T/3499TL/2778/9

27 January 1997

Copy No.: ____

CLOUD COVER
CONFIDENTIALITY KEY INFRASTRUCTURE
PART 2: CKI KEY MANAGEMENT PROTOCOL

ISSUE 0.A

FOREWORD

This paper is issued by the Communications-Electronics Security Group (CESG) of Government Communications Headquarters as part of its responsibility to advise HMG on Electronic Information Systems Security (Infosec).

It suggests an architecture for a public key infrastructure (PKI) to support confidentiality between communicating systems. The paper forms part of a suite of documents which collectively provide advice on the implementation of a PKI, and the use of the services enabled by such an infrastructure (eg electronic mail). The architecture as described in the paper is an initial attempt at defining a PKI, and CESG will take into account any comments on its feasibility.

Technical correspondence in connection with this document should be addressed to:

Communications-Electronics Security Group (X27)
Government Communications Headquarters
PO Box 144
Cheltenham GL52 5UE
United Kingdom

AMENDMENT RECORD		
Issue	**Date**	**Description**
Initial draft	6 December 1996	
0.A	27 January 1997	Reformat & release for internal review

CONTENTS

REFERENCES

[HMG] Securing Electronic Mail within HMG - Part I: Infrastructure and Protocol, Draft C, T/3113TL/2776/11 21st March 1996

[DH76] New Directions in Cryptography, IEEE Trans. In Information Theory IT-22 (1976) pages 644-655 W. Diffie and M. Hellman

[RHC] A proposed Architecture for Trusted Third Party Services, N. Jefferies, C. Mitchell, M. Walker, Information Security Group, Royal Holloway

[PKI-1] Internet Public Key Infrastructure Part I: X.509 Certificate and CRL Profile, June 1996, Internet Draft

[PKI-3] Internet Public Key Infrastructure Part III: Certificate Management Protocols, November 1996, Internet Draft

[RFC 793] "Transmission Control Protocol", J. Postel, 09/01/1981

[RFC 822] "Standard for the format of ARPA Internet text messages", D. Crocker, 08/13/1982

[RFC 1521] "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", N. Borenstein, N. Freed, 09/23/1993

[X.214] ITU-T X.214 (95) | ISO/IEC 8072:1996 Information technology - Open systems interconnection - Transport service definition

[X.420] ITU-T X.420 (to be published) | ISO 10021-7 Information technology - Message Handling Systems (MHS) - Interpersonal Messaging System

Note: this is equivalent to X.420(92) plus implementor's guide version 8.

[X.500] ITU-T Recommendation X.500 to X.525 (1993) | ISO/IEC 9594:1994, Information technology – Open Systems Interconnection – The Directory

[X.509DAM] Final Text of Draft Amendments DAM 4 to ISO/IEC 9594-2, DAM 2 to ISO/IEC 9594-6, DAM 1 to ISO/IEC 9594-7, and DAM 1 to ISO/IEC 9594-8 on Certificate Extensions ISO/IEC JTC 1/SC 21/WG 4 and ITU-T Q15/7 Collaborative Editing Meeting on the Directory, Geneva, April 1996 - Final draft 30th June 1996

[X.509TC] Technical Corrigenda to Rec. X.500 | ISO/IEC 9594 resulting from Defect Reports 9594/128

[X.509] ITU-T X.509 (93) | ISO/IEC 9594-8: 1995 Information Technology – Open Systems Interconnection – The Directory: Authentication Framework

[X.511] ITU-T X.511 (93) | ISO/IEC 9594-3: 1995 Information Technology – Open Systems Interconnection – The Directory: Abstract Service Definition

[X.690] ITU-T X.690 (94) | ISO/IEC 8825-1:1995 Information Technology Information technology- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

DEFINITIONS

The following terms and associated concepts are described in the CKI Architecture and Concept of Operation (Part 1):

a. Certificate Management Authority (CMA)

b. CKI certificate

c. CKI User Agent (CKI UA)

d. CMA certificate

e. Domain

f. Domain certificate

g. Domain public / private key

h. External

i. Interoperability key

j. Local

k. Name constraints

l. Receive certificate

m. Receive public / private key

n. Recipient

o. Revoked user list

p. Seed key

q. Seed key identifier

r. Send certificate

s. Send public / private key

t. Sender

u. Shared secret key

v. Top Level Certificate Management Authority (TLCMA)

X.509 Authentication Framework Definitions

The following terms are defined in the X.509 Authentication Framework [X.509]:

a. Certification Authority (CA)

b. Certificate

c. CA Certificate

d. Certificate Revocation List (CRL)

ABBREVIATIONS

AKI	Authentication Key Infrastructure
CA	Certification Authority
CKI	Confidentiality Key Infrastructure
CMA	Certificate Management Authority
CRL	Certificate Revocation List
PKI	Public Key Infrastructure
TLCMA	Top Level Certificate Management Authority
UA	User Agent

I. INTRODUCTION

1. This document specifies the key management protocols for a Confidentiality Key Infrastructure (CKI).

2. The CKI uses asymmetric cryptographic techniques in the generation of a shared symmetric key for confidentiality.

3. This specification is part 2 of a set of specifications for the CKI, which includes:

Part 1: Architecture and concept of operation for the CKI;

Part 2: CKI key management protocol;

Part 3: Profile for the use of X.509 certificates in support of the CKI;

Part 4: Schema for the use of an X.500 directory in support of the CKI;

Part 5: Mapping of the CKI key management protocol onto communication and messaging protocols.

4. The use of X.500 directories is an optional part of the CKI.

5. The CKI is based on the Diffie-Hellman key agreement mechanism [DH76] with support of trusted third party services [RHC].

6. The CKI was initially developed to support secure electronic mail within and between UK government departments [HMG]. However, it is designed to be applicable to a range of application and communication services, and can be used to support confidentiality for governmental, commercial or any other type of organisation.

7. The CKI supports the management of confidentiality keys. It forms part of a public key infrastructure which can also incorporate an infrastructure for the management of authentication keys (called the Authentication Key Infrastructure - AKI). The AKI can be used to provide certified keys for signing CKI certificates and protecting protocol exchanges required for the CKI.

8. The design of the CKI takes account of the ongoing development of standards for public key infrastructures as they exist at the time this specification was developed (e.g. Internet PKI as defined in [PKI-1] and [PKI-3]).

9. These protocols extend the Internet public key infrastructure certificate management protocols [PKI-3]. This document repeats those parts of the Internet PKI protocol required to support the CKI.

II. GENERAL PROTOCOL EXCHANGE STRUCTURE

A. Use of Internet PKI Message Structure

10. This protocol is defined in terms of protocol exchanges which are carried in the **PKIMessage** structure defined in the Internet public key infrastructure (PKI) certificate management protocols [PKI-3].

11. The PKI management protocols operate using a set exchanges of **PKIMessages** which either carry a related "request" / "response" sequence, or a single message to "announce" data.

12. The use of an underlying messaging or communication service to transport **PKIMessages** is specified in Part 5.

13. The **PKIMessage** structure (as defined in [PKI-3]) is:

```
PKIMessage ::= SEQUENCE {
       header          PKIHeader,
       body            PKIBody,
       protection   [0]  PKIProtection OPTIONAL,
       extraCerts   [1] SEQUENCE OF Certificate OPTIONAL
            }
```

14. The PKI message header (as defined in [PKI-3]) contains the following information:

```
PKIHeader ::= SEQUENCE {
      pvno                INTEGER {ietf-version1 (0),
                         cki-version1 (20),
                         aki-version1 (30),
                         cki-aki-version1 (40)},
      sender               GeneralName,
      -- identifies the sender
      recipient            GeneralName,
      -- identifies the intended recipient
      messageTime     [0] GeneralizedTime        OPTIONAL,
      -- time of production of this message (used when sender
      -- believes that the transport will be "suitable"; i.e.,
      -- that the time will still be meaningful upon receipt)
      protectionAlg   [1] AlgorithmIdentifier    OPTIONAL,
      -- algorithm used for calculation of protection bits
      senderKID       [2] KeyIdentifier          OPTIONAL,
      recipKID        [3] KeyIdentifier          OPTIONAL,
      -- to identify specific keys used for protection
      transactionID   [4] OCTET STRING           OPTIONAL,
      -- identifies the transaction, i.e. this will be the same in
      -- corresponding request, response and confirmation messages
      senderNonce     [5] OCTET STRING           OPTIONAL,
      recipNonce      [6] OCTET STRING           OPTIONAL,
      -- nonces used to provide replay protection, senderNonce
      -- is inserted by the creator of this message; recipNonce
      -- is a nonce previously inserted in a related message by
      -- the intended recipient of this message
      freeText        [7] PKIFreeText            OPTIONAL
      -- this may be used to indicate context-specific
```

```
      -- instructions (this field is intended for human
      -- consumption)
  }
  PKIFreeText ::= CHOICE {
      iA5String  [0] IA5String,
      bMPString  [1] BMPString
  }
```

15. The header field **pvno** indicates the protocol version number and the use of the protocol to carry Internet PKI and / or CKI protocol exchanges as follows:

a. **ietf-version1 (0)** indicates that Internet PKI protocol exchanges are to be used as defined in [PKI-3];

b. **cki-version1 (20)** indicates that CKI protocol exchanges alone are to be used;

c. **aki-version1 (30)** indicates that the Authentication Key Infrastructure (AKI) protocol exchanges alone are to be used.

d. **cki-aki-version1 (40)** indicates that the CKI and AKI protocol exchanges are to be used.

