NISTIR 90-4262

SECURE DATA NETWORK SYSTEM (SDNS) KEY MANAGEMENT DOCUMENTS

Charles Dinkel
Editor

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary

Lee Mercer, Deputy Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR 90-4262

SECURE DATA NETWORK SYSTEM (SDNS) KEY MANAGEMENT DOCUMENTS

Charles Dinkel
Editor

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Gaithersburg, MD 20899

February 1990

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
Lee Mercer, Deputy Under Secretary for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

Table of Contents

FOREWORD

The Secure Data Network System (SDNS) architecture and a set of associated specifications were developed through a multi-organizational project sponsored by the National Security Agency (NSA). They are presented here as a basis for standardization of security services in the Open Systems Interconnection (OSI) architecture. The National Institute of Standards and Technology (NIST) intends to encourage widespread adoption of the resulting standards and the implementation of these security services into a wide spectrum of vendor products.

NIST is publishing the specifications that resulted from Phase I of the SDNS project for review and comment from potential government and commercial users of security products. The specifications are not complete or totally consistent, either internally or with a number of other security projects in the National and International Standards arena. Readers of these documents should recognize that these specifications are subject to modification for various reasons as they progress through the standards process. The sponsor and participants in the SDNS project are acknowledged for the work accomplished and their support in developing and releasing these specifications.

The SDNS project was initiated by NSA to investigate methods of implementing security in a distributed computer network. The results of this project include a set of specifications that include security services, protocols and mechanisms for protecting user data in networks that are based on the OSI computer network model. Productive security services that protect user data are specified and supportive security services, such as key management and access control, are also provided. No cryptographic algorithms are included in these specifications.

NIST is working with NSA and industry to identify and develop a framework of base standards for network security. In 1989, NIST established the OSI Security Laboratory where interested researchers from government and industry develop and demonstrate new ideas in network security. The major goals of NIST's network security activities are to:

- Identify and develop security standards for open systems
- Specify a key management system that supports these security standards
- Encourage the development of interoperable equipment

The documents resulting from Phase I of the SDNS project are as follows:

SDN.301 - Security Protocol 3 (SP3)
SDN.401 - Security Protocol 4 (SP4)
SDN.601 - Key Management Profile - Communication Protocol Requirements for Support of the SDNS Key Management Protocol
SDN.701 - Message Security Protocol
SDN.702 - SDNS Directory Specifications for Utilization with the SDNS Message Security Protocol
SDN.801 - Access Control Concepts Document
SDN.802 - Access Control Specification
SDN.902 - Key Management Protocol - Definition of Services Provided by the Key Management Application Service Element
SDN.903 - Key Management Protocol - Specification of the Protocol for Services Provided by the Key Management Application Service Element
SDN.906 - Key Management Protocol - SDNS Traffic Key Attribute Negotiation

Because of the wide spread interest in the SDNS project, NIST is publishing these ten documents as three Reports entitled: **Security Protocols, Key Management,** and **Access Control.** The following diagram shows the relationship and contents of these reports.

NIST REPORT				
SECURITY PROTOCOLS	SDN.301 SECURITY PROTOCOL 3 (SP3)	SDN.401 SECURITY PROTOCOL 4 (SP4)	SDN.701 MESSAGE SECURITY PROTOCOL	SDN.702 DIRECTORY SPECS FOR USE WITH MSP
KEY MANAGEMENT	SDN.601 KEY MANAGEMENT PROFILE	SDN.902 KMP DEFINITION OF SERVICES PROVIDED BY KM ASE	SDN.903 KMP SERVICES PROVIDED BY KM ASE	SDN.906 KMP TRAFFIC KEY ATTRIBUTE NEGOTIATION
ACCESS CONTROL	SDN.801 ACCESS CONTROL CONCEPT DOCUMENT	SDN.802 ACCESS CONTROL SPECIFICATION		

INTRODUCTION

NISTIR 90-4262 includes four documents dealing with key management which were developed by the National Security Agency (NSA) as output from the Secure Data Network System (SDNS) project. **SDN.601, Communication Protocol Requirements for Support of the SDNS Key Management Protocol**, supplies a profile for the implementation of SDNS Key Management services in Open Systems. It is primarily concerned with providing guidance for implementation agreements on the variety of protocol stacks that are needed to satisfy the Key Management Application Process (KMAP) requirements in various communication specific environments. It also specifies the protocol support required at the transport, network, data link, and physical layers for varying communication environments; namely Packet Switched Networks (PSN), Local Area Networks (LAN), and direct dial Public Switched Telephone Networks (PSTN).

SDN.902, Definition of Services Provided by the Key Management Application Service Element (KMASE), defines in an abstract way the key management services within the OSI Application Layer in terms of:

a) the primitive actions and events of the service;

b) the parameter data associated with each primitive action and event; and

c) the relationship between and the valid sequences of these actions and events.

The standard only defines the services in terms of an abstract model. It implies neither a particular implementation of the services, nor does it imply a particular representation of the service primitives.

The protocol specified in **SDN.903, Specification of the Protocol for Services Provided by the Key Management Application Service Element**, describes the KMASE services provided to the Key Management Application Process (KMAP) to support applications in a distributed open systems environment. Key management provides for the generation, distribution, and updating of traffic encryption keys. Some management capabilities for authentication and access control are provide by the KMAP.

SDN.906, SDNS Traffic Key Attribute Negotiation, specifies the framework of the SDNS security attribute negotiation service. It supplements **SDN.902** and **SDN.903** where traffic key security service attributes negotiation is specified.

The four key management documents of **NISTIR 90-4262** support the security protocols addressed in **NISTIR 90-4250** and the access control documents covered in **NISTIR 90-4259.**

Comments and feedback are solicited by NIST.

