignature** is placed in the CSP heading by the originator to demonstrate that other individuals agree with the encapsulated content, to demonstrate that the encapsulated content was generated before a time provided by a trusted third party, or both. Additionally, the originator of a message may place **AddSignature** in the CSP heading to accomplish interoperability of digital signatures with non-CSP environments; e.g., NATO (See ACP120 Supplement 2).

b. After message delivery, **AddSignature** may be used by the originator, or any recipient, to store additional signatures or time stamped signatures associated with the message content, or a signed receipt. The **addSigAlgorithm** identifies the algorithm used to generate the digital signature including the hash function. The **addSigCertificate** is an X.509 certification path which establishes the binding between the third party's Subject Name and that party's digital signature public cryptographic material. To compute an additional signature the third party calculates a digital signature by signing a hash using the third party's signature cryptographic material.

```
AddSignature ::= SEQUENCE {
        addSigAlgorithm          AlgorithmIdentifier,
        addSigValue              OCTET STRING,
        addSigControlInfo        AddSigControlInfo,
        addSigCertificate        CertificationPath,
        addAttrCertificates      SEQUENCE OF AttributeCertificationPath
                                         OPTIONAL }
```

323.a **AddSigControlInfo**

AddSigControlInfo contains an identifier assigned by the third party, **addSigData**, and optionally a time stamp. If the originator of the message is also generating **AddSignature**, then **addSigIdentifier** is omitted.

```
AddSigControlInfo  ::=  SEQUENCE  {
          addSigIdentifier               OCTET STRING OPTIONAL,
          addSigData                     AddSigData,
          timeStampData                  TimeStampData OPTIONAL  }
```

323.b **AddSigData**

(1) **AddSigData** contains information from the third party to identify the signed message content, message content and signature block, or receipt.

(2) A message content is identified by the **signedContentIdentifier** assigned by the message originator.

(3) A message content and an associated signature block is identified by the combination of the **signedContentIdentifier** assigned by the message originator and the **GeneralNames** of the message originator.

(4) A receipt is identified by the combination of the **signedContentIdentifier** assigned by the message originator and the **GeneralNames** of the recipient that generated the receipt.

```
AddSigData  ::= CHOICE  {
          addSigOnContent                      [0]  ContentIdentifier,
          addSigOnContentAndSigBlock           [1]  ContentAndSigBlockIdentifier,
          addSigOnReciept                      [2]  ReceiptIdentifier  }

ContentIdentifier    ::=  OCTET STRING

ContentAndSigBlockIdentifier  ::=  SEQUENCE  {
          contentIdentifier              ContentIdentifier,
          originatorName                 GeneralNames  }

ReceiptIdentifier  ::=  SEQUENCE  {
          contentIdentifier              ContentIdentifier,
          recieptSenderName              GeneralNames  }
```

323.c **TimeStampData**

TimeStampData contains the time. **TimeStampData** may contain additional information which can be used by the CSP UA validating the signature to obtain additional confidence that the trusted third party is operating correctly.

```
TimeStampData  ::=  SEQUENCE  {
          timeStamp                            GeneralizedTime,
          addTimeStampData                     AddTimeStampData OPTIONAL  }

AddTimeStampData  ::=  SEQUENCE  {
          timeStampPolicy                      OBJECT IDENTIFIER,
          timeStampVerifyData                  OCTET STRING OPTIONAL  }
```

324. **Generating Additional Signatures on Message Content**

a. The CSP UA calculates a hash value over the (original unprotected text) message content, and stores this value in **msgHash**. As part of the proof of content origin processing, the CSP UA forms a **signedContentIdentifier**, computes second hash value over the concatenation of the **msgHash** and **signatureInformation**, calculates the **signatureValue**, and forms the **receiptRequests**. The CSP UA transfers the message content or the **msgHash** to the third party. The **signatureBlock** may also be transferred. The transfer may be accomplished by sending a CSP protected message to the third party or by other means. An additional signature is computed by signing a hash using the third party's signature cryptographic material.

b. To compute an additional signature over the message content, the third party may calculate a hash over the encapsulated content or use the **msgHash** calculated by the originator. Then, a second hash value is calculated over the concatenation of the **msgHash** and **addSigControlInfo**. The **addSigControlInfo** may include a trusted time stamp. Finally, the second hash value is used as the input to the signature algorithm, and the resulting signature value is included as the **addSigValue**.

c. To compute an additional signature over the message content and the unencrypted originator signature value, the third party may calculate a hash over the encapsulated content or use the **msgHash** calculated by the originator. Then, a second hash value is calculated over the concatenation of the **msgHash**, **signatureBlock**, and **addSigControlInfo**. The **addSigControlInfo** may include a trusted time stamp. Finally, the second hash is used as the input to the signature algorithm, and the resulting signature value is included as the **addSigValue**.

325. **Validating Additional Signatures on Message Content**

The CSP UA calculates a hash value over of the content; this value should be equal to the **msgHash**. The **contentIdentifier** in **addSigData** must match the **signedContentIdentifier** within the **signatureBlock**. If the **originatorName** is present, it must match the Subject Name in the **signatureCertificate**. A second hash value is calculated over the concatenation of the **msgHash**, optionally the **signatureBlock**, and **addSigControlInfo**. The **signatureBlock** is omitted if **addSigData** is the CHOICE **addSigOnContent**, and it is included if **addSigData** is the CHOICE **addSigOnContentAndSigBlock**. The **addSigControlInfo** may include a trusted time stamp. This second hash is used as the input to the signature algorithm, which the CSP UA uses to validate the **addSigValue**.

326. **Generating Additional Signatures on Receipt**

a. The CSP UA transfers to a third party the message content or the **msgHash**, and signed receipt associated with that message content. The transfer may be accomplished by sending an CSP protected message to the third party or by other means. An additional signature is computed by signing a hash using the third party's signature cryptographic material.

b. To compute an additional signature over a receipt, the third party may calculate a hash over the encapsulated content or use the **msgHash** calculated by the originator. Then, a second hash value is calculated over the concatenation of the **msgHash** and **ReceiptInformation** (this includes the context-specific tag [1] within the **ControlInformation** CHOICE). Then, a third hash value is calculated over the concatenation of the second hash value and **addSigControlInfo**. The **addSigControlInfo** may include a trusted time stamp. Finally, the third hash is used as the input to the signature algorithm, and the resulting signature value is included as the **addSigValue**.

327. Validating Additional Signatures on Receipt

The CSP UA calculates a hash value over of the content; this value should be equal to the **msgHash.** The **contentIdentifier** in **addSigData** must match the **signedContentIdentifier** within the **signatureBlock**, and the **receiptSenderName** must match the Subject Name in the **signatureCertificate.** A second hash value is calculated over the concatenation of the **msgHash**, and **receiptInformation** (this includes the context-specific tag [1] within the **ControlInformation** CHOICE). Then, a third hash value is calculated over the concatenation of the second hash value and **addSigControlInfo.** The **addSigControlInfo** may include a trusted time stamp. This third hash is used as the input to the signature algorithm, which the CSP UA uses to validate the **addSigValue.**

SECTION X

Procedures for Extensions

328. CSP with Extensions

a. The presence of an extension may impact interoperability with a recipient unable to parse the extension. While certain extensions may, in the future, be agreed to for widespread use, the following general guidelines for extensions are suggested:

(1) All non-critical extensions not recognized by the receiving CSP UA are ignored.

(2) Receipt of an CSP message with a critical extension not recognized by the receiving CSP UA results in terminating processing of the message.

(3) All extensions intended for use within a national or domain boundary should be removed at the exit gateway from that nation or domain. A domain is defined by the organization that registered the extension.

b. When an extension is registered, along with the its syntax, the elements of procedure for the originator and recipient must be supplied.

```
CspExtension  ::=  SEQUENCE {
        extnID                          EXTENSION.&id,
        critical                        BOOLEAN DEFAULT FALSE,
        extnValue                       OCTET STRING
                -- the value extnValue is the DER encoded EXTENSION.&ExtnType  }

EXTENSION  ::=  CLASS {
        &id        OBJECT IDENTIFIER,
        &ExtnType }
WITH SYNTAX  {
        SYNTAX              &ExtnType,
        IDENTIFIED BY       &id UNIQUE }
```

329. Construction of Extensions

If security policy requires extensions, the CSP UA shall include them in **Csp.** Depending on the site policy, the CSP UA may then sign the complete protected content, as defined in section 4 of this document. Any additional processing rules for an extension, must be specified at the time the extension is registered.

330. **Extension Processing**

The CSP UA checks to see if any **extensions** are present. For each **extension**, if the CSP UA recognizes the **extnID** of the **extension**, the CSP UA performs the procedures specified as part of the registration of the **extension**. If the CSP UA does not recognize the **extnID** and critical is FALSE, the CSP UA ignores the **extension**. If the CSP UA does not recognize the **extnID** and critical is TRUE, the CSP UA terminates processing of the message and a security error has occurred.

SECTION XI

Blind Carbon Copy

331. **CSP with Blind Carbon Copy**

When a Blind Carbon Copy of a message is signed and signed receipts are requested, then the **signedContentIdentifier** that is copied from the IPMIdentifier is prefixed with the IA5 string: BCC[n]<space>. The n is a positive integer that is the number of the copy being sent. This allows the CSP UA to examine the **signedContentIdentifier** and determine that the signed receipt is for a Blind Carbon Copy message and not the original message. This allows the CSP UA to find the information, without a number of trials that would be egregious for anyone awaiting the result.