Note: The Authentication Key Infrastructure is defined separately.

The need for separate protocol identifiers needs to be reviewed once the future standardisation of PKI protocols has stabilised.

16. The PKI message field **PKIBody** is used to carry the content of the protocol exchange. The syntax for content used in the CKI is as follows:

```
PKIBody ::= CHOICE {
-- As defined in [PKI-3]
     krr       [6]     KeyRecReqContent,
     krp       [7]     KeyRecRepContent,
       crlann    [15]   CRLAnnContent,
-- Specific to CKI
     ckiTrr    [30]    CKITlcmaReqContent,
     ckiTrp    [31]    CKITlcmaRepContent,
     ckiTann   [32]    CKITlcmaAnnContent,
     ckiPrr    [33]    CKIPeerCmaReqContent,
     ckiPrp    [34]    CKIPeerCmaRepContent,
     ckiESrr   [35]    CKIExtSeedKeyReqContent,
     ckiESrp   [36]    CKIExtSeedKeyRepContent,
     ckiESann  [37]    CKIExtSeedKeyAnnContent,
     ckiSCrr   [38]    CKISendCertReqContent,
     ckiSCrp   [39]    CKISendCertRepContent,
     ckiSCann  [40]    CKISendCertAnnContent,
     ckiRCrr   [41]    CKIRecCertReqContent,
     ckiRCrp   [42]    CKIRecCertRepContent,
     ckiRCann  [43]    CKIRecCertAnnContent,
     ckiRULann [44]    CKIRevokedUserListAnnContent,
     ckiBindrr [45]    CKIBindReqContent,
     ckiBindrp [46]    CKIBindRepContent }
```

B. Protocol Exchange Protection

17. As stated later in this specification certain CKI protocol exchanges require integrity and authentication, and also in some cases confidentiality, protection.

18. The PKI message field **PKIProtection** is not used in the CKI. However, it is included in this specification for completeness:

```
PKIProtection ::= BIT STRING
```

19. The required security services are provided by the underlying messaging or communications service as described in Part 5 (for example, using secure messaging as defined in [HMG]).

20. Additional protection may be applied as required under the security policy in force.

C. Use of PKI Status

21. All the response messages include status information carried in a **PKIStatusInfo** the field (as defined in [PKI-3]) structured as follows:

```
PKIStatusInfo ::=SEQUENCE{
     status      PKIStatus,
     failInfo    PKIFailureInfo    OPTIONAL }
```

22. The **PKIStatus** field is used to carry a status code as follows (as defined in [PKI-3]):

```
PKIStatus ::= INTEGER {
     granted     (0),  -- you got exactly what you asked for
     grantedWithMods   (1),
       -- you got something like what you asked for; the
       -- requester is responsible for ascertaining the differences
       -- possible reasons for CKI protocol functions are given
       --  later in this specification
     rejection   (2),
           -- you don't get it, more information elsewhere in the
           -- message
     waiting     (3)
            -- the request body part has not yet been processed,
            -- expect to hear more later
     -- Other values in [PKI-3] are not required for the CKI
  }
```

23. The **PKIFailureInfo** field may be used to provide more information about failure cases, as follows:

```
PKIFailureInfo ::= BIT STRING {
    -- since we can fail in more than one way!
     badAlg            (0),
      badMessageCheck         (1),
     -- Bits 2-7 reserved for PKI use
```

```
    -- CKI Specific failure codes
    unrecognisedDomain            (8),
     -- CMA or user name not for a recognised domain
    interoperabilityKeyUnavailable (9),
     -- Interoperability key required to generate certificate
     -- or seed key not available.
    unauthorisedAccess            (10),
     -- CKI User and CMA not authorised to access CMA or TLCMA
    invalidRequestParameters (11),
    noKeyAvailable          (12),
     -- No key is available which matches the key recovery request
    invalidCredentials            (13),
    userInMoreThanOneCMADomain (14),
     -- A user is in the name constraints for more than one CMA
    alternativeBaseAndModulusRequired (15),
    userInRevokedUserList   (16)
}
```

III. CKI PROTOCOL FUNCTIONS

A. Obtain Domain Parameters from TLCMA

Overview

24. This protocol exchange is used by a CMA to request from the TLCMA interoperability keys and other parameters required for operation within and between domains, or by a TLCMA to announce to a CMA new interoperability keys.

25. This protocol exchange can be used to establish external interoperability keys for domains that trust the TLCMA to provide keys.

26. An alternative mechanism to establish external interoperability keys using CMA to CMA exchanges is described in §III.B. This alternative is for use where both domains do not trust the TLCMA.

27. The parameters that can be obtained by a CMA from a TLCMA in this protocol exchange are:

a. Base and modulus for the local domain;

b. Interoperability key for the local domain;

c. CMA certificate for the local domain;

d. The local domain private key and domain certificate for subsequent use in generating external interoperability keys using peer CMA to CMA exchanges (see §III.B);

e. Interoperability keys for use with external domains;

f. A "my" (local) and "your" (external domain) reference for each interoperability key.

g. The external CMA's certificate.

h. An initial revoked user list for the local and external domain;

Note: Subsequent changes to the revoked user list are announced as described in §III.G.

28. Domain(s) may be identified in a request by either (but not both):

a. the distinguished name of the CMA responsible for that domain, or

b. the distinguished name of a user within that domain.

29. A domain is identified in a response or announce by the name of the CMA responsible for the domain.

30. An announce protocol exchange may be initiated by the TLCMA on an event including:

a. Replacement of an interoperability key required, for example, due to the imminent expiry of the current key;

b. A change to the base and modulus required.

31. A request / response protocol exchange may be initiated by the CMA on an event including:

a. Initialisation of a CMA;

b. New inter-domain communications being established;

c. Replacement of an interoperability key required, for example, due to the imminent expiry of the current key;

Procedure

32. The CMA sends a **`CKITlcmaReq`** message to the TLCMA indicating the domains for which interoperability keys are required.

33. The TLCMA responds with the requested interoperability keys and associated parameters in a **`CKITlcmaRep`** message.

34. Unsolicited changes to a local or external interoperability key are announced by the TLCMA using a **`CKITlcmaAnn`** message.

35. If considered necessary the TLCMA may respond with interoperability keys for domains not requested by the CMA.

36. An interoperability key and associated domain parameters comes into effect at the indicated start time for the interoperability key.

37. The validity period requested by a CMA need not be the validity period applied by the TLCMA. However, it is recommended that the validity period covers a period greater than or equal to the requested validity period, unless required by the security policy in force.

38. Protocol exchanges containing **`CKITlcmaRep`** and **`CKITlcmaAnn`** shall be protected by mechanisms providing data origin authentication, integrity and confidentiality.

> **Note:** Protection may be applied directly to the interoperability key and any domain private keys instead of, or as well as, the confidentiality of the underlying service. The mechanism used is outside the scope of the CKI.

39. **`PKIStatus`** is set to **`grantedWithMods`** if:

a. Interoperability keys could be provided for some of the identified external domains (e.g. as the named domain is not known or an interoperability key is not available);

b. The validity period for one or more interoperability keys is not as requested.

40. Possible failure flags for this exchange include:

    a. Unrecognised domain

    b. Interoperability key unavailable

    c. Unauthorised access

    d. Invalid request parameters

Protocol Exchange Content Format

41. TLCMA request:

```
CKITlcmaReqContent ::= SEQUENCE {
     reqLocalIK        BOOLEAN,
           -- Whether new local interoperability key
           --  and other local information required
     localIKValidity   [0]   IKValidity  OPTIONAL,
           -- Requested validity period for local IK
     externalRqInfo    [1] ExternalRqInfo OPTIONAL
       -- Information about external domains and IK requirements
  }

IKValidity ::= Validity      -- As defined in [X.509]

ExternalRqInfo ::= SEQUENCE OF SEQUENCE {
     CHOICE {
           externalCMA [0]   CMAName,
           externalUser      [1]   Name    },
     externalIKValidity      [2]   IKValidity  OPTIONAL}

CMAName ::= Name
```

42. TLCMA response:

```
CKITlcmaRepContent ::= SEQUENCE {
     status            PKIStatusInfo,
     localDomainInfo   [0]   LocalDomainInfo   OPTIONAL,
     externalDomainInfo      [1]   ExternalDomainInfo      OPTIONAL }

LocalDomainInfo ::= SEQUENCE {
     localBaseMod      BaseModInfo,
     localIKInfo IKInfo,
     localCMACertificate [0] Certificate OPTIONAL,
           -- present if directory or PKI not used
           -- to distribute the CMA's certificate
     localDomainCertificateAndPrivateKey
                 [1] CertificateAndPrivateKey OPTIONAL,
           -- Present if external interoperability keys are
           -- established by peer CMA to CMA exchange
     localRevokedUL    [2] RevokedUserList }

CertificateAndPrivateKey ::= SEQUENCE{
     certificate [1]   Certificate,
     privateKey  [2]   EncryptedValue
```

```
  }
 -- EncryptedValue is defined later under Key recovery

ExternalDomainInfo ::= SEQUENCE OF SEQUENCE {
     externalCMA CMAName,
     externalIKInfo    IKInfo,
     externalRevokUL   RevokedUserList,
     externalCMACertificate  [0] Certificate   OPTIONAL
           -- present if AKI not used
           -- to distribute the CMA's certificate
 }

BaseModInfo ::= CHOICE {
     baseModRef  [0]   BaseModRef,
     baseModValue      [1]   BaseModValue }
BaseModRef       ::= OBJECT IDENTIFIER
BaseModValue ::= SEQUENCE {
     modulus     INTEGER,
     base        INTEGER }

IKInfo ::= SEQUENCE {
     myIKRef     [0]   IKReference,
     yourIKRef   [1]   IKReference OPTIONAL,
           -- Not present in localIKInfo
     validity    [2]   IKValidity,
     ikValue     [3]   IKValue }

IKReference ::=OCTET STRING
IKValue ::= BIT STRING       -- Encrypted value of interoperability key
     --  The interoperability key may be protected by a mechanism
     --  outside the scope of this specification.

RevokedUserList ::= SEQUENCE OF SEQUENCE {
     user        Name,
     reason      RevokeUserReason }
     -- If no user is revoked then an empty sequence is sent.
RevokeUserReason ::= ENUMERATED {
     unspecified (0),
     keyCompromise     (1),
     userBlackListed   (2),
     affiliationChanged (3) }
```

43. TLCMA announce:

```
CKITlcmaAnnContent ::= SEQUENCE {
     localDomainInfo   [0]   LocalDomainInfo   OPTIONAL,
     externalDomainInfo      [1]   ExternalDomainInfo      OPTIONAL }
```

B. Peer CMA to CMA Exchange

Overview

44. This protocol exchange supports the establishment between peer CMAs of interoperability keys and other parameters required for inter-domain operation.

45. An alternative mechanism of obtaining domain parameters using a TLCMA is described in §III.A for use where the TLCMA is trusted by both parties.

46. The parameters that can be established between CMAs using this protocol exchange are:

a. Domain public keys to create an interoperability key for use between the peer domains;

b. A "my" and "your" reference to the interoperability key;

c. The peer CMA certificate.

d. An initial revoked user list for the peer domain;

Note: Subsequent changes to the revoked user list are announced as described in §III.G.

47. The protocol exchange is addressed to the peer CMA responsible for the peer domain.

48. The interoperability key is created using the private key and public key values for each domain fed into a Diffie-Hellman based algorithm as described in Part 1 (CKI architecture).

49. The domain public key is exchanged between CMAs in a domain certificate.

50. The validity period for the interoperability key is the intersection of the validity of the domain certificates for the two domains.

51. The base and modulus to be used in generating the interoperability key is carried (either by reference or value) in the domain certificate. If different base and modulus values are provided by the peer CMAs then the protocol exchange fails.

Note: It is presumed that the base and modulus to be used between domains is established by prior agreement and pre-configured into a CMA.

A sub-string of the generated value may be used as the interoperability key by prior agreement and pre-configured into a CMA.

52. Where requests from both parties for the establishment of inter-domain parameters occur simultaneously (or within the period of the expected network round-trip delay), the same parameter values should be used in both exchanges (i.e. in the request and response from a CMA).

Procedure

53. A CMA initiates the establishment of domain parameters by sending a **`CKIPeerCmaReq`** message to its peer containing its parameter values.

54. The peer CMA replies with its parameters in a **`CKIPeerCmaRep`** message.

55. Protocol exchanges containing **`CKIPeerCmaReq`** and **`CKIPeerCmaRep`** shall be protected by mechanisms providing data origin authentication and integrity.

56. **PKIStatus** should not be set to **grantedWithMods** for this protocol exchange

57. Possible failure flags for this exchange include:

    a. Unauthorised access

    b. Invalid request parameters

    c. Alternative base and modulus required

Protocol Exchange Content Format

58. Peer CMA exchange request:

```
CKIPeerCmaReqContent ::= PeerExchangeInfo
PeerExchangeInfo ::= SEQUENCE {
     domainCertificate       Certificate,
     myIKRef           IKReference,
     revokedUsers            RevokedUserList,
     externalCMACertificate  [0] Certificate   OPTIONAL
           -- present if directory or PKI not used
           -- to distribute the CMA's certificate
}
```

59. Peer CMA exchange response:

```
CKIPeerCmaRepContent ::= SEQUENCE {
     status            PKIStatusInfo,
     peerExchangeInfo        [0] PeerExchangeInfo OPTIONAL }
```

C. Obtain External Seed Key

Overview

60. This protocol exchange is initiated by a user to request, from its CMA, an external seed key for communication with an external domain, or by a CMA to announce a new external seed key following the establishment of a new external interoperability key.

61. A request for an external seed key from a user may be initiated by an event including receipt of a certificate containing an unknown external seed key identifier.

62. An announcement of an external seed key from a CMA may be initiated by an event including the establishment of a new interoperability key and user known to be in frequent communication with an external domain.

> **Note:** A list of users frequently communicating with external domains may be established by local management.

63. The external CMA's certificate is passed with the external key for use in verifying the send and receive certificates.

Procedure

64. A user requests a seed key by passing to its CMA a **CKIExtSeedKeyReq** message containing the receive certificate for which an associated external seed key is required.

65. The CMA validates that the authenticated originator of the request is the user named in the certificate or an entity authorised to act on behalf of the user.

66. The CMA validates the receive certificate.

67. The CMA creates the external seed key using the appropriate external interoperability key with the name and seed key in the receive certificate.

68. The external seed key validity period ends at end of the validity period of the interoperability key from which the seed key was generated.

69. The CMA sends the protected seed key and validity period to the user in a **CKIExtSeedKeyRep** message if in response to a request, or in a **CKIExtSeedKeyAnn** message if unsolicited.

70. Protocol exchanges containing **CKIExtSeedKeyReq** shall be protected by mechanisms providing data origin authentication and integrity.

71. Protocol exchanges containing **CKIExtSeedKeyAnn** and **CKIExtSeedKeyRep** shall be protected by mechanisms providing data origin authentication, integrity and confidentiality.

> **Note:** Protection may be applied directly to the external seed key instead of, or as well as, the confidentiality of the underlying service. The mechanism used is outside the scope of the CKI.

72. **PKIStatus** should not be set to **grantedWithMods** for this protocol exchange.