Chapter 4

Definitive CSP ASN.1 Module

401. CSP ASN.1

```
CommonSecurityProtocol{id-infosec(joint-iso-ccitt 16 840 1 101 2 1)
                                        id-modules(0) id-csp(16)}

DEFINITIONS IMPLICIT TAGS  ::=


BEGIN

IMPORTS

-- X.411 Abstract Services

        ORName SecurityLabel
                FROM MTSAbstractService {joint-iso-ccitt mhs-motis(6) mts(3)
                        modules(0) mts-abstract-service(1)}

        ub-content-length, ub-recipients, ub-built-in-content-type
                FROM MTSUpperBounds {joint-iso-ccitt mhs-motis(6) mts(3)
                        modules(0) upper-bounds (3)}

-- X.420 Interpersonal Messaging System

        ub-subject-field, MessageParameters
                FROM IPMSUpperBounds {joint-iso-ccitt mhs-motis(6) ipms(1)
                        modules(0) upper-bounds (10)}

-- Directory definitions

        Certificate, CertificationPath, AlgorithmIdentifier, SIGNED, CertificateSerialNumber
                FROM AuthenticationFramework {joint-iso-ccitt ds(5)
                        module(1) authentication-framework(7) 2}

        GeneralNames
                FROM CERTIFICATEEXTENSIONS {joint-iso-ccitt ds(5) module(1)
                        certificateExtension(26) 0}

-- Draft 1997 X.509 Recommendation Definitions

        AttributeCertificate, AttributeCertificationPath, ACPathData
                FROM AuthenticationFramework {joint-iso-ccitt ds(5)
                        module(1) authentication-framework(7) 3} ;

CommonSecurityProtocol  ::=  CHOICE {
        csp                     [0] Csp,
        signedCsp               [1] SIGNED Csp }

Csp  ::=  SEQUENCE  {
        originatorSecurityData                  [0]  OriginatorSecurityData OPTIONAL,
        signatureBlock                          [1]  SignatureBlock OPTIONAL,
        recipientSecurityData                   [2]  SEQUENCE (SIZE (1..ub-recipients))
                                                        OF PerRecipientToken OPTIONAL,
        contentDescription              [3]  TeletexString (SIZE (1..ub-subject-field)) OPTIONAL,
        mlExpansionHistory                      [10] MlExpansionHistory OPTIONAL,
        additionalSignatures                    [13] SEQUENCE OF AddSignature OPTIONAL,
        cspExtensions                           [12] SEQUENCE OF CspExtension OPTIONAL,
        cspSequenceSignatureCertificate         [7]  CertificationPath OPTIONAL,
        cspSequenceSigAttrCerts                 [14] SEQUENCE OF AttributeCertificationPath
                                                        OPTIONAL,
        encapsulatedContent                     [8]  OCTET STRING (SIZE (1..ub-content-length))
                                                        OPTIONAL}
```

```
OriginatorSecurityData  ::=  SEQUENCE  {
            confidentialityAlgorithm                AlgorithmIdentifier,
            integrityAlgorithm                      AlgorithmIdentifier,
            tokenProtectionAlgorithm                AlgorithmIdentifier,
            msdProtectionAlgorithm                  AlgorithmIdentifier OPTIONAL,
            keyManagementCertificate                [0]  CertificationPath OPTIONAL,
            messageSecurityData                     [4]  ProtectedMessageSecurityData OPTIONAL  }

ProtectedMessageSecurityData  ::=  OCTET STRING
            -- Protected form of MessageSecurityData encrypted with msgKey

MessageSecurityData ::=  SEQUENCE  {
            label                   SecurityLabel,
            kmAttrCerts             SEQUENCE OF AttributeCertificationPath OPTIONAL  }

SignatureBlock  ::= SEQUENCE  {
            signatureAlgorithm              AlgorithmIdentifier,
            signatureValue                  SignatureValue,
            controlInformation              ControlInformation,
            signatureCertificate            [0]  CertificationPath OPTIONAL,
            sigAttrCerts                    [1]  SEQUENCE OF AttributeCertificationPath
                                                        OPTIONAL }

SignatureValue  ::= CHOICE  {
            sigValue                        SigValue,
            encSigData                      EncSigData  }

EncSigData  ::=  SEQUENCE  {
            encSigAlgorithm                 AlgorithmIdentifier,
            encSigValue                     ProtectedSigValue  }

ProtectedSigValue  ::= OCTET STRING
            --Protected form of SigValue

SigValue  ::=  OCTET STRING

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

SignatureInformation  ::=  SEQUENCE  {
            encapsulatedContentType         CspContentType OPTIONAL,
            signedContentIdentifier         OCTET STRING,
            receiptRequests                 ReceiptsIndicator,
            receiptsTo                      ReceiptsTo  OPTIONAL  }

ReceiptsTo  ::=  CHOICE  {
            newReceiptsTo                   [0]  SEQUENCE SIZE (1..ub-receiptsTo) OF GeneralNames,
            oldReceiptsTo                   SEQUENCE (SIZE (1..ub-receiptsTo)) OF ORName  }

CspContentType  ::=  CHOICE {
            built-in                        BuiltinContentType,
            external                        ExternalContentType,
            externalWithSubtype             ExternalContentWithSubtype  }

BuiltinContentType  ::=  [APPLICATION 6] INTEGER  {
            unidentified (0),
            external (1),
            interpersonal-messaging-1984 (2),
            interpersonal-messaging-1988 (22)  }  (0..ub-built-in-content-type)

ExternalContentType  ::=  OBJECT IDENTIFIER

ExternalContentWithSubtype  ::=  SEQUENCE  {
            external                ExternalContentType,
            subtype                 INTEGER  }

ReceiptsIndicator  ::=  CHOICE  {
            allOrNone               [0]  AllOrNone,
            oldReceiptList          [1]  SEQUENCE OF ORName,
            receiptList             [2]  SEQUENCE OF GeneralNames  }
```

```
AllOrNone  ::=  INTEGER  {
            noReceipt             (0),
            allReceipts           (1),
            firstTierRecipients (2)  }

ReceiptInformation  ::=  SEQUENCE  {
            encapsulatedContentType              CspContentType OPTIONAL,
            signedContentIdentifier              OCTET STRING }

LocalReceiptInformation  ::=  SEQUENCE  {
            encapsulatedContentType              CspContentType OPTIONAL,
            signedContentIdentifier              OCTET STRING,
            originatorSignatureValue             OCTET STRING }

PerRecipientToken   ::=  SEQUENCE  {
            tag                                  Tag,
            protectedRecipientKeyToken           ProtectedRecipientKeyToken  }

Tag  ::=  CHOICE {
            pairwise                   PairwiseTag,
            mlTag                      [0] MLTag,
            keyTransfer                [1] KeyTransferTag  }

PairwiseTag  ::= SEQUENCE  {
            kmid                       Kmid,
            edition                    INTEGER (1..ub-edition-size),
            date                       UTCTime OPTIONAL  }

MLTag  ::=  SEQUENCE  {
            mlid                       Kmid,
            mlKeyDate                  UTCTime  }

KeyTransferTag  ::=  SEQUENCE  {
            issuer                               GeneralNames,
            serialNumber                         CertificateSerialNumber,
            pkEncryptedTokenProtectionKey        OCTET STRING  }

Kmid  ::=  OCTET STRING

ProtectedRecipientKeyToken  ::=  OCTET STRING
            -- Protected form of RecipientKeyToken

RecipientKeyToken  ::=  SEQUENCE  {
            msgKey                     OCTET STRING,
            msgHash           OCTET STRING,
            signatureBlockIndicator    BOOLEAN,
            encapsulatedContentType    CspContentType,
            msdHash           OCTET STRING OPTIONAL  }

MlExpansionHistory  ::=  SEQUENCE OF MlData

MlData  ::=  SEQUENCE  {
            mlid                       Kmid,
            expansionTime              UTCTime,
            mLReceiptPolicy            MLReceiptPolicy OPTIONAL  }

MLReceiptPolicy  ::=  CHOICE  {
            none                       [0]  NULL,
            insteadOf                  [1]  SEQUENCE (SIZE (1..ub-insteadOf)) OF
                                                 GeneralNames,
            inAdditionTo               [2]  SEQUENCE (SIZE (1..ub-inAdditionTo)) OF
                                                 GeneralNames  }

AddSignature  ::=  SEQUENCE  {
            addSigAlgorithm            AlgorithmIdentifier,
            addSigValue                OCTET STRING,
            addSigControlInfo          AddSigControlInfo,
            addSigCertificate          CertificationPath,
            addAttrCertificates SEQUENCE OF AttributeCertificationPath
                                                 OPTIONAL }
```

```
AddSigControlInfo  ::=  SEQUENCE  {
          addSigIdentifier              OCTET STRING OPTIONAL,
          addSigData                    AddSigData,
          timeStampData                 TimeStampData OPTIONAL  }

AddSigData  ::=  CHOICE  {
          addSigOnContent                         [0]  ContentIdentifier,
          addSigOnContentAndSigBlock              [1]  ContentAndSigBlockIdentifier,
          addSigOnReceipt                         [2]  ReceiptIdentifier  }

ContentIdentifier  ::=  OCTET STRING

ContentAndSigBlockIdentifier  ::=  SEQUENCE  {
          contentIdentifier             ContentIdentifier,
          originatorName                GeneralNames  }

ReceiptIdentifier  ::=  SEQUENCE  {
          contentIdentifier             ContentIdentifier,
          receiptSenderName             GeneralNames  }

TimeStampData  ::=  SEQUENCE  {
          timeStamp                     GeneralizedTime,
          addTimeStampData              AddTimeStampData OPTIONAL  }

AddTimeStampData  ::=  SEQUENCE  {
          timeStampPolicy               OBJECT IDENTIFIER,
          timeStampVerifyData           OCTET STRING OPTIONAL  }

CspExtension  ::=  SEQUENCE  {
          extnID              EXTENSION.&id,
          critical            BOOLEAN DEFAULT FALSE,
          extnValue           OCTET STRING  }
                    -- the value extnValue is the DER encoded EXTENSION.&ExtnType

EXTENSION  ::=  CLASS {
          &id                 OBJECT IDENTIFIER,
                              &ExtnType  }
WITH SYNTAX  {
          SYNTAX              &ExtnType,
          IDENTIFIED BY       &id UNIQUE  }

forwarded-csp-message-body-part       EXTENDED-BODY-PART-TYPE
          PARAMETERS          CSPMessageParameters
                              IDENTIFIED BY csp-forwarded-message-parameters-id
          DATA                CommonSecurityProtocol
                    ::=       forwarded-csp-msg-body-part

CSPMessageParameters  ::=  MessageParameters

-- Upper Bounds

ub-receiptsTo  ::=  16
ub-edition-size  ::=  2
ub-insteadOf  ::=  16
ub-inAdditionTo  ::=  16

-- OIDS

ID  ::= OBJECT IDENTIFIER

id-infosec                              ID  ::= { 2 16 840 1 101 2 1 }
id-modules                              ID  ::= { id-infosec 0 }
id-formats                              ID  ::= { id-infosec 2 }
id-csp                                  ID  ::= { id-modules 16 }
forwarded-csp-msg-body-part             ID  ::= { id-formats 74 }
csp-forwarded-message-parameters-id     ID  ::= { id-formats 75 }

END  -- of CSP
```

Chapter 5

Error Conditions

501. Originator Error Conditions

On each originator error condition, processing is suspended and the originator is notified of the error, and additional processing and procedures are subject to national policy.

501.a Access Control Failure

One of the recipients is not authorized to receive this message because the rule based access control check has failed.