73. Possible failure flags for this exchange include:

    a. Unrecognised domain

    b. Unauthorised access

    c. Invalid request parameters

    d. Interoperability key unavailable

Protocol Exchange Content Format

74. External seed key request:

```
CKIExtSeedKeyReqContent ::=  Certificate
```

75. External seed key response:

```
CKIExtSeedKeyRepContent ::= SEQUENCE {
    status      PKIStatusInfo,
    extSeedKeyInfo    [0] ExtSeedKeyInfo      OPTIONAL,
    externalCMACertificate  [1] Certificate   OPTIONAL
```

```
          -- present if directory or PKI not used
          -- to distribute the CMA's certificate
}

ExtSeedKeyInfo ::= SEQUENCE {
     externalCMA CMAName,
     user        Name,
     seedKey     BIT STRING,
--   The seed key may be protected by a mechanism
--   outside the scope of this specification.
     seedKeyValidity   IKValidity }
```

76. External seed key announce:

```
CKIExtSeedKeyAnnContent ::= SEQUENCE {
     extSeedKeyInfo    [0] ExtSeedKeyInfo,
     externalCMACertificate  [1] Certificate   OPTIONAL
           -- present if directory or PKI not used
           -- to distribute the CMA's certificate
   }
```

D. Obtain Send Certificate

Overview

77. This protocol exchange is initiated by a user to request a send certificate from its CMA, or by a CMA to announce a new send certificate.

78. A request to obtain a send certificate may be initiated by an event including:

a. The receipt of a new local seed key,

b. The pending expiry of an existing send certificate.

79. An announcement of new send certificate from a CMA may be initiated by an event, including the pending expiry of an existing send certificate.

Procedure

80. A user requests a send certificate by sending the CMA a **`CKISendCertReq`** message containing the user's name, a suggested validity start time and optionally a suggested validity end time.

81. The CMA generates the send certificate for the user. The validity may differ from that suggested by the user.

82. The CMA sends the send certificate in a **`CKISendCertRep`** message, if in response to a request, or a **`CKISendCertAnn`** message if unsolicited.

83. No security services are mandated for this protocol exchange.

84. **PKIStatus** is set to **grantedWithMods** if the certificate validity period is different from the suggested validity start time, or if provided, validity end time.

85. Possible failure flags for this exchange include:

    a. Unauthorised access

    b. Invalid request parameters

    c. Interoperability key unavailable

Protocol Exchange Content Format

86. Send certificate request:

```
CKISendCertReqContent ::= SEQUENCE {
     user              Name,
     suggestedValidityStart  GeneralizedTime,
     suggestedValidityEnd    GeneralizedTime   OPTIONAL }
```

87. Send certificate response:

```
CKISendCertRepContent ::= SEQUENCE {
     status      PKIStatusInfo,
     sendCertificate   SendCertificate OPTIONAL }

SendCertificate ::= Certificate
```

88. Send certificate announce:

```
CKISendCertAnnContent ::=    SendCertificate
```

E. Obtain Receive Certificates

Overview

89. This protocol exchange is initiated by a user to request receive certificates from its CMA, or by a CMA to announce new receive certificates.

90. A request for receive certificates may be initiated by an event including:

    a. When a message is to be sent to an external domain,

    b. The pending expiry of an existing receive certificate.

91. An announcement of new receive certificates from a CMA may be initiated by an event, including the pending expiry of an existing receive certificate.

Procedure

92. A user requests receive certificates by sending the CMA a **CKIRecCertReq** message containing a list of recipient user names, a suggested validity start time and optionally a suggested validity end time.

93. The user's home domain is identified using the name constraints held in the local and external CMA certificates. If the name fits in the name constraints for two or more CMAs then the request fails.

94. If the user is in the revoked user list then the request fails.

95. The CMA generates receive certificates for each recipient user. The validity may differ from that suggested. The CMA may use certificates that have been generated from a previous request.

96. The CMA sends the receive certificates in a **CKIRecCertRep** message if in response to a request, or in a **CKIRecCertAnn** message if unsolicited.

97. No security services are mandated for this protocol exchange.

98. **PKIStatus** is set to **grantedWithMods** if the certificate validity period is different from the suggested validity start time, or if provided, validity end time.

99. Possible failure flags for this exchange include:

    a. Unrecognised domain

    b. Unauthorised access

    c. Invalid request parameters

    d. Interoperability key unavailable

    e. User in more than one CMA domain

    f. User in revoked user list

Protocol Exchange Content Format

100. Receive certificate request:

```
CKIRecCertReqContent ::= SEQUENCE {
     recipientUsers          SEQUENCE OF Name,
     suggestedValidityStart  GeneralizedTime,
     suggestedValidityEnd    GeneralizedTime   OPTIONAL }
```

101. Receive certificate response:

```
CKIRecCertRepContent ::= SEQUENCE {
     status      PKIStatusInfo,
     receiveCerts      SEQUENCE OF Certificate OPTIONAL }
```

102. Receive certificate announce:

```
CKIRecCertAnnContent ::= SEQUENCE OF Certificate
```

F. CRL Distribution

Overview

103. This protocol exchange is used to distribute certificate revocation lists (CRLs):

a. From a TLCMA to CMAs

b. Between CMAs

c. From a CMA to all users in its domain.

104. This protocol exchange may be used to distribute CRLs to support either the AKI or the CKI. Furthermore, CRLs containing CA certificate (authority) revocation lists may be distributed by this protocol.

105. The use of CRLs in the confidentiality key infrastructure is described in Part 1 - CKI Architecture.

106. This protocol exchange is the same as used for CRL publication in [PKI-3].

Procedure

107. The CRLs to be distributed are placed in **`CLRAnnContent`**.

108. No security services are mandated for this protocol exchange.

Protocol Exchange Content Format

109. Distribute CRL:

```
CRLAnnContent ::= SEQUENCE OF CertificateList
```

G. Revoked User List

Overview

110. This protocol exchange is used by a CMA, or a TLCMA, to inform another CMA that a user key is suspected to have been compromised or that the user has been blacklisted.

111. Any unexpired certificates for the revoked user should be revoked and no new certificates generated for the user until the user is removed from the revocation list.

112. An initial list of revoked users is distributed when a new interoperability key is established either by a TLCMA or peer to peer CMA exchange . This list is updated by the new list distributed by this protocol exchange.

Procedure

113. On detection that a user's seed key in a domain has been compromised, or when a user is blacklisted, then that user's CMA, or the TLCMA, should send a revised list of revoked users in that domain to all CMAs with which an interoperability key is shared. This list is carried in a **`CKIRevokedUserListAnn`** message.

114. A CMA receiving such a list shall include any unexpired receive certificates produced for the revoked users in a CRL and distribute the revised CRL to all users in its domain.

115. No further receive certificates shall be produced for that user until a new revoked user list is received which does not include the user. In addition, any current seed key identifier for that revoked user shall not be re-used.

116. Protocol exchanges containing **`CKIRevokedUserListAnn`** shall be protected by mechanisms providing data origin authentication and integrity.

Protocol Exchange Content Format

117. Announce new user revocation list:

```
CKIRevokedUserListAnnContent ::= SEQUENCE {
          domain      CMAName,
          list  RevokedUserList }
```

H. CKI Key Recovery

Overview

118. This protocol exchange is for use by a law enforcement agency or authorised manager to obtain a private key related to a given certified send or receive public key.

119. This protocol uses the Internet PKI key recovery protocol exchange.

120. The use of this protocol exchange differs slightly (without affecting interoperability) from that defined in the Internet PKI key recovery protocol exchange in that:

a. Not all the previous keys for the user need be provided, only that associated with the supplied **`CertTemplate`**.

b. The client for this request is not the key user,

c. Certain optional fields are specifically required or not required for CKI key recovery.

121. Support for this function is an optional part of the CKI.

Procedure

122. Information from the certified public key, for which the related private key is required, is placed in the **`CertTemplate`** field of the **`KeyRecReq`** message. The subject, validity, **`subjectKeyIdentifier`** and **`keyUsage`** fields of **`CertTemplate`** must be present. It is recommended that other fields present in the certificate are also passed in the request.

123. The CMA recreates the required private key value and returns this in the **`keyPairHist`** field of the **`KeyRecRepContent`**.

124. The protocol exchanges containing **`KeyRecReqContent`** shall be protected by mechanisms providing data origin authentication and integrity.

125. Protocol exchanges containing **`KeyRecRepContent`** shall be protected by mechanisms providing data origin authentication, integrity and confidentiality.

> **Note:** Protection may be applied directly to the private keys carried instead of, or as well as, the confidentiality of the underlying service. The mechanism used is outside the scope of the CKI.

126. PKIStatus is set to grantedWithMods if private keys for only some of the certificate templates are returned.

127. Possible failure flags for this exchange include:

a. Unauthorised access

b. Invalid request parameters

c. No key available

Protocol Exchange Content Format

128. CKI key recovery request:

```
KeyRecReqContent ::= InitReqContent

InitReqContent ::= SEQUENCE {
     protocolEncKey  [0] SubjectPublicKeyInfo  OPTIONAL,
                 -- Not required for CKI key recovery
       fullCertTemplates   FullCertTemplates }

FullCertTemplates ::= SEQUENCE OF FullCertTemplate

FullCertTemplate ::= SEQUENCE {
     certReqId   INTEGER,
      -- to match this request with corresponding response
      -- (note:  must be unique over all FullCertReqs in this message)
     certTemplate           CertTemplate
 }
-- Other OPTIONAL fields in FullCertTemplate defined in
-- [PKI-3] are not required for the CKI.
```

```
CertTemplate ::= SEQUENCE {
       version    [0] Version               OPTIONAL,
       serial     [1] INTEGER               OPTIONAL,
       signingAlg [2] AlgorithmIdentifier   OPTIONAL,
       subject    [3] Name                  OPTIONAL,
           -- Required for CKI key recovery
       validity   [4] OptionalValidity      OPTIONAL,
           -- Required for CKI key recovery
       issuer     [5] Name                  OPTIONAL,
       publicKey  [6] SubjectPublicKeyInfo  OPTIONAL,
           -- Required for CKI key recovery
       issuerUID  [7] UniqueIdentifier      OPTIONAL,
       subjectUID [8] UniqueIdentifier      OPTIONAL,
       extensions [9] Extensions            OPTIONAL
           -- Key Usage extension required for CKI key recovery
}

  OptionalValidity ::= SEQUENCE {
       notBefore  [0] GeneralizedTime OPTIONAL,
           -- Required for CKI key recovery
       notAfter   [1] GeneralizedTime OPTIONAL
           -- Required for CKI key recovery
   }
-- Note: UTCTime is used in the current PKI protocol, but it is
-- expected that this will change in line with updates to X.509
```

129. CKI key recovery response:

```
KeyRecRepContent ::= SEQUENCE {
     status            PKIStatusInfo,
     newSigCert  [0]   Certificate OPTIONAL,
     caCerts     [1]   SEQUENCE OF Certificate OPTIONAL,
     keyPairHist [2]   SEQUENCE OF CertifiedKeyPair  OPTIONAL
                       -- Required for CKI key recovery
  }

CertifiedKeyPair ::= SEQUENCE {
       certificate     [0] Certificate         OPTIONAL,
                       -- Required if
                       -- SEQUENCE OF CertifiedKeyPair
                       -- contains >1 element
       privateKey      [2] EncryptedValue    OPTIONAL
                       -- Required for CKI key recovery
     -- Other optional fields in [PKI-3] are not required
 }

  EncryptedValue ::= SEQUENCE {
      encValue          BIT STRING,
      -- the encrypted value itself
      intendedAlg   [0] AlgorithmIdentifier  OPTIONAL,
      -- the intended algorithm for which the value will be used
      symmAlg       [1] AlgorithmIdentifier  OPTIONAL,
      -- the symmetric algorithm used to encrypt the value
      encSymmKey    [2] BIT STRING           OPTIONAL,
      -- the (encrypted) symmetric key used to encrypt the value
      keyAlg        [3] AlgorithmIdentifier  OPTIONAL
      -- algorithm used to encrypt the symmetric key
       }
```

I. Secure Bind

Overview

130. This protocol exchange supports the establishment of a secure bind between communicating CKI entities (CKI UA, CMA and TLCMA) to provide peer entity authentication.

131. This exchange is to be used when

a. the CKI key management protocol is mapped onto a direct peer to peer communication service such as the Internet TCP or the OSI transport service,

b. the required authentication of application entities is not provided by the underlying service.

132. This exchange is to be used immediately following establishment of the underlying connection.

Procedure

133. Once the underlying connection has been established the initiator of the connection sends a **`CKIBindReq`**.

134. The peer entity responds a **`CKIBindRep.`**

135. The Credentials field of **`CKIBindReq`** and **`CKIBindRep`** contains strong credentials as defined in [X.511] clause 8.1.

136. The **`senderNonce`** in the **`PKIHeader`** for this and subsequent protocol exchanges is derived from the value sent in the **`Random`** field of the **`Token`** in the strong credentials (e.g. previous value plus 1).

137. Possible failure flags for this exchange include:

a. Unauthorised access

b. Invalid credentials

Protocol Exchange Content Format

138. Bind request:

```
CKIBindReqContent ::= Credentials

-- The syntax for credentials is as defined in
--  ITU-T X.511 | ISO/IEC 9594-3 clause 8.1.1.
```

139. Bind response:

```
CKIBindRepContent ::= SEQUENCE {
     status      PKIStatusInfo,
     credentials Credentials }
```