501.b Recipient Certificate Validation Failure

The certificate of an intended recipient is not validated successfully, because the certificate has expired, the certificate (or any certificate in its certificate path) has been revoked, the certificate (or any certificate in its certificate path) has been compromised, or the signature of the certificate (or any certificate in its certificate path) has failed. The entire message is not processed until a valid certificate is obtained.

502. Recipient Error Conditions

On each recipient error condition, processing is suspended and the recipient is notified of the error, and additional processing and procedures are subject to national policy.

502.a Access Control Failure

This is caused by failure on the verification of the **msdHash**, or the **msdHash** being absent when **ProtectedMessageSecurityData** is present, or **msdHash** being present when **ProtectedMessageSecurityData** is absent. Also, this failure is caused by the failure of the rule based access control check. This message cannot be processed.

502.b Originator Certificate Validation Failure

(1) The certificate of originator is not validated successfully, because the certificate has expired, the certificate (or any certificate in its certificate path) has been revoked, the certificate (or any certificate in its certificate path) has been compromised, or the signature of the certificate (or any certificate in its certificate path) has failed.

(2) The recipient cannot respond to the purported originator because his identity is unknown.

502.c **Signature (sigValue) Verification Failure**

(1) The signature. Generated by the originator, fails to be verified successfully by the recipient. When content confidentiality and integrity have been applied, the content integrity verification was successful, since the content must be decrypted prior to signature verification. In this case, the integrity of the content is valid, but the purported identity of the originator cannot be verified. The content of the message can be reported to the recipient.

(2) The recipient cannot respond to the purported originator because his identity is unknown.

502.d **Message Hash (msgHash) Verification Failure**

The message hash verification fails. This is the final check to determine that the message has been decrypted, and this is a failure condition. Because decryption has failed, no assumption about the identity of the originator can be made.

502.e **Sequence Signature Verification Failure**

(1) The certificate of CSP process that generated the sequence signature is not validated successfully, because the certificate has expired, the certificate (or any certificate in its certificate path) has been revoked, the certificate (or any certificate in its certificate path) has been compromised, or the signature of the certificate (or any certificate in its certificate path) has failed.

(2) The recipient cannot respond to the purported originator because the CSP message has been tampered with.

Chapter 6

Profiles

601. Overview

a. Chapters 2 through 5 define the services provided in the ACP 120 environment and the policies and procedures to be used with these services. A CSP implementation may claim conformance to this specification if it implements all of the mandatory elements of service (EoS). An implementaion is not required to include services specified as optional; however, if a profile states that some optional EoS shall be supported, then an implementation cannot claim conformance to the specified profile without implementing those optional services. This chapter provides additional information about which of the services and procedures apply in specific messaging environments. In the following sections, the requirements associated with each of the optional EoS are defined.

b. If additional policies or procedures are required in order to conform to the specified profile, these additional requirements are also defined in this chapter.

SECTION II

Taxonomy

602. Profile Taxonomy

The profiles specified by the ACP are organized according to Chapter 5 of ACP 123. The profiles are organized according to a taxonomy similar to the International Standardized Profile (ISP) taxonomy defined by ISO/IEC TR 10000. The taxonomy defines the relationship between each of the profiles required to implement the MMHS. There are two relevant types of profiles are included in the taxonomy: A-profiles which specify the requirements for connection-oriented applications and F-profiles which specify the requirements for abstract information objects conveyed by A profiles. If a requirement for a connectionless application profile is identified it may be included as a B profile. The CSP-relevant portions of the ACP taxonomy are depicted in Figure 6-1.

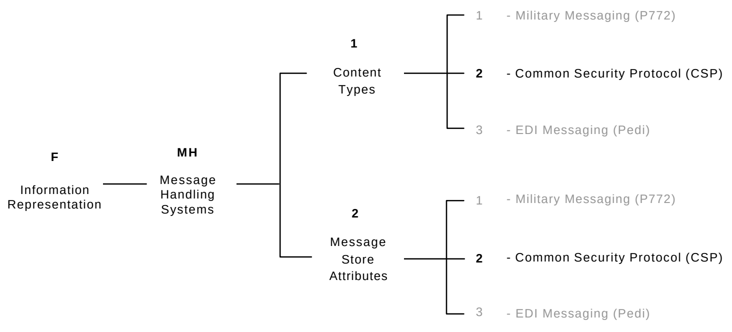
F
Information Representation
MH
Message Handling Systems
1
Content Types
2
Message Store Attributes
1 - Military Messaging (P772)
2 - Common Security Protocol (CSP)
3 - EDI Messaging (Pedi)
1 - Military Messaging (P772)
2 - Common Security Protocol (CSP)
3 - EDI Messaging (Pedi)

Figure 6 -1 - CSP F Profiles

603. **F Profiles**

a. F-profiles specify requirements for abstract information objects conveyed by A and B profiles. Abstract information objects conveyed by A and B profiles include CSP content type and CSP-specific MS attributes.

b. This ACP specifies two F-profiles in Annexes C and D as follows:

– FMH12(D) – CSP content (CSP)
– FMH22(D) – CSP-specific MS Attributes

SECTION III

Standard Profile

604. **Profile Definition**

a. FMH12(D) states the conformance requirements for all components claiming to originate and receive CSP message contents. The profile states the additional requirements beyond the Information Object Implementation Conformance Statement (IO-ICS), located in Annex B, required to achieve an interoperable CSP environment. In order to provide a common level of services to users the following Elements of Service are mandatory for all components originating and receiving CSP messages:

– Content Confidentiality and Integrity (encryption and key distribution) with authenticated key distribution and support for rule-based access control
– Proof of Content Origin (message signature)
– Proof of Content Delivery After Security Processing (signed receipts)
– Integrity of CSP Encapsulation (sequence signature)
– Plain Text Content Description (content description)
– Extensions
– Forwarding
– Group Addressing
– Blind Copies

b. In addition to the support requirements stated in Chapter 2 through 5, the following Elements of Service are mandatory for reception but optional for origination in FMH12(D):

– Mail List Expansion History (ML Expansion History)
– Additional Signatures
– Content Type and Message Identification

c. FMH22(D) state the conformance requirements for components claiming to implement a P7 interface requiring access to CSP-specific MS attributes.

605. **Implementation Requirements**

a. Implementations claiming to originate and receive CSP message contents shall implement FMH12(D).

b. Implementation claiming to support CSP-specific MS attributes shall implement FMH22(D).

Annex A

CSP Message Store Attributes

A-101. **Overview**

A Message Store (MS) provides the User Agent with a repository of messages. These messages are information objects which may be characterized by MS attributes that are specific to a particular content type. Other generic attributes also exist for any content type. This Annex defines the MS attributes specific to CSP. Table C-1 lists these MS attributes; whether an attribute is Single or Multi-valued; the field within the CSP heading that serves as the source for values of the attribute; and whether the attribute may be used for lists, alerts, or summaries.

Attribute-type-name	Single/ multi valued	Source parameter	Available for list, alert	Available for summarize
SecurityServices	S	tokenProtectionAlgorithm signatureAlgorithm cspSequenceSignatureAlgorithm	Y	
CertificationPaths	S	originatorCertificate signatureCertificate cspSequenceSignatureCertificate	Y	
ContentDescription	S	contentDescription	Y	
EncapsulatedContentType	S	ContentType (from: SignatureInformation)	Y	Y
OrganizationMessage	S	signatureCertificate	Y	Y
Scid	S	signedContentIdentifier	Y	
MLExpansionHistory	S	MlData	Y	
SignReceipt	S	ReceiptInformation	Y	Y

Table A-1 CSP Message Store Attributes

A-102. **Attributes**

A MS can report the attributes described below regarding a CSP information object.

A-102.a **Security Services**

A Sequence of Object Identifiers representing the security mechanisms and the corresponding algorithms applied to this message. Up to three OBJECT IDENTIFIERs can be reported, if the message was encrypted, if the message was signed, or if the protected message (CSP) was signed.

```
securityServices        ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX  SecurityServices
    MATCHES FOR EQUALITY
    SINGLE VALUE
    ::=  id-securityServices
```

```
SecurityServices  ::=  SEQUENCE {
    encrypt             [0] OBJECT IDENTIFIER OPTIONAL,
            -- message was encrypted
            -- from tokenProtectionAlgorithm
    signed              [1] OBJECT IDENTIFIER OPTIONAL,
            -- message was signed
            -- from signatureAlgorithm
    seq-signed          [2] OBJECT IDENTIFIER OPTIONAL
            -- protected message (CSP) was signed
            -- from cspSequenceSignatureAlgorithm   }
```

A-102.b **Certification Paths**

A Sequence of Certification paths of the originator of the message that generated a token, the content signature, and the Message Sequence Signature.

```
certificationPaths          ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX  CertificationPaths
    MATCHES FOR EQUALITY
    SINGLE VALUE
    ::=  id-certificationPaths

CertificationPaths   ::= SEQUENCE {
    -- at least one of the following must be present
    -- if the message is signed and encrypted, but only one certificate is present
    -- in the SignatureBlock, then only the messageSigner is present
    messageSigner       [0] CertificationPath OPTIONAL,
            -- present if message was signed
    tokenEncrypter      [1] CertificationPath OPTIONAL,
            -- present if message was encrypted
    sequenceSigner      [2] CertificationPath OPTIONAL
            -- present if CSP Message Sequence was signed  }
```

Note: CertificationPath is defined in the 1988 X.509 Recommendation, page 79.

A-102.c **Content Description**

This attribute is the content description from the CSP heading. The content description is an unprotected string, which may offer some processing information to the recipient.

```
contentDescription          ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX  TeletexString
    MATCHES FOR SUBSTRINGS
    SINGLE VALUE
    ::=  id-contentDescription
```

A-102.d **Encapsulated Content Type**

This attribute is the content type of the original protected message. This attribute may exist for a message only if it is signed.

```
encapsulatedContentType          ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX  CspContentType
    MATCHES FOR EQUALITY
    SINGLE VALUE
    ::=  id-encapsulatedContentType
```

A-102.e **<u>Organizational Message</u>**

This attribute is a simple Boolean and is true if the message is purportedly organizational. The attribute is set to a true value if the privilege information (contained in either the Signature or Signature and KM Certificate) indicates this message was signed by a Release Authority. Only after the recipient validates the certificates and the message signature can the message be conclusively determined to be Organizational.

```
organizationMessage    ATTRIBUTE
      WITH ATTRIBUTE-SYNTAX  BOOLEAN
      MATCHES FOR EQUALITY
      SINGLE VALUE
      ::=  id-origanizationMessage
```

A-102.f **<u>Signed Content Identifier</u>**

This attribute is the content identifier generated by the originator of the message. If the message is an interpersonal message or the message is a military message, then the identifier is the IPMIdentifier of the message. If the message contains a signed receipt, then the identifier if from the original signed message.

```
sCID              ATTRIBUTE
      WITH ATTRIBUTE-SYNTAX  SignedContentIentifier
      MATCHES FOR EQUALITY
      SINGLE VALUE
      ::=  id-SCID

signedContentIdentifier  ::=  OCTET STRING
```

A-102.g **<u>Mail List Expansion History</u>**

This attribute is a list of Mail List Identifiers of the Mail List Agents that have processed this message.

```
mLExpansionHistory     ATTRIBUTE
      WITH ATTRIBUTE-SYNTAX  SEQUENCE OF MLID
      MATCHES FOR EQUALITY
      SINGLE VALUE
      ::=  id-mLExpansionHistory

MLID                                    ::= Kmid

Kmid                                    ::= OCTET STRING
```

A-102.h **<u>Signed Receipt</u>**

This attribute is true if the message contains a signed receipt.

```
signedRequested          ATTRIBUTE
      WITH ATTRIBUTE-SYNTAX  BOOLEAN
      MATCHES FOR EQUALITY
      SINGLE VALUE
      ::=  id-receiptRequested
```

A-103. **Definitive CSP MS Attribute ASN.1 Module**

```
CSPMSAttributes {id-infosec(joint-iso-ccitt 16 840 1 101 2 1) id-modules(0) id-cspMSAttributes(15)}

Definitions IMPLICIT TAGS ::=

BEGIN

IMPORTS

-- Directory definitions

        CertificationPath
                FROM AuthenticationFramework {joint-iso-ccitt ds(5)
                        module(1) authentication-framework(7) 2}

-- CSP

        CspContentType, Kmid
                FROM CommonSecurityProtocol {id-infosec(joint-iso-ccitt 16 840 1 101 2 1)
                        id-modules(0) id-csp(16)} ;

securityServices    ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX  SecurityServices
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::=  id-securityServices

securityServices    ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX  SecurityServices
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::=  id-securityServices

SecurityServices  ::=  SEQUENCE {
        encrypt                 [0] OBJECT IDENTIFIER OPTIONAL,
                -- message was encrypted
                -- from tokenProtectionAlgorithm
        signed                  [1] OBJECT IDENTIFIER OPTIONAL,
                -- message was signed
                -- from signatureAlgorithm
        seq-signed              [2] OBJECT IDENTIFIER OPTIONAL
                -- protected message (CSP) was signed
                -- from cspSequenceSignatureAlgorithm   }

certificationPaths   ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX  CertificationPaths
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::=  id-certificationPaths

CertificationPaths   ::= SEQUENCE {
        -- at least one of the following must be present
        -- if the message is signed and encrypted, but only one certificate is present
        -- in the SignatureBlock, then only the messageSigner is present
        messageSigner      [0] CertificationPath OPTIONAL,
                -- present if message was signed
        tokenEncrypter     [1] CertificationPath OPTIONAL,
                -- present if message was encrypted
        sequenceSigner     [2] CertificationPath OPTIONAL
                -- present if CSP Message Sequence was signed  }

contentDescription ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX  TeletexString
        MATCHES FOR SUBSTRINGS
        SINGLE VALUE
        ::=  id-contentDescription
```

```
encapsulatedContentType                  ATTRIBUTE
               WITH ATTRIBUTE-SYNTAX  CspContentType
               MATCHES FOR EQUALITY
               SINGLE VALUE
               ::=  id-encapsulatedContentType

organizationMessage            ATTRIBUTE
               WITH ATTRIBUTE-SYNTAX  BOOLEAN
               MATCHES FOR EQUALITY
               SINGLE VALUE
               ::=  id-origanizationMessage

sCID           ATTRIBUTE
               WITH ATTRIBUTE-SYNTAX  SignedContentIentifier
               MATCHES FOR EQUALITY
               SINGLE VALUE
               ::=  id-SCID

signedContentIdentifier  ::=  OCTET STRING

mLExpansionHistory             ATTRIBUTE
               WITH ATTRIBUTE-SYNTAX  SEQUENCE OF MLID
               MATCHES FOR EQUALITY
               SINGLE VALUE
               ::=  id-mLExpansionHistory

MLID  ::=  Kmid

signedRequested   ATTRIBUTE
               WITH ATTRIBUTE-SYNTAX  BOOLEAN
               MATCHES FOR EQUALITY
               SINGLE VALUE
               ::=  id-receiptRequested

-- OIDS

ID  ::= OBJECT IDENTIFIER

id-infosec                     ID  ::=  { 2 16 840 1 101 2 1 }
id-modules                     ID  ::=  { id-infosec 0 }
id-cspMSAttributes             ID  ::=  { id-modules 15 }

END – CSPMSAttributes
```

Annex B

Common Security Protocol

Information Object Implementation Conformance Statement (IO-ICS) Proforma

1 Scope

This annex is an Information Object Implementation Conformance Statement (IO-ICS) Proforma for the CSP information object as specified in ACP120. This IO-ICS Proforma is in compliance with the relevant requirements, and in accordance with the relevant guidance for IO-ICS Proforma given in ISO/IEC 9642-7.

Details of the use of this proforma are provided in Appendix A.

The scope of this annex is the specification of the conformance statements for the content specific functionality for any implementation that supports the CSP content type. It has been developed by examining the ASN.1 defined in Chapter 4 of this document. If an ASN.1 element is optional then the corresponding classification applied to the element in this IO-ICS Proforma is optional. If an ASN.1 element is mandatory then the corresponding classification applied to the element in this IO-ICS Proforma is mandatory. Implementors shall state herein the level of support for all CSP EoS and protocol elements.

2 Normative references

- SDN 701 Message Security Protocol 4.0, Revision A, February 1997
- SDN 702: SDNS Directory Specifications for Utilization with the SDNS Message Protocol, Revision 2, 5 May 1997
- ISO 8824: 1990, Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).
- ISO/IEC TR 10000-1: 1990, Information technology - Framework and taxonomy of International Standardized Profiles - Part 1: Framework.
- ISO/IEC TR 10000-2: 1990, Information technology - Framework and taxonomy of International Standardized Profiles - Part 2: Taxonomy.
- ISO/IEC 10021-1: 1990, Information technology - Text Communication - Message-Oriented Text Interchange Systems (MOTIS) - Part 1: Service Overview. [see also CCITT Recommendation X.400(1988)]
- ISO/IEC 10021-2: 1990, Information technology - Text Communication - Message-Oriented Text Interchange Systems (MOTIS) - Part 2: Overall Architecture. [see also CCITT Recommendation X.402(1992)]
- ISO/IEC 10021-4: 1990, Information technology - Text Communication - Message-Oriented Text Interchange Systems (MOTIS) - Part 4: Message Transfer System: Abstract Service Definition and Procedures. [see also CCITT Recommendation X.411(1992)]
- ISO/IEC 10021-5: 1990, Information technology - Text Communication - Message-Oriented Text Interchange Systems (MOTIS) - Part 5: Message Store: Abstract Service Definition. [see also CCITT Recommendation X.413(1992)]

3 Definitions

This document uses terms defined in ACP 120.

This Specification uses the following terms defined in ISO/IEC 9646-1

Implementation Conformance Statement proforma
Implementation Conformance Statement
Information Object Implementation Conformance Statement proforma
Information Object Implementation Conformance Statement

4 Abbreviations

ACP	Allied Communication Publication
AMH	Application Message Handling
CCITT	International Telegraph and Telephone Consultative Committee
CSP	Common Security Protocol
CSP-UA	Common Security Protocol User Agent
DoD	Department of Defense
EoS	Element of Service
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
IUT	Implementation Under Test
MS	Message Store
MHS	Message Handling System
PICS	Protocol Implementation Conformance Statement
SDNS	Secure Data Network System
UA	User Agent
UKM	User Keying Material

5 Conventions

The IO-ICS Proforma is provided in Appendix A to this Annex.

6 Conformance

The supplier of an information object implementation that is claimed to conform to CSP is required to complete a copy of the IO-ICS Proforma provided in Appendix A and is required to provide the information necessary to identify both the supplier and the implementation.

The scope of conformance to this IO-ICS covers content specific functionality for any implementation that supports the MM content type. Conformance to this IO-ICS does not imply the provision of a standard OSI communications protocol for access to the MTS.

This IO-ICS states requirements upon implementations to achieve interworking. A claim of conformance to this IO-ICS is a claim that all requirements in the relevant base standards are satisfied, and that all requirements in the following clauses and in appendix A of this IO-ICS are satisfied. Appendix A states the relationship between these requirements and those of the base standards.

6.1 Conformance statement

For each implementation claiming conformance to ACP 120 as specified in this IO-ICS, an ICS shall be made available stating support or non-support of each option identified in this IO-ICS.

6.2 MHS conformance

This IO-ICS specifies implementation options or selections such that conformant implementations will satisfy the non-procedural conformance requirements of ACP 120 as well as those of ISO/IEC 10021.

Implementations whose information objects conform to ACP 120 as specified in this IO-ICS shall implement all the mandatory support (m) features identified as base requirements in appendix A of this IO-ICS, and shall state which optional support (o) features are implemented.

6.3 Error handling

Various distinguished integer values may be defined in supplements to ACP 120 that are not defined in the base ACP. These are not expected to cross national boundaries without bilateral agreement. When an unknown distinguished value is encountered, it should, in general, not result in an error. The value should be treated transparently or ignored, wherever possible.

Appendix A to Annex B
Common Security Protocol
Information Objects Implementation Conformance Statement (IO-ICS) Proforma

A.1 Identifications

A.1.1 Identification of PICS

Ref	Question	Response
1	Date of Statement (DD/MM/YY)	
2	PICS serial number	
3	System Conformance statement cross-reference	

A.1.2 Identification of Implementation and/or System

Ref	Question	Response
1	Implementation Name	
2	Implementation Version	
3	Machine Name	
4	Machine Version	
5	Operating System Name	
6	Operating System Version	
7	Special Configuration	
8	Other Information	

A.1.3 Identification of System Supplier and/or Test Laboratory Client

Ref	Question	Response
1	Organization name	
2	Contract Name(s)	
3	Address	
4	Telephone Number	
5	Telex Number	
6	Fax Number	
7	E-Mail Address	
8	Other Information	

A.2 Identification of the protocol

Ref	Question	Response
1	Title, Reference Number and date of publication of the protocol standard	
2	Protocol Version Numbers	
3	Addenda/amendments/corrigenda	
4	Defect reports implemented	

A.3 Global statement of conformance

Ref	Question	Response
1	Are all mandatory base standards requirements met?	