Annex A ASN.1 Module

```
CKIKeyManagementProtocol {iso(1) member-body(2) uk(826) disc(0)
  cesg(1145) infosec(1) cki(4) module(1) ckiKeyManagementProtocol(1)}

DEFINITIONS ::=
BEGIN
IMPORTS
   Certificate, CertificateList, Version, AlgorithmIdentifier,
     Validity, SubjectPublicKeyInfo, UniqueIdentifier, Extensions
           FROM AuthenticationFramework {joint-iso-ccitt ds(5)
            module(1) authenticationFramework(7) 2}

   NameConstraintsSyntax, GeneralName, KeyIdentifier FROM
           CertificateExtensions {joint-iso-ccitt ds(5) module(1)
                  certificateExtensions(26) 0}

   Name FROM InformationFramework {joint-iso-ccitt ds(5)
            module(1) informationFramework(1) 2}

   Credentials FROM DirectoryAbstractService {joint-iso-ccitt ds(5)
            module(1) directoryAbstractService(2) 2};


  -- Definitions from the Internet PKI key management protocol
  -- are repeated in this module.


-- Internet PKI Message Structure
-- ______________________________

PKIMessage ::= SEQUENCE {
       header          PKIHeader,
       body            PKIBody,
       protection   [0]  PKIProtection OPTIONAL,
       extraCerts   [1] SEQUENCE OF Certificate OPTIONAL
            }

PKIHeader ::= SEQUENCE {
      pvno                INTEGER {ietf-version1 (0),
                         cki-version1 (20),
                         aki-version1 (30),
                         cki-aki-version1 (40)},
      sender               GeneralName,
      -- identifies the sender
      recipient            GeneralName,
      -- identifies the intended recipient
      messageTime     [0] GeneralizedTime        OPTIONAL,
      -- time of production of this message (used when sender
      -- believes that the transport will be "suitable"; i.e.,
      -- that the time will still be meaningful upon receipt)
      protectionAlg   [1] AlgorithmIdentifier    OPTIONAL,
      -- algorithm used for calculation of protection bits
      senderKID       [2] KeyIdentifier          OPTIONAL,
      recipKID        [3] KeyIdentifier          OPTIONAL,
```

```
       -- to identify specific keys used for protection
      transactionID   [4] OCTET STRING          OPTIONAL,
      -- identifies the transaction, i.e. this will be the same in
      -- corresponding request, response and confirmation messages
      senderNonce     [5] OCTET STRING          OPTIONAL,
      recipNonce      [6] OCTET STRING          OPTIONAL,
      -- nonces used to provide replay protection, senderNonce
      -- is inserted by the creator of this message; recipNonce
      -- is a nonce previously inserted in a related message by
      -- the intended recipient of this message
      freeText        [7] PKIFreeText           OPTIONAL
      -- this may be used to indicate context-specific
      -- instructions (this field is intended for human
      -- consumption)
  }

  PKIFreeText ::= CHOICE {
      iA5String  [0] IA5String,
      bMPString  [1] BMPString
  }


PKIBody ::= CHOICE {
-- As defined in [PKI-3]
     krr       [6]     KeyRecReqContent,
     krp       [7]     KeyRecRepContent,
       crlann    [15]   CRLAnnContent,
-- Specific to CKI
     ckiTrr    [30]    CKITlcmaReqContent,
     ckiTrp    [31]    CKITlcmaRepContent,
     ckiTann   [32]    CKITlcmaAnnContent,
     ckiPrr    [33]    CKIPeerCmaReqContent,
     ckiPrp    [34]    CKIPeerCmaRepContent,
     ckiESrr   [35]    CKIExtSeedKeyReqContent,
     ckiESrp   [36]    CKIExtSeedKeyRepContent,
     ckiESann  [37]    CKIExtSeedKeyAnnContent,
     ckiSCrr   [38]    CKISendCertReqContent,
     ckiSCrp   [39]    CKISendCertRepContent,
     ckiSCann  [40]    CKISendCertAnnContent,
     ckiRCrr   [41]    CKIRecCertReqContent,
     ckiRCrp   [42]    CKIRecCertRepContent,
     ckiRCann  [43]    CKIRecCertAnnContent,
     ckiRULann [44]    CKIRevokedUserListAnnContent,
     ckiBindrr [45]    CKIBindReqContent,
     ckiBindrp [46]    CKIBindRepContent }


PKIProtection ::= BIT STRING

PKIStatusInfo ::=SEQUENCE{
     status      PKIStatus,
     failInfo    PKIFailureInfo    OPTIONAL }

PKIStatus ::= INTEGER {
     granted     (0),  -- you got exactly what you asked for
     grantedWithMods   (1),
       -- you got something like what you asked for; the
       -- requester is responsible for ascertaining the differences
```

```
      -- possible reasons for CKI protocol functions are given
      --  later in this specification
    rejection   (2),
          -- you don't get it, more information elsewhere in the
          -- message
    waiting     (3)
           -- the request body part has not yet been processed,
           -- expect to hear more later
    -- Other values in [PKI-3] are not required for the CKI
  }

 PKIFailureInfo ::= BIT STRING {
    -- since we can fail in more than one way!
    badAlg            (0),
     badMessageCheck       (1),
    -- Bits 2-7 reserved for PKI use

    -- CKI Specific failure codes
    unrecognisedDomain            (8),
     -- CMA or user name not for a recognised domain
    interoperabilityKeyUnavailable (9),
     -- Interoperability key required to generate certificate
     -- or seed key not available.
    unauthorisedAccess            (10),
     -- CKI User and CMA not authorised to access CMA or TLCMA
    invalidRequestParameters (11),
    noKeyAvailable          (12),
     -- No key is available which matches the key recovery request
    invalidCredentials            (13),
    userInMoreThanOneCMADomain (14),
     -- A user is in the name constraints for more than one CMA
    alternativeBaseAndModulusRequired (15),
    userInRevokedUserList   (16)
}


-- Obtain Domain Parameters from TLCMA
-- ___________________________________

CKITlcmaReqContent ::= SEQUENCE {
    reqLocalIK        BOOLEAN,
          -- Whether new local interoperability key
          --  and other local information required
    localIKValidity   [0]   IKValidity  OPTIONAL,
          -- Requested validity period for local IK
    externalRqInfo    [1] ExternalRqInfo OPTIONAL
      -- Information about external domains and IK requirements
  }

IKValidity ::= Validity      -- As defined in [X.509]

ExternalRqInfo ::= SEQUENCE OF SEQUENCE {
    CHOICE {
          externalCMA [0]   CMAName,
          externalUser      [1]   Name    },
    externalIKValidity      [2]   IKValidity  OPTIONAL}
```

```
CMAName ::= Name

CKITlcmaRepContent ::= SEQUENCE {
     status            PKIStatusInfo,
     localDomainInfo   [0]   LocalDomainInfo   OPTIONAL,
     externalDomainInfo      [1]   ExternalDomainInfo      OPTIONAL }

LocalDomainInfo ::= SEQUENCE {
     localBaseMod      BaseModInfo,
     localIKInfo IKInfo,
     localCMACertificate [0] Certificate OPTIONAL,
           -- present if directory or PKI not used
           -- to distribute the CMA's certificate
     localDomainCertificateAndPrivateKey
                 [1] CertificateAndPrivateKey OPTIONAL,
           -- Present if external interoperability keys are
           -- established by peer CMA to CMA exchange
     localRevokedUL    [2] RevokedUserList }

CertificateAndPrivateKey ::= SEQUENCE{
     certificate [1]   Certificate,
     privateKey  [2]   EncryptedValue
  }
 -- EncryptedValue is defined later under Key recovery

ExternalDomainInfo ::= SEQUENCE OF SEQUENCE {
     externalCMA CMAName,
     externalIKInfo    IKInfo,
     externalRevokUL   RevokedUserList,
     externalCMACertificate  [0] Certificate   OPTIONAL
           -- present if AKI not used
           -- to distribute the CMA's certificate
 }

BaseModInfo ::= CHOICE {
     baseModRef  [0]   BaseModRef,
     baseModValue      [1]   BaseModValue }
BaseModRef       ::= OBJECT IDENTIFIER
BaseModValue ::= SEQUENCE {
     modulus     INTEGER,
     base        INTEGER }

IKInfo ::= SEQUENCE {
     myIKRef     [0]   IKReference,
     yourIKRef   [1]   IKReference OPTIONAL,
           -- Not present in localIKInfo
     validity    [2]   IKValidity,
     ikValue     [3]   IKValue }

IKReference ::=OCTET STRING
IKValue ::= BIT STRING       -- Encrypted value of interoperability key
     --  The interoperability key may be protected by a mechanism
     --  outside the scope of this specification.

RevokedUserList ::= SEQUENCE OF SEQUENCE {
     user        Name,
```

```
     reason      RevokeUserReason }
     -- If no user is revoked then an empty sequence is sent.
RevokeUserReason ::= ENUMERATED {
     unspecified (0),
     keyCompromise     (1),
     userBlackListed   (2),
     affiliationChanged (3) }

CKITlcmaAnnContent ::= SEQUENCE {
     localDomainInfo   [0]   LocalDomainInfo   OPTIONAL,
     externalDomainInfo      [1]   ExternalDomainInfo      OPTIONAL }


-- Peer CMA to CMA Exchange
-- ___________________________

CKIPeerCmaReqContent ::= PeerExchangeInfo
PeerExchangeInfo ::= SEQUENCE {
     domainCertificate       Certificate,
     myIKRef           IKReference,
     revokedUsers            RevokedUserList,
     externalCMACertificate  [0] Certificate   OPTIONAL
           -- present if directory or PKI not used
           -- to distribute the CMA's certificate
}

CKIPeerCmaRepContent ::= SEQUENCE {
     status            PKIStatusInfo,
     peerExchangeInfo        [0] PeerExchangeInfo OPTIONAL }


-- Obtain External Seed Key
-- ___________________________
CKIExtSeedKeyReqContent ::=  Certificate

CKIExtSeedKeyRepContent ::= SEQUENCE {
     status      PKIStatusInfo,
     extSeedKeyInfo    [0] ExtSeedKeyInfo      OPTIONAL,
     externalCMACertificate  [1] Certificate   OPTIONAL
           -- present if directory or PKI not used
           -- to distribute the CMA's certificate
}

ExtSeedKeyInfo ::= SEQUENCE {
     externalCMA CMAName,
     user        Name,
     seedKey     BIT STRING,
--  The seed key may be protected by a mechanism
--  outside the scope of this specification.
     seedKeyValidity   IKValidity }

CKIExtSeedKeyAnnContent ::= SEQUENCE {
     extSeedKeyInfo    [0] ExtSeedKeyInfo,
     externalCMACertificate  [1] Certificate   OPTIONAL
           -- present if directory or PKI not used
           -- to distribute the CMA's certificate
   }
```

```
-- Obtain Send Certificate
-- ________________________

CKISendCertReqContent ::= SEQUENCE {
     user              Name,
     suggestedValidityStart  GeneralizedTime,
     suggestedValidityEnd    GeneralizedTime   OPTIONAL }

CKISendCertRepContent ::= SEQUENCE {
     status      PKIStatusInfo,
     sendCertificate   SendCertificate OPTIONAL }

SendCertificate ::= Certificate

CKISendCertAnnContent ::=    SendCertificate


-- Obtain Receive Certificates
-- ___________________________

CKIRecCertReqContent ::= SEQUENCE {
     recipientUsers          SEQUENCE OF Name,
     suggestedValidityStart  GeneralizedTime,
     suggestedValidityEnd    GeneralizedTime   OPTIONAL }

CKIRecCertRepContent ::= SEQUENCE {
     status      PKIStatusInfo,
     receiveCerts      SEQUENCE OF Certificate OPTIONAL }

CKIRecCertAnnContent ::= SEQUENCE OF Certificate


-- CRL Distribution
-- ________________________

CRLAnnContent ::= SEQUENCE OF CertificateList


-- Revoked User List
-- _________________

CKIRevokedUserListAnnContent ::= SEQUENCE {
           domain      CMAName,
           list  RevokedUserList }


-- Key Recovery
-- ____________

KeyRecReqContent ::= InitReqContent

InitReqContent ::= SEQUENCE {
```

```
    protocolEncKey  [0] SubjectPublicKeyInfo  OPTIONAL,
                -- Not required for CKI key recovery
      fullCertTemplates   FullCertTemplates }

FullCertTemplates ::= SEQUENCE OF FullCertTemplate

FullCertTemplate ::= SEQUENCE {
     certReqId   INTEGER,
     -- to match this request with corresponding response
     -- (note:  must be unique over all FullCertReqs in this message)
     certTemplate           CertTemplate
 }
-- Other OPTIONAL fields in FullCertTemplate defined in
-- [PKI-3] are not required for the CKI.

CertTemplate ::= SEQUENCE {
       version    [0] Version               OPTIONAL,
       serial     [1] INTEGER               OPTIONAL,
       signingAlg [2] AlgorithmIdentifier   OPTIONAL,
       subject    [3] Name                  OPTIONAL,
           -- Required for CKI key recovery
       validity   [4] OptionalValidity      OPTIONAL,
           -- Required for CKI key recovery
       issuer     [5] Name                  OPTIONAL,
       publicKey  [6] SubjectPublicKeyInfo  OPTIONAL,
           -- Required for CKI key recovery
       issuerUID  [7] UniqueIdentifier      OPTIONAL,
       subjectUID [8] UniqueIdentifier      OPTIONAL,
       extensions [9] Extensions            OPTIONAL
           -- Key Usage extension required for CKI key recovery
}

  OptionalValidity ::= SEQUENCE {
       notBefore  [0] GeneralizedTime OPTIONAL,
           -- Required for CKI key recovery
       notAfter   [1] GeneralizedTime OPTIONAL
           -- Required for CKI key recovery
   }
-- Note: UTCTime is used in the current PKI protocol, but it is
-- expected that this will change in line with updates to X.509
KeyRecRepContent ::= SEQUENCE {
     status            PKIStatusInfo,
     newSigCert  [0]   Certificate OPTIONAL,
     caCerts     [1]   SEQUENCE OF Certificate OPTIONAL,
     keyPairHist [2]   SEQUENCE OF CertifiedKeyPair  OPTIONAL
                       -- Required for CKI key recovery
  }


CertifiedKeyPair ::= SEQUENCE {
       certificate     [0] Certificate         OPTIONAL,
                       -- Required if
                       -- SEQUENCE OF CertifiedKeyPair
                       -- contains >1 element
       privateKey      [2] EncryptedValue    OPTIONAL
                       -- Required for CKI key recovery
     -- Other optional fields in [PKI-3] are not required
 }
```

```
  EncryptedValue ::= SEQUENCE {
      encValue          BIT STRING,
      -- the encrypted value itself
      intendedAlg   [0] AlgorithmIdentifier  OPTIONAL,
      -- the intended algorithm for which the value will be used
      symmAlg       [1] AlgorithmIdentifier  OPTIONAL,
      -- the symmetric algorithm used to encrypt the value
      encSymmKey    [2] BIT STRING           OPTIONAL,
      -- the (encrypted) symmetric key used to encrypt the value
      keyAlg        [3] AlgorithmIdentifier  OPTIONAL
      -- algorithm used to encrypt the symmetric key
       }



-- Secure Bind
-- ___________


CKIBindReqContent ::= Credentials

-- The syntax for credentials is as defined in
--  ITU-T X.511 | ISO/IEC 9594-3 clause 8.1.1.

CKIBindRepContent ::= SEQUENCE {
     status      PKIStatusInfo,
     credentials Credentials }


END
```

Annex B Use of Internet PKI Protocol

I. Introduction

B.1 The CKI key management protocol operates as an extension to the Internet public key infrastructure certificate management protocols [PKI-3].

B.2 The basic PKI message structure is used for the CKI, with alternative content to support the CKI specific protocol exchanges.

B.3 Where an existing PKI protocol exchange meets the requirements of the CKI, it is adopted as part of CKI.

B.4 The CKI protocol exchanges can be used to manage confidentiality keys alongside the use of PKI protocol exchanges to manage certificates used for authentication. Alternatively, the CKI key management protocol can be used on its own.

II. PKI Management Model

B.5 The CKI architecture incorporates [CKI-4] the following entities as described in the PKI management model:

a. Subjects and end entities

b. Certification Authority

c. Registration Authority

B.6 Certificate Management Authority (CMA) is a special form of Certification Authority for the management of confidentiality keys.

III. PKI Management Requirements

B.7 A system conforming to the requirements of the PKI can also conform to the CKI requirements, with the following restrictions: (relevant clause number in [PKI-3] is given at the start of each item):

- (2.1.2.6) user keys for the CKI are generated by the CMA

B.8 The following PKI management requirements are also applicable to the CKI (the number of relevant clause in [PKI-3] is given at the start of each item):

- (2.1.2.1) PKI & CKI management must conform to ISO 9594-8 and the associated draft amendment (DAM)
- (2.1.2.9) PKI & CKI management protocols must be usable over a variety of "transport" mechanisms, specifically including, mail, HTTP, TCP/IP and FTP.
- (2.1.2.10) Final authority for certificate creation rests with the CA; no RA or end entity equipment should assume that any certificate issued by a CA will contain

what was requested -- a CA may alter certificate field values or may add, delete or alter extensions according to its operating policy; the only exception to this is the public key, which the CA may not modify (assuming that the CA was presented with the public key value). In other words, all PKI entities (end entities, RAs and CAs) must be capable of handling responses to requests for certificates in which the actual certificate issued is different from that requested -- for example, a CA may shorten the validity period requested.

- (2.1.2.11) A graceful, scheduled change-over from one non-compromised CA key pair to the next must be supported (CA key update).

Note: The last has been stated before for CASM but could be a useful requirement.

IV. Common Operations

B.9 The PKI management operation for CRL announcement is also used in support of the CKI.

B.10 The PKI management operation for key recovery is also used in the CKI to obtain private keys under a legal warrant.

B.11 Other operations are unique to CKI.

V. Data Structures

B.12 The overall PKI message and status information structures are used for the CKI as described in §II.

B.13 A new protocol version number is defined for the CKI or use of the CKI with the PKI.

B.14 New content type numbers and tags are used for the CKI specific content.

B.15 The PKI Protection field is not used to protect CKI protocol exchanges.

B.16 The PKI Status codes are used to indicate whether a response is successful or whether the values returned differ from those requested.

Annex C Illustration of CKI Key Management Protocol Use

I. Set up Domain Parameters Using TLCMA

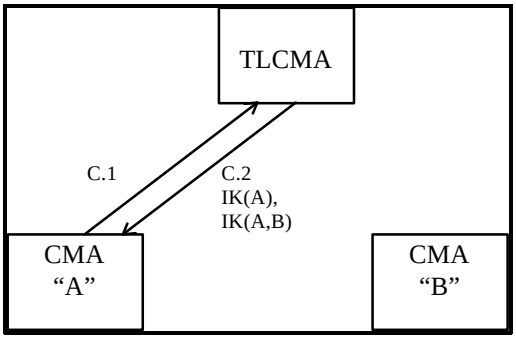

Figure 1 Set up domain parameters using TLCMA

C.1 CMA A starts up and sends a **`CKITlcmaReq`** to request its local interoperability key and external interoperability key for domain B from the TLCMA:

C.2 The TLCMA responds with a **`CKITlcmaRep`** containing the requested interoperability keys and associated information.

II. Set up Domain Parameters - Peer to Peer

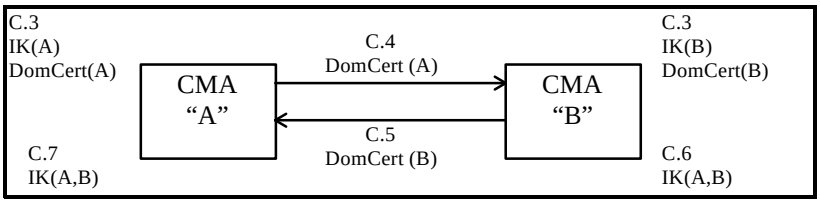

Figure 2 Set up domain parameters - peer to peer

C.3 On start up CMA A has pre-loaded its local interoperability key - IK(A) - domain certificate - DomCert(A). Similarly, CMA B its equivalent information pre-loaded - IK(B) DomCert(B).

C.4 CMA A sends CMA B a **`CKIPeerCmaReq`** containing A's domain certificate (and associated information).

C.5 On receipt of this message CMA B responds with a **`CKIPeerCmaReq`** containing its domain certificate (and associated information).

C.6 CMA B generates the interoperability key - IK(A,B) - using its domain private key and the public key received from A - DomCert(A).

C.7 CMA A generates the interoperability key - IK(A,B) - using its domain private key and the public key received from B - DomCert(B).