Note - Answering “No” to this section indicates non-conformance to the protocol specification. Unsupported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is non-conformant. Such information shall be provided in A.7, “Other information”.

A.4 Instruction for completing the PICS Proforma

A.4.1 Definition of support

A capability is said to be supported if the Implementation Under Test (IUT) is able:

– to generate the corresponding service parameters (either automatically or because the end user explicitly requires the capability);
– To interpret, handle and when required make available to the end user the corresponding service parameter(s).

A capability is referred to as either originator capability (when the MHS end-user is acting as an originator) or a recipient capability (when an MHS end-user is acting as a recipient).

A protocol element is said to be supported for origination if the IUT is able to generate it under some circumstances (either automatically or because end-user explicitly requires the related capability).

A protocol element is said to be supported for reception if it is correctly interpreted and handled and also when required, made available to the end-user.

A.4.2 Base column

The column indicates the level of support required for conformance to CSP.

The values are as follows:

m	Mandatory support is required.
o	Optional support is permitted for conformance to CSP. If implemented it must conform to the specifications and restrictions contained in CSP. These restrictions may affect the optionality of other items.
c	The item is conditional; the support of this item is subject to a predicate which is referenced in the note column.

A.4.3 Support column

This column shall be completed by the supplier or implementor to indicate the level of implementation of each item. The proforma has been designed such that the only entries required in that column are:

Y	Yes, the item has been implemented;
N	No, the item has not been implemented;
–	The item is not applicable.

In the PICS Proforma tables, every leading items marked “m” should be supported by the IUT. Sub-elements marked “m” should be supported if the corresponding leading feature is supported by the IUT.

All entries within the PICS shall be made in ink. Alterations to such entries shall be made by crossing out, not erasing nor making the original entry illegible, and writing the new entry alongside. All such alterations to records shall be initialed by the staff making them.

A.4.4 Note column

The “Note” column has to be read as follows:

Notexx	Refers to Note xx.
pxx	Refers to predicate xx.

A.4.5 Item column

Each row within the PICS Proforma which requires implementation details to be entered is numbered in a separate column. This numbering is included as a means of uniquely identifying all possible implementation details within the PICS Proforma.

A.4.6 Predicate definitions

If the classification of an Element of Service (EoS) or a Protocol Element is subject to a predicate, support of the item is mandatory if the related predicate is true. Otherwise support of the item is optional.

Within this appendix, predicates are clearly defined where:

p1	Content Confidentiality and Integrity.
p2	Proof of Content Origin.
p3	Proof of Content Delivery After Signature Processing.
p4	Integrity of CSP Encapsulation.
p5	Plain Text Content Description.
p6	Mail List Expansion History.
p7	Additional Signatures.
p8	Extensions.
p9	Forwarding.
p10	Group Addressing.
p11	Content Type and Message Identification.
p12	Blind Copies.
p13	If the originator’s AttributeCertificates are associated with the user’s X.509 key management certificate then support is m, else o.
p14	If the originator’s AttributeCertificates are associated with the user’s X.509 signature certificate then support is m, else o.
p15	If messageSecurityData is present them support is m, else o.
p16	If access control is required then support is m, else o.
p17	If the message is encrypted and the signature is required to be encrypted then support is m, else o.
p18	If Content Confidentiality and key management is performed with a key transfer algorithm (e.g., RSA), then support is m, else c.

A.5 Capabilities and Options

The following tables list the detailed requirements for a CSP implementations.

A.5.1 Supported Options

Ref	Question		Base	Support	Note
1	Supported Content Types	CSP	m		
		MM (P772)	m		
		IPMS (P22)	o		
		pEDI (P35)	o		
2	CSP use of X.509 certificate?		m		
3	CSP use of	X.400 rekey agent	o		
		SDNS Key management system	o		
4	CSP use of X.411 abstract services?		o		
5	CSP use of X.500 directory?		o		

A.5.2 CSP EoS

Item	Element of Service[1]	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	Content Confidentiality and Integrity (encryption and key distribution) with authenticated key distribution and support for rule-based access control.	o	o			
2	Proof of Content Origin (message signature).	o	o			
3	Proof of Content Delivery After Security Processing (signed receipts).	o	o			
4	Integrity of CSP Encapsulation (sequence signature) [2].	o	o			Note 2
5	Plain Text Content Description (content description).	o	o			
6	Mail List Expansion History (ML Expansion History).	o	o			
7	Additional Signatures	o	o			
8	Extensions	o	o			
9	Forwarding	o	o			Note 3
10	Group Addressing	o	o			Note 4
11	Content Type and Message Identification	o	o			
12	Blind Copies	o	o			Note 5

Notes:

1. The names of these EoS are CSP specific.
2. Support for this EoS implies the ability to generate a sequence signature over the CSP message, not to sign the message contents.
3. This a requirement for the ACP123 profile. Support of this service depends on the ACP123 UA supporting forwarding
4. This is a requirement if bilateral national agreements exist to support Mail Lists and Mail List Agents. Support for this EoS on origination is mandatory only for MLAs and support on reception means you can receive a message that has been processed by a ML.
5. This is a requirement for the ACP 123 profile. Support of this service depends on the ACP 123 UA supporting Blind Copies via separate message submissions

A.5.3 Supported Information Objects

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	CommonSecurityProtocol	m	m			
1.1	Csp	m	m			see A.5.3.1
1.2	signedCsp	c	m			p4, p18, see A.5.3.2

A.5.3.1 Csp

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	originatorSecurityData	c	m			p1, p16
1.1	confidentialityAlgorithm	m	m			see A.5.3.1.1
1.2	integrityAlgorithm	m	m			see A.5.3.1.1
1.3	tokenProtectionAlgorithm	m	m			see A.5.3.1.1
1.4	msdProtectionAlgorithm	c	m			p16, see A.5.3.1.1
1.5	keyManagementCertificate	c	m			p1, see A.5.3.3(1)
1.6	messageSecurityData	c	m			p16, Note 1
1.6.1	label	m	m			see A.5.3.1.2
1.6.2	kmAttrCerts	c	m			p13, see A.5.3.3(2)
2	signatureBlock	c	m			p2, p3
2.1	signatureAlgorithm	m	m			see A.5.3.1.1
2.2	signatureValue	m	m			
2.2.1	sigValue	c	m			p2, p3
2.2.2	encSigData	c	m			p17, Note 2
2.2.2.1	encSigAlgorithm	m	m			see A.5.3.1.1
2.2.2.2	encSigValue	m	m			
2.3	controlInformation	m	m			
2.3.1	signatureInformation	c	m			p2
2.3.1.1	encapsulatedContentType	c	m			p11, see A.5.3.1.3
2.3.1.2	signedContentIdentifier	m	m			
2.3.1.3	receiptRequests	m	m			
2.3.1.3.1	allOrNone	m	m			
2.3.1.3.1.1	noReceipt	m	m			
2.3.1.3.1.2	allReceipts	m	m			
2.3.1.3.1.3	firstTierRecipients	m	m			
2.3.1.3.2	oldReceiptList	o	m			
2.3.1.3.3	receiptList	m	m			

A.5.3.1 Continued)

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
2.3.1.4	receiptTo	o	o			
2.3.1.4.1	newReceiptsTo	o	o			See A.5.3.3(3)
2.3.1.4.2	oldReceiptsTo	o	o			See A.5.3.3(4)
2.3.2	recipientInformation	c	m			p3
2.3.2.1	encapsulatedContentType	c	m			p11, see A.5.3.1.3
2.3.2.2	signedContentIdentifier	m	m			
2.4	signatureCertificate	c	m			p2, p3, see A.5.3.3(1)
2.5	sigAttrCerts	c	m			p14, see A.5.3.3(2)
3	recipientSecurityData	c	m			p1, p16
3.1	tag	m	m			
3.1.1	pairwise	o	m			
3.1.1.1	kmid	m	m			
3.1.1.2	edition	m	m			
3.1.1.3	date	o	m			
3.1.2	mlTag	c	m			p10
3.1.2.1	mlid	m	m			
3.1.2.2	mlKeyDate	m	m			
3.1.3	keyTransfer	o	m			
3.1.3.1	issuer	m	m			see A.5.3.3(3)
3.1.3.2	serialNumber	m	m			
3.1.3.3	pkEncryptedTokenProtectionKey	m	m			
3.2	protectedRecipientKeyToken	m	m			see A.5.3.1.4, Note 3
4	contentDescription	c	m			p5
5	mlExpansionHistory	c	m			p6
5.1	mlid	m	m			
5.2	expansionTime	m	m			
5.3	mlReceiptPolicy	o	m			
5.3.1	none	m	m			
5.3.2	insteadOf	m	m			See A.5.3.3(3)
5.3.3	inAdditionTo	m	m			See A.5.3.3(3)
6	additionalSignatures	c	m			p7
6.1	addSigAlgorithm	m	m			see A.5.3.1.1
62	addSigValue	m	m			

A.5.3.1 (Continued)

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
6.3	addSigControlInfo	m	m			
6.3.1	addSigIdentifier	o	m			
6.3.2	addSigData	m	m			
6.3.2.1	addSigOnContent	m	m			
6.3.2.2	addSigOnContentAndSigBlock	m	m			
6.3.2.2.1	contentIdentifier	m	m			
6.3.2.2.2	originatorName	m	m			See A.5.3.3(3)
6.3.2.3	addSigOnReceipt	m	m			
6.3.2.3.1	contentIdentifier	m	m			
6.3.2.3.2	recipientSenderName	m	m			See A.5.3.3(3)
6.3.3	timeStampData	o	m			
6.3.3.1	timeStamp	m	m			
6.3.3.2	addTimeStampData	o	m			
6.3.3.2.1	timeStampPolicy	m	m			
6.3.3.2.2	timeStampVerifyData	o	m			
6.4	addSigCertificate	m	m			see A.5.3.3(1)
6.5	addAttrCertificates	c	m			p14, see A.5.3.3(2)
7	cspExtensions	c	m			p8
7.1	extnID	m	m			
7.2	critical	m	m			d(false)
7.3	extnValue	m	m			
8	cspSequenceSignatureCertificate	o	m			see A.5.3.3(1)
9	cspSequenceSigAttrCerts	o	m			see A.5.3.3(2)
10	encapsulatedContent	c	m			p1, p2

Notes:

1 Protected form of MessageSecurityData, encrypted using msgKey.

2 Protected form of SigValue.

3 Protected form of RecipientKeyToken.

A.5.3.1.1 Algorithm Identifier

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	algorithm	m	m			
2	parameters	m	m			

A.5.3.1.2 Security Label

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	security-policy-identifier	o	m			
2	security-classification	o	m			
2.1	unclassified	m	m			
2.2	confidential	m	m			
2.3	secret	m	m			
2.4	top-secret	m	m			
3	privacy-mark	o	m			
4	security-categories	o	m			

A.5.3.1.3 CSP Content Type

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	built-in	m	m			
1.1	unidentified	m	m			
1.2	external	m	m			
1.3	interpersonal-messaging-1984	m	m			
1.4	interpersonal-messaging-1988	m	m			
2	external	m	m			
3	externalWithSubtype	m	m			
3.1	external	m	m			
3.2	subtype	m	m			

A.5.3.1.4 Recipient Key Token

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	msgKey	m	m			
2	msgHash	m	m			
3	signatureBlockIndicator	m	m			
4	encapsulatedContentType	m	m			see A.5.3.1.3
5	msdHash	c	m			p15

A.5.3.2 Signed CSP

Item	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	toBeSigned	m	m			see A.5.3.1
2	algorithmIdentifier	m	m			see A.5.3.1.1
3	encrypted	m	m			

A.5.3.3 Common Data Types

Ref	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
1	CertificationPath					
1.1	userCertificate	m	m			
1.2	theCACertificates	m	m			
1.3	CertificatePair	m	m			
1.4	forward	m	m			
1.5	reverse	o	o			
2	AttributeCertificationPath					
2.1	attributeCertificate	m	m			
2.2	acPath	o	m			
2.2.1	certificate	o	m			
2.2.2	attributeCertificate	o	m			
3	GeneralNames					
3.1	GeneralName	m	m			
3.1.1	otherName	o	o			
3.1.2	rfc822Name	o	o			
3.1.3	dNSName	o	o			
3.1.4	x400Address	o	o			
3.1.5	directoryName	o	o			
3.1.6	ediPartyName	o	o			
3.1.6.1	nameAssigner	o	o			
3.1.6.2	partyName	m	m			
3.1.7	uniformResourceIdentifier	o	o			
3.1.8	iPAddress	o	o			
3.1.9	registeredID	o	o			

A.5.3.3 (Continued)

Ref	Element	Base		Support		Note
		Orig.	Rec.	Orig.	Rec.	
4	ORName					
4.1	ORAddress	o	o			
4.2	DirectoryName	o	o			

A.5.3.4 Forwarded CSP Message Body Part

Ref	Element	Base		Support		References
		Orig.	Rec.	Orig.	Rec.	
1	forwarded-CSP-MSG-body-part	m	m			

A.6 Message Store Attributes

As described in Recommendation X.413, an MS maintains and provides access to certain attributes (e.g., a content descriptor) of each information object it holds. An attribute comprises a type and, depending upon the type, one or more values.

This section classifies the MS attributes specific to CSP and only applies to a MS.

For each Csp it holds or handled the implementation shall support the following attributes.

Item	Element	Base	Support	Note
1	securityServices	o		
2	certificationPaths	o		
3	contentDescription	c		p5
4	EncapsulatedContentType	o		
5	organizationalMessage	o		
6	sCID	o		
7	mlExpansionHistory	o		
8	signedRequest	o		

A.7 Other Information

Item	**Other Information**

Annex C

Profile for Messaging in the Combined Messaging and Directory Environment

Standardized Profile

TITLE: Information technology - Standardized Profiles -
Military Message Handling Systems - Content Types

FMH12(D) - CSP Content (CSP)

SOURCE: ACP 123 Editor

STATUS: Draft text, 1997-09-08

This document is a Standardized Profile (SP) for Military Message Handling System (MMHS) covering Military requirements. It is outside the scope of the current Taxonomy Framework for International Standardized Profiles (ISP). This SP is a content specific profile for the Common Security Protocol content type referred to as CSP as defined in the Allied Communication Publication (ACP) 120. This SP only specifies requirements for any implementation that supports the CSP content type.

Introduction

This Standardized Profile (SP) is defined within the context of functional standardization, in accordance with the principles specified by ISO/IEC TR 10000, “Framework and Taxonomy of International Standardized Profiles”. The context of functional standardization is one part of the overall field of Information Technology (IT) standardization activities - covering base standards, profiles, and registration mechanisms. A profile defines a combination of base standards that collectively perform a specific well-defined IT function. Profiles standardize the use of options and other variations in the base standards to promote system interoperability and to provide a basis for the development of uniform, internationally recognized system tests.

One of the most important roles for a SP is to serve as the basis for the development of recognized tests. SPs also guide implementors in developing systems that fit the needs of the MMHS. SPs are produced not simply to ‘legitimize’ a particular choice of base standards and options, but to promote real system interoperability. The development and widespread acceptance of tests based on this and other SPs is crucial to the successful realization of this goal.

FMH12(D) covers content specific functionality for any implementation that supports the MM content type. It specifies support of the CSP content 'protocol' in terms of basic requirements and optional functional groups and defines conformance requirements for implementations with respect to support of the CSP content and associated functionality.

FMH12(D) contains two normative appendices:

Appendix A CSP Elements of Service

Appendix B SPICS Requirements List for FMH12(D)

Information technology - Standardized Profiles - Military Message Handling Systems - Content Types FMH12(D) - CSP Content (CSP)

1 Scope

1.1 General

FMH12(D) covers representation of the CSP content type by conforming implementations (see also figure 1). These specifications form part of the MMHS application functions.

1.2 Position within the taxonomy

FMH12(D) specifies the profile which states requirements for the CSP content-type.

1.3 Scenario

The model assumes the exchange of military messages (content type CSP) by two cooperating implementations. The CSP information objects originated and received by the implementations may be transferred via either:

1. a direct connection between the implementations (i.e., an A-profile);
2. direct connections to a relaying system (which need not comply with this profile);
3. by other bilaterally agreed means.

The specific transfer mechanism used must be mutually agreed but is outside the scope of this profile. This relationship is illustrated in figure 1.

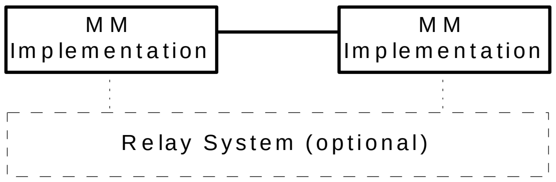

Figure 1 - FMH12(D) scenario

The MHS services and functions covered by the FMH12(D) profile are specified in ACP 120. There are no OSI upper layer services and protocols within the scope of the FMH12(D) profile.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of FMH12(D). At the time of publication, the editions indicated were valid. All documents are subject to revision, and parties to agreements based on FMH12(D) are warned against automatically applying any more recent editions of the documents listed below, since the nature of references made by SPs to such documents is that they may be specific to a particular edition. Members of IEC and ISO maintain registers of currently valid International Standards and ISPs, and CCITT maintains published editions of its current Recommendations.

Technical corrigenda to the base standards referenced are listed in appendix B of the IO-ICS.

NOTE - References in the body of FMH12(D) to specific clauses of ISO/IEC documents shall be considered to refer also to the corresponding clauses of the equivalent CCITT Recommendations (as noted below) unless otherwise stated.

ACP 120: Common Security Protocol, May 1997.

ISO/IEC TR 10000-1: 1992, Information technology - Framework and taxonomy of International Standardized Profiles -Part 1: Framework.

ISO/IEC TR 10000-2: 1992, Information technology - Framework and taxonomy of International Standardized Profiles -Part 2: Taxonomy.

ISO/IEC 10021-1:1990/Am.2, *Information technology – Message-Oriented Text Interchange System (MOTIS) – Part 1: System and Service Overview, Amendment 2 Message Store Extensions 1994.*

ISO/IEC 10021-2:1990/Am.1, *Information technology – Message-Oriented Text Interchange System (MOTIS) – Part 2: Overall Architecture; Amendment 1: Representation of O/R Addresses for Human Exchange* 1994.

ISO/IEC 10021-2:1990/Am.2*, Information technology – Message-Oriented Text Interchange System (MOTIS) – Part 2: Overall Architecture; Amendment 2: Minor Enhancements: Multinational Organizations and Terminal-form Addresses 1994.*

(Application for copies of these documents should be addressed to the American National Standards Institute, 11 West 42nd Street, NY, NY 10036 or to ISO, Van Demonstrate 94, 1013 CN Amsterdam, Netherlands.)

3 Definitions

For the purposes of FMH12(D), the following definitions apply. Terms used in FMH12(D) are defined in the referenced base standards. In addition, the following terms are defined.

3.1 General

MM base standard : the base standard referred to in this F-profile is ACP 120.

Basic requirement : an Element of Service, protocol element, procedural element or other identifiable feature specified in the base standards which is required to be supported by all MMHS implementations conforming to this SP.

3.2 Support classification

To specify the support level of arguments, results and other protocol features for FMH12(D), the following terminology is defined.

The classification of information objects and items (elements) is relative to that of the containing information element, if any. Where the constituent elements of a non-primitive element are not individually specified, then each shall be considered to have the classification of that element. Where the range of values to be supported for an element is not specified, then all values defined in the MM base standard shall be supported.

3.2.1 Static capability

The following classifications are used in FMH12(D) to specify static conformance requirements - i.e., capability.

mandatory support (**m**) : the element or feature shall be supported. An implementation shall be able to generate the element, and/or receive the element and perform all associated procedures (i.e., implying the ability to handle both the syntax and semantics of the element) as relevant, as specified in the MM base standard. Where support for origination (generation) and reception are not distinguished, then both capabilities shall be assumed.

NOTE - Where required by the base standards, mandatory support also implies that the implementation shall be able to pass the element on the origination port/reception port to/from the corresponding element on the submission port/delivery port/retrieval port.

optional support (**o**) : an implementation is not required to support the element. If support is claimed, the element shall be treated as if it were specified as mandatory support. If support for origination is not claimed, then the element is not generated. If support for reception is not claimed, then an implementation may ignore the element on delivery, but will not treat it as an error.