III. Obtain Send Certificate

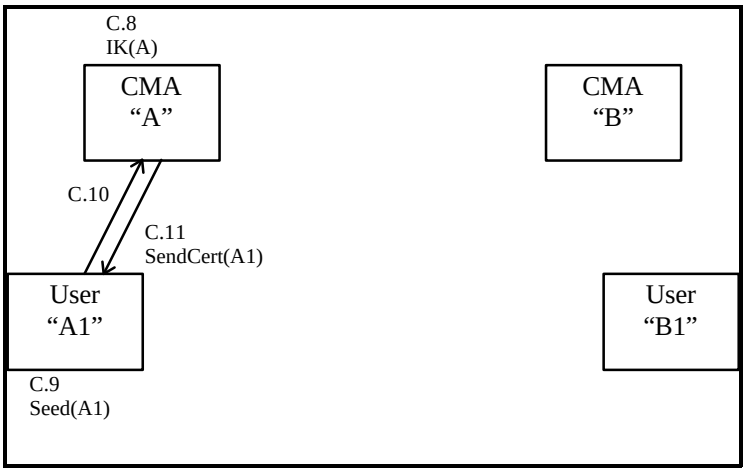

Figure 3 Obtain send certificate

C.8 CMA A has its local interoperability key - IK(A) - loaded using, for example, one of the two methods described above.

C.9 User A1 has its local seed key pre-loaded.

C.10 User A1 requests its sends a **`CKISendCertReq`** to request a send certificate.

C.11 CMA A generates the send certificate using IK(A) and replies with the send certificate - SendCert(A1) - in **`CKISendCertRep`**.

IV. Obtain Receive Certificate

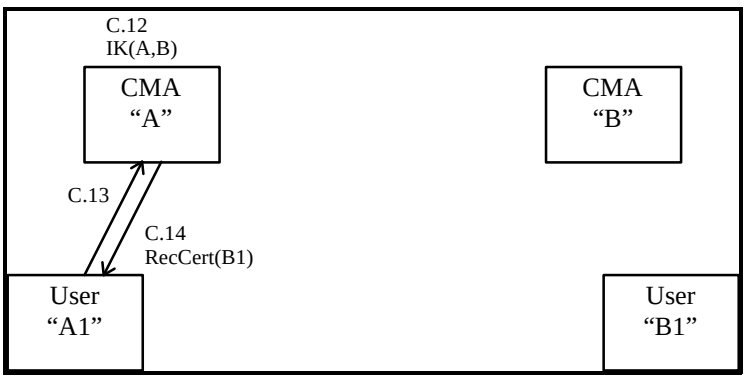

Figure 4 Obtain receive certificate.

C.12 CMA A has the interoperability key shared with B - IK(A,B) - loaded using one of the mechanisms described above.

C.13 User A1 sends to a **`CKIRecCertReq`** to requests the receive certificate for user B1.

C.14 CMA A uses the external interoperability key for the domain containing user B1 - IK(A,B) - to generate a receive certificate for user B1 which it sends to user A1 in a **`CKIRecCertRep.`**

V. Send Protected Message

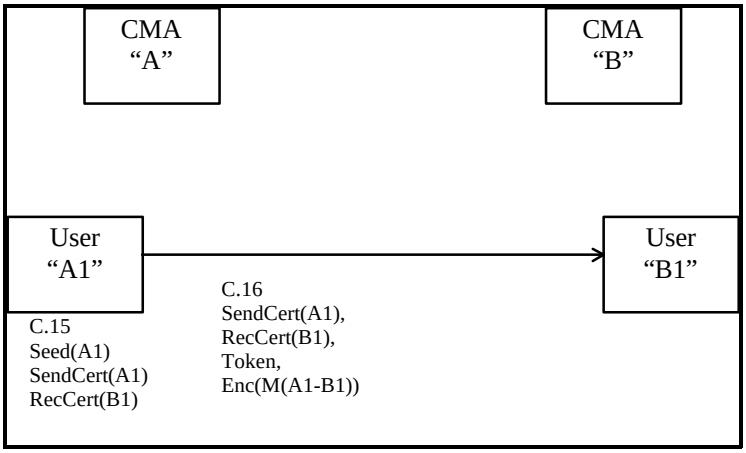

Figure 5 Send protected message

C.15 User A1 has loaded, as described above:

a. Seed(A1)

b. SendCert(A1)

c. RecCert(B1)

C.16 User A1 generates a token key using Seed(A1) and RecCert(B1) which it then uses in protect the data encryption key used to encrypt the message to B1 - Token, Enc(M(A1-B1)). User A1 then sends SendCert(A1), RecCert(B1) along with the token and protected message to user B1.

VI. Obtain External Seed Key

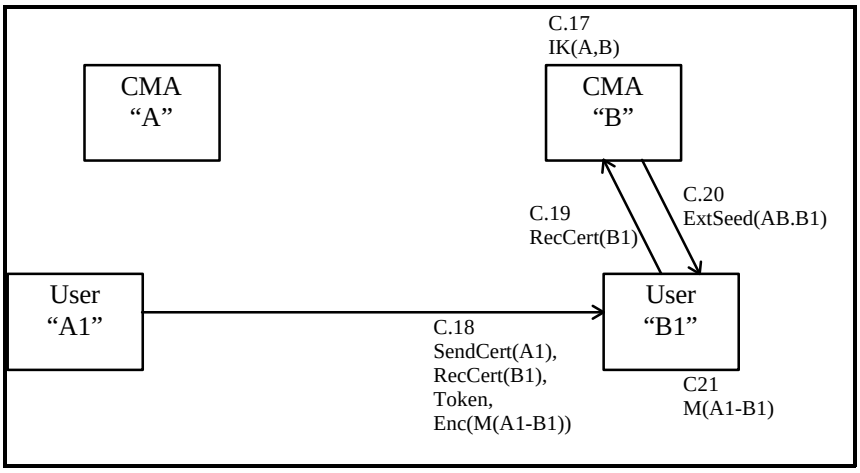

Figure 6 Obtain external seed key.

C.17 CMA B has loaded an interoperability key shared with A - IK(A,B) using, for example, one of the mechanisms described above.

C.18 User B1 receives SendCert(A1), RecCert(B1) and Enc(M(A1-B1)) from user A1.

C.19 User B1 requests the external seed key by sending RecCert(B1) to CMA B in a **`CKIExtSeeReq`**.

C.20 CMA B generates an external seed key for B1 using IK(A,B) and the parameters in RecCert(B1) - ExtSeed(AB,B1) - which it sends back to user B1.

C.21 User B1 generates the token key using ExtSeed(AB,B1) and SendCert(A1) which then it uses to obtain the data encryption key and so decrypt M(A1-B1).

VII. Revocation - User Compromised

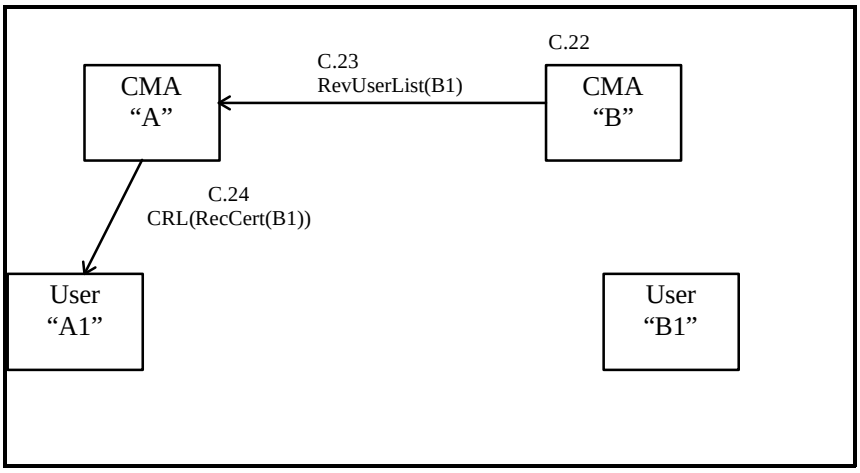

Figure 7 Revocation - user compromised.

C.22 CMA B finds out that user B1's seed key is compromised.

C.23 CMA B sends a **`CKIRevokedUserListAnn`** to CMA B announcing a new revoked user list including B1 - RevUserList(B1).

C.24 CMA A sends a CRLAnn message to all its users including any receive certificates produced for B1 - CRL(RecCert(B1)).

C.25 Any future request for a receive certificate for B1 is rejected until a revoked user list is received not containing B1.