4 Abbreviations and Acronyms

ACP	Allied Communication Publication
AMH	Application Message Handling
CCITT	International Telegraph and Telephone Consultative Committee
EoS	Element of Service
IEC	International Electrotechnical Commission
ISO	International Standards Organization
ISP	International Standardized Profile
IT	Information Technology
ITU	International Telecommunications Union
ITU-T	ITU Telecommunications Standardization Sector
MHS	Message Handling Systems
MM	Military Message
MN	Military Notification
MOTIS	Message-Oriented Text Interchange Systems
MTS	Message Transfer System
OSI	Open Systems Interconnection
PICS	Protocol Implementation Conformance Statement
SP	Standardized Profile
SPICS	Standardized Profile Implementation Conformance Statement

Support level for protocol elements and features (see clause 3.2):

m	mandatory full support
o	optional support

5 Conformance

The scope of conformance to profile FMH12(D) covers content specific functionality for any implementation that supports the MM content type. Conformance to profile FMH12(D) does not imply the provision of a standard OSI communications protocol for access to the MTS.

FMH12(D) states requirements upon implementations to achieve interworking. A claim of conformance to FMH12(D) is a claim that all requirements in the relevant base standards are satisfied, and that all requirements in the following clauses and in appendices A and B of

FMH12(D) are satisfied. Appendix B states the relationship between these requirements and those of the base standards.

5.1 Conformance statement

For each implementation claiming conformance to profile FMH12(D), a IO-ICS shall be made available stating support or non-support of each option identified in FMH12(D).

5.2 MHS conformance

FMH12(D) specifies implementation options or selections such that conformant implementations will satisfy the conformance requirements for ACP 120 common security protocol messages.
Implementations conforming to profile FMH12(D) shall implement all the mandatory support (m) features identified as profile requirements in appendices A and B and shall state which optional support (o) features are implemented.

6 Basic Requirements

Appendix A specifies the basic requirements for support of CSP EoS for conformance to FHM12(D). This clause qualifies the basic requirements for specific CSP features.

6.1 Message Length

If an implementation imposes any constraint on the size of the message content, then such constraints shall be stated in the IO-ICS.

6.2 Number of recipients

If an implementation imposes any limit on the number of recipients that can be specified in an CSP heading, then such a limit shall be stated in the IO-ICS.

7 Naming and Addressing

Support for directory names is mandatory.

Appendix A to Annex C
(normative)
CSP Elements of Service

A.1 CSP-specific Elements of Service

The following tables specify the requirements for support of CSP-specific Elements of Service by an MTS-user in an MM environment (i.e. MM-UA) for conformance to FHM12(D).

In the following tables, the "Profile" column reflects the basic requirements for conformance to FMH12(D) - i.e. the minimum level of support required by all implementations claiming conformance to this profile (see clause 6).

Table A.1 - Elements of Service Belonging to the Basic CSP Service (CSP EoS)

Ref	**Element of Service**[1]	**Profile**		**Note**
		Orig	**Rec.**	
1	Content Confidentiality and Integrity (encryption and key distribution) with authenticated key distribution and support for rule-based access control.	m	m	
2	Proof of Content Origin (message signature).	m	m	
3	Proof of Content Delivery After Security Processing (signed receipts).	m	m	
4	Integrity of CSP Encapsulation (sequence signature) .	m	m	Note 2
5	Plain Text Content Description (content description).	m	m	
6	Mail List Expansion History (ML Expansion History).	o	m	
7	Additional Signatures	o	m	
8	Extensions	m	m	
9	Forwarding	m	m	Note 3
10	Group Addressing	m	m	Note 4
11	Content Type and Message Identification	o	m	
12	Blind Copies	m	m	Note 5

Notes:

1 The names of these EoS are CSP specific.

2 Support for this EoS implies the ability to generate a sequence signature over the CSP message, not to sign the message contents.

3 This a requirement for the ACP123 profile. Support of this service depends on the ACP123 UA supporting forwarding

4 This is a requirement if bilateral national agreements exist to support Mail Lists and Mail List Agents. Support for this EoS on origination is mandatory only for MLAs and support on reception means you can receive a message that has been processed by a ML.

5 This is a requirement for the ACP 123 profile. Support of this service depends on the ACP 123 UA supporting Blind Copies via separate message submissions

Appendix B to Annex C
(normative)
SPICS REQUIREMENTS LIST FOR FMH12(D)

In the event of a discrepancy becoming apparent in the body of FMH12(D) and the tables in this appendix, this appendix is to take precedence.

This appendix specifies the support constraints and characteristics of FMH12(D) on what shall or may appear in the implementation columns of a completed IO-ICS.

Clause B.1 specifies the basic requirements for conformance to profile FMH12(D).

In each table, the "Profile" column reflects the level of support required for conformance to this SP (using the classification and notation defined in clause 3.2).

The "References" column has cross references to other relevant parts of this appendix. When it specifies another table, that table is an expansion of the line at which the reference occurs. If a line number is specified, only that line is related to the line at which the reference occurs. A number in parenthesis () after a table reference refers to that line in the specified table.

B.1 Basic Requirements

B.1.1 Supported information objects

Item	Element	Profile		Note
		Orig.	Rec.	
1	CommonSecurityProtocol	m	m	
1.1	Csp	m	m	see B.1.1.1
1.2	signedCsp	m	m	see B.1.1.2

B.1.1.1 Csp

Item	Element	Profile		Note
		Orig.	Rec.	
1	originatorSecurityData	m	m	
1.1	confidentialityAlgorithm	m	m	see B.1.1.1.1
1.2	integrityAlgorithm	m	m	see B.1.1.1.1
1.3	tokenProtectionAlgorithm	m	m	see B.1.1.1.1
1.4	msdProtectionAlgorithm	m	m	see B.1.1.1.1
1.5	keyManagementCertificate	m	m	see B.1.2(1)
1.6	messageSecurityData	m	m	Note 1
1.6.1	label	m	m	see B.1.1.1.2
1.6.2	kmAttrCerts	o	m	see B.1.2(2)

B.1.1.1 (Continued)

Item	Element	Profile		Note
		Orig.	Rec.	
2	signatureBlock	m	m	
2.1	signatureAlgorithm	m	m	see B.1.1.1.1
2.2	signatureValue	m	m	
2.2.1	sigValue	m	m	
2.2.2	encSigData	o	m	Note 2
2.2.2.1	encSigAlgorithm	m	m	see B.1.1.1.1
2.2.2.2	encSigValue	m	m	
2.3	controlInformation	m	m	
2.3.1	signatureInformation	m	m	
2.3.1.1	encapsulatedContentType	o	m	see B.1.1.1.3
2.3.1.2	signedContentIdentifier	m	m	
2.3.1.3	receiptRequests	m	m	
2.3.1.3.1	allOrNone	m	m	
2.3.1.3.1.1	noReceipt	m	m	
2.3.1.3.1.2	allReceipts	m	m	
2.3.1.3.1.3	firstTierRecipients	m	m	
2.3.1.3.2	receiptsList	m	m	
2.3.1.4	receiptTo	m	m	
2.3.1.4.1	newReceiptsTo	m	m	See B.1.2(3)
2.3.1.4.2	oldReceiptsTo	o	o	See B.1.2(4)
2.3.2	recipientInformation	m	m	
2.3.2.1	encapsulatedContentType	m	m	see B.1.1.1.3
2.3.2.2	signedContentIdentifier	m	m	
2.4	signatureCertificate	m	m	see B.1.2(1)
2.5	sigAttrCerts	o	m	see B.1.2(2)
3	recipientSecurityData	m	m	
3.1	tag	m	m	
3.1.1	pairwise	m	m	
3.1.1.1	kmid	m	m	
3.1.1.2	edition	m	m	
3.1.1.3	date	o	m	
3.1.2	mlTag	m	m	
3.1.2.1	mlid	m	m	
3.1.2.2	mlKeyDate	m	m	

B.1.1.1 (Continued)

Item	Element	Profile		Note
		Orig.	Rec.	
3.1.3	keyTransfer	o	m	
3.1.3.1	issuer	m	m	see B.1.2(3)
3.1.3.2	serialNumber	m	m	
3.1.3.3	pkEncryptedTokenProtectionKey	m	m	
3.2	protectedRecipientKeyToken	m	m	see B.1.1.1.4, Note 3
4	contentDescription	m	m	
5	mlExpansionHistory	o	m	
5.1	mlid	m	m	
5.2	expansionTime	m	m	
5.3	mlReceiptPolicy	m	m	
5.3.1	none	m	m	
5.3.2	insteadOf	m	m	See B.1.2(3)
5.3.3	inAdditionTo	m	m	See B.1.2(3)
6	additionalSignatures	o	m	
6.1	addSigAlgorithm	m	m	see B.1.1.1.1
62	addSigValue	m	m	
6.3	addSigControlInfo	m	m	
6.3.1	addSigIdentifier	o	m	
6.3.2	addSigData	m	m	
6.3.2.1	addSigOnContent	m	m	
6.3.2.2	addSigOnContentAndSigBlock	m	m	
6.3.2.2.1	contentIdentifier	m	m	
6.3.2.2.2	originatorName	m	m	See B.1.2(3)
6.3.2.3	addSigOnReceipt	m	m	
6.3.2.3.1	contentIdentifier	m	m	
6.3.2.3.2	recipientSenderName	m	m	See B.1.2(3)
6.3.3	timeStampData	o	m	
6.3.3.1	timeStamp	m	m	
6.3.3.2	addTimeStampData	o	m	
6.3.3.2.1	timeStampPolicy	m	m	
6.3.3.2.2	timeStampVerifyData	o	m	
6.4	addSigCertificate	m	m	see B.1.2(1)
6.5	addAttrCertificates	o	m	see B.1.2(2)

B.1.1.1 (Continued)

Item	Element	Profile		Note
		Orig.	Rec.	
7	cspExtensions	m	m	
7.1	extnID	m	m	
7.2	critical	m	m	d(false)
7.3	extnValue	m	m	
8	cspSequenceSignatureCertificate	m	m	see B.1.2(1)
9	cspSequenceSigAttrCerts	o	m	see B.1.2(2)
10	encapsulatedContent	m	m	

Notes:

1 Protected form of MessageSecurityData, encrypted using msgKey.

2 Protected form of SigValue.

3 Protected form of RecipientKeyToken.

B.1.1.1.1 Algorithm Identifier

Item	Element	Profile		Note
		Orig.	Rec.	
1	algorithm	m	m	
2	parameters	m	m	

B.1.1.1.2 Security Label

Item	Element	Profile		Note
		Orig.	Rec.	
1	security-policy-identifier	m	m	
2	security-classification	m	m	
2.1	unclassified	m	m	
2.2	confidential	m	m	
2.3	secret	m	m	
2.4	top-secret	m	m	
3	privacy-mark	m	m	
4	security-categories	m	m	

B.1.1.1.3 CSP Content Type

Item	Element	Profile		Note
		Orig.	Rec.	
1	built-in	m	m	
1.1	unidentified	m	m	
1.2	external	m	m	
1.3	interpersonal-messaging-1984	m	m	
1.4	interpersonal-messaging-1988	m	m	
2	external	m	m	
3	externalWithSubtype	m	m	
3.1	external	m	m	
3.2	subtype	m	m	

B.1.1.1.4 Recipient Key Token

Item	Element	Profile		Note
		Orig.	Rec.	
1	msgKey	m	m	
2	msgHash	m	m	
3	signatureBlockIndicator	m	m	
4	encapsulatedContentType	m	m	see B.1.1.1.3
5	msdHash	m	m	

B.1.1.2 signedCsp

Item	Element	Profile		Note
		Orig.	Rec.	
1	toBeSigned	m	m	see B.1.1.1
2	algorithmIdentifier	m	m	see B.1.1.1.1
3	encrypted	m	m	

B.1.2 Common Data Types

Ref	Element	Profile		References
		Orig.	Rec.	
1	CertificationPath			
1.1	userCertificate	m	m	
1.2	theCACertificates	m	m	
1.3	CertificatePair	m	m	
1.4	forward	m	m	
1.5	reverse	m	m	
2	AttributeCertificationPath			
2.1	attributeCertificate	m	m	
2.2	acPath	o	m	
2.2.1	certificate	o	m	
2.2.2	attributeCertificate	o	m	
3	GeneralNames			
3.1	GeneralName	m	m	
3.1.1	otherName	o	o	
3.1.2	rfc822Name	o	o	
3.1.3	dNSName	o	o	
3.1.4	x400Address	o	o	
3.1.5	directoryName	m	m	
3.1.6	ediPartyName	o	o	
3.1.6.1	nameAssigner	o	o	
3.1.6.2	partyName	m	m	
3.1.7	uniformResourceIdentifier	o	o	
3.1.8	iPAddress	o	o	
3.1.9	registeredID	o	o	
4	ORName			
4.1	ORAddress	o	o	
4.2	DirectoryName	m	m	

B.1.3 Body Part Types

Ref	Element	Profile		References
		Orig.	Rec.	
1	forwarded-CSP-MSG-body-part	m	m	

Annex D

Profile for Common Security Protocol Message Store Attributes

Standardized Profile

TITLE: Information technology – Standardized Profiles –
Military Message Handling Systems – Message Store Attributes

FMH22(D) – CSP-specific MS Attributes

SOURCE: ACP 123 Editor

STATUS: Draft text, 1997-09-08

This document is a Standardized Profile (SP) for Military Message Handling System (MMHS) requirements for Common Security Protocol (CSP) specific Message Store (MS) attributes. It is outside the scope of the current Taxonomy Framework for International Standardized Profiles (ISP). This SP is a content specific profile for the CSP-specific MS attributes as defined in the Allied Communication Publication (ACP) 120.

Introduction

This Standardized Profile (SP) is defined within the context of functional standardization, in accordance with the principles specified by ISO/IEC TR 10000, “Framework and Taxonomy of International Standardized Profiles”. The context of functional standardization is one part of the overall field of Information Technology (IT) standardization activities – covering base standards, profiles, and registration mechanisms. A profile defines a combination of base standards that collectively perform a specific well-defined IT function. Profiles standardize the use of options and other variations in the base standards to promote system interoperability and to provide a basis for the development of uniform, internationally recognized system tests.

One of the most important roles for a SP is to serve as the basis for the development of recognized tests. SPs also guide implementors in developing systems that fit the needs of the MMHS. SPs are produced not simply to ‘legitimize’ a particular choice of base standards and options, but to promote real system interoperability. The development and widespread acceptance of tests based on this and other SPs is crucial to the successful realization of this goal.

FMH22(D) covers information representation by Military Messaging User Agents (MM-UA) and Military Messaging Message Stores (MM-MS). It specifies support of the CSP-specific attributes defined in ACP 120.

FMH22(D) contains one normative appendices:

Appendix A	SPICS Requirements List for FMH22(D)

Information technology – Standardized Profiles – Military Message Handling Systems – Message Store Attributes FMH22(D) – CSP-specific MS Attributes

1 Scope

1.1 General

FMH22(D) covers the representation of information, specifically MS attributes, by MM-MSs and an MM-UAs.

1.2 Position within the taxonomy

FMH22(D) specifies the profile which states requirements for CSP-specific MS attributes.

1.3 Scenario

The model used is one of information representation of CSP-specific MS attributes by the MM-MS and MM-UA (see figure 1).

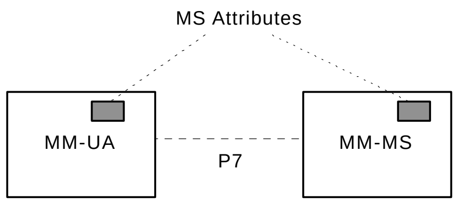

Figure 1 – FMH22(D) Scenario

There are no Open System Interconnection (OSI) upper layer services and protocols within the scope of the FMH22(D) profile.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of FMH22(D). At the time of publication, the editions indicated were valid. All documents are subject to revision, and parties to agreements based on FMH22(D) are warned against automatically applying any more recent editions of the documents listed below, since the nature of references made by SPs to such documents is that they may be specific to a particular edition. Members of IEC and ISO maintain registers of currently valid International Standards and ISPs, and CCITT maintains published editions of its current Recommendations.

NOTE – References in the body of FMH22(D) to specific clauses of ISO/IEC documents shall be considered to refer also to the corresponding clauses of the equivalent CCITT Recommendations (as noted below) unless otherwise stated.

ACP 120: *Common Security Protocol, May 1997.*

ISO/IEC 10021-1: 1990, *Information technology – Text Communication – Message-Oriented Text Interchange Systems (MOTIS) – Part 1: Service Overview. [see also CCITT Recommendation X.400(1992)]*

ISO/IEC 10021-2: 1990, *Information technology – Text Communication – Message-Oriented Text Interchange Systems (MOTIS) – Part 2: Overall Architecture. [see also CCITT Recommendation X.402(1992)]*

ISO/IEC 10021-5: 1990, *Information technology – Text Communication – Message-Orientated Text Interchange Systems (MOTIS) – Part 5: Message Store: Abstract Service Definition. [see also CCITT Recommendation X.413 (1992)]*

CCITT Recommendation X.400(1992), *Message handling system and service overview.*

CCITT Recommendation X.402(1992), *Message handling systems: Overall architecture.*

CCITT Recommendation X.413(1992), *Message handling systems: Message store: Abstract service definition.*

(Application for copies of these documents should be addressed to the American National Standards Institute, 11 West 42nd Street, NY, NY 10036 or to ISO, Van Demonstrate 94, 1013 CN Amsterdam, Netherlands.)

3 Definitions

For the purposes of FMH22(D), the following definitions apply. Terms used in FMH22(D) are defined in the referenced base standard. In addition, the following terms are defined.

3.1 General

MM base standard : the base standard referred to in this profile is ACP 120.

Basic requirement : a CSP-specific MS attribute which is required to be supported by all MMHS implementations conforming to this SP.

3.2 Support classification

To specify the support level of features for FMH22(D), the following terminology is defined.

3.2.1 Static capability

The following classification is used in FMH22(D) to specify static conformance requirements – i.e., capability.

mandatory support (**m**) : the element or feature shall be supported. An implementation shall be able to generate the element, and/or receive the element and perform all associated procedures (i.e., implying the ability to handle both the syntax and semantics of the element) as relevant, as specified in the MM base standard. Where support for generation and reception are not distinguished, then both capabilities shall be assumed.

optional support (**o**) : an implementation is not required to support the element. If support is claimed, the element shall be treated as if it were specified as mandatory support. If support for generation is not claimed, then the element is not generated. If support for reception is not claimed, then an implementation may ignore the element on reception, but will not treat it as an error.

4 Abbreviations and Acronyms

ACP	Allied Communication Publication

CCITT	International Telephone and Telegraph Consultative Committee
IEC	International Electrotechnical Commission
IO-ICS	Information Object Implementation Conformance Statement
ISO	International Standards Organization
ISP	International Standardized Profile
ITU	International Telecommunications Union
ITU-T	ITU Telecommunications Standardization Sector
MHS	Message Handling Systems
MM	Military Message
MMHS	Military Message Handling System
MM-	Military Messaging
MM-MS	Military Messaging Message Store
MM-UA	Military Messaging User Agent
MS	Message Store
MOTIS	Message-Oriented Text Interchange Systems
MTS	Message Transfer System
OSI	Open Systems Interconnection
SP	Standardized Profiles
SPICS	Standardized Profiles Implementation Conformance Statement
UA	User Agent

Support level for protocol elements and features (see clause 3.2):

m	mandatory full support
o	optional support

5 Conformance

The scope of conformance to profile FMH22(D) covers MM-MSs and MM-UAs only. Conformance to profile FMH22(D) does not imply the provision of a standard OSI communications protocol for access to the Message Transfer System (MTS).

FMH22(D) states requirements upon implementations to achieve interworking. A claim of conformance to FMH22(D) is a claim that all requirements in the relevant base standard are satisfied, and that all requirements in the following clauses and in appendix A of FMH22(D) are satisfied. Appendix A states the relationship between these requirements and those of the base standard.

5.1 Conformance statement

For each implementation claiming conformance to profile FMH22(D), an Information Object Implementation Conformance Statement (IO-ICS) shall be made available stating support or non-support of each option identified FMH22(D). The IO-ICS Proforma in annex C of ACP 120 shall be used to generate the IO-ICS.

Appendix A to Annex D
(normative)

SPICS REQUIREMENTS LIST FOR FMH22(D)

In the event of a discrepancy becoming apparent in the body of FMH22(D) and the tables in this appendix, this appendix is to take precedence.

This appendix specifies the support constraints and characteristics of FMH22(D) on what shall or may appear in the implementation columns of an IO-ICS.

Clause A.1 specifies the basic requirements for conformance to profile FMH22(D) (reference numbers correspond to items in the IO-ICS).

In each table, the "Profile" column reflects the level of support required for conformance to this SP (using the classification and notation defined in clause 3.2).

In each table, the "References" column includes references to FMH11(D).

A.1 Basic requirements

A.1.1 CSP-specific Attributes

Ref	Element	Profile		References
		UA	MS	
1	securityServices	o	m	
2	certificatePaths	o	m	
3	contentDescription	m	m	
4	encapsulatedContentType	o	m	
5	mLEnpansionHistory	o	m	
6	organizationMessage	o	m	
7	sCID	o	m	
8	mlExpansionHistory	o	m	
9	signedRequest	o	m	