SPECIFICATION SDN.601
Revision 1.5 1989-08-11

SDNS
Secure Data
Network System

Key Management Profile

Communication Protocol Requirements for Support of the SDNS Key Management Protocol

Source: The SDNS Protocol and Signaling Working Group

Introductory note:

This document provides the framework for the SDNS Key Management Profile. It is subject to change during the development phase of SDNS.

TABLE OF CONTENTS

0. Introduction

This document specifies a profile for the implementation of SDNS Key Management services in Open Systems. It is primarily concerned with providing guidance for implementation agreements on the variety of protocol stacks that are needed to satisfy the Key Management Application Process (KMAP) requirement in various communication specific environments.

1. Scope and field of Application

This profile specifies the protocol support services the KMAP requires from the Application Control Service Elements (ACSE), the ISO Presentation layer kernel, and the ISO Session layer. It also specifies the protocol support required at the transport, network, data link, and physical layers for varying communication environments; namely, Packet Switched Networks (PSN), Local Area Networks (LAN), and direct dial Public Switched Telephone Networks (PSTN).

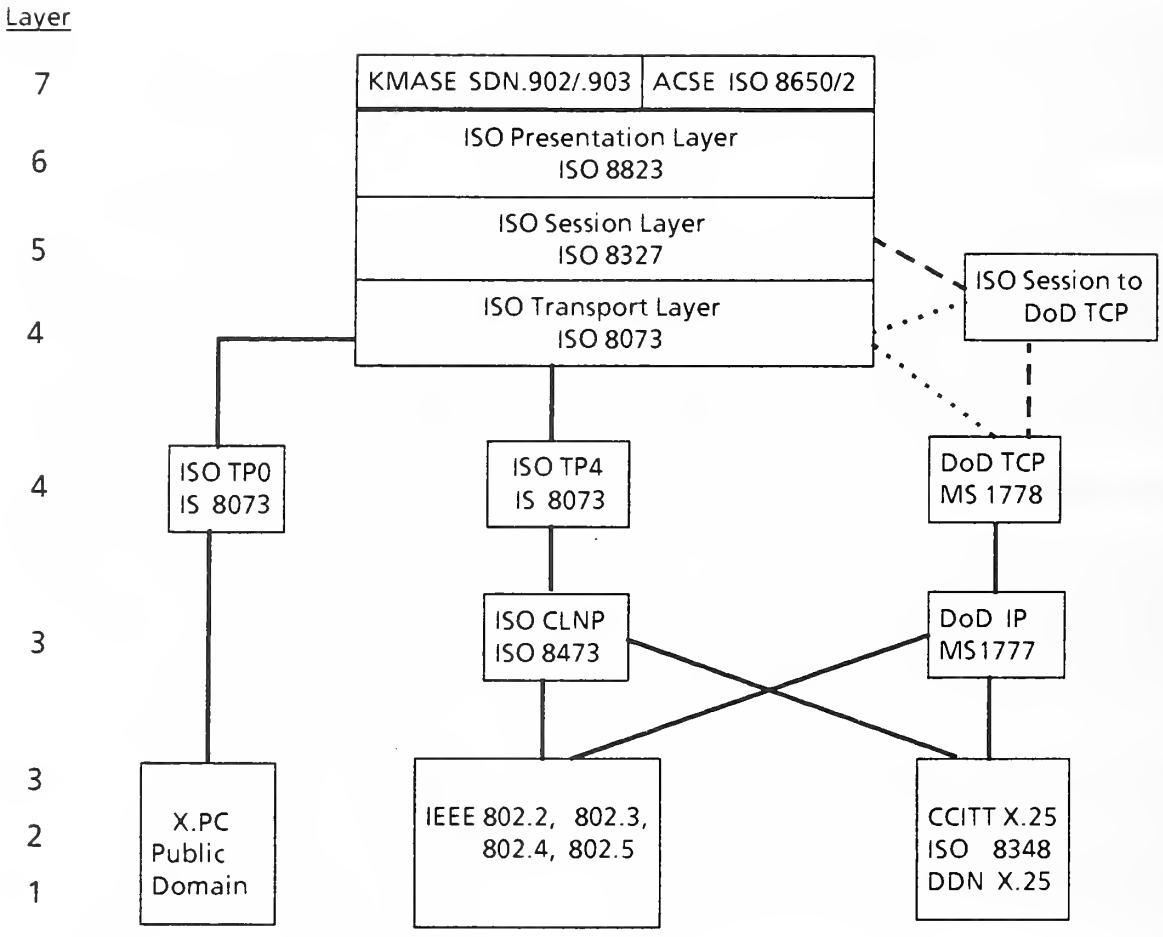
2. Use of ASEs and Lower Layer Services

The protocols referenced in the following paragraphs are required to provide the basic connection oriented message traffic, at ISO layers seven through five, for use in performing key management functions. At layer four and below, several different options are available to support a variety of envisioned SDNS connection and connectionless oriented environments.

The KMAP protocol stack alternatives are depicted in Figure 1. As illustrated in figure 1, the KMASE always requires the services of the ISO layer 7 ACSE, the ISO layer 6 Presentation kernel, and the ISO Session Protocol, at layer 5. The Presentation and Session layer protocols shall be implemented in accordance with Government Open System Interconnection Profile (GOSIP) standards, which supplement ISO standards for SDNS components developed for U.S. Government use.

For Department of Defense (DoD) Transmission Control Protocol (TCP) and Internet Protocol (IP) implementations, the ISO Session layer must be mapped to the DoD Transport layer. The specifics of this mapping are beyond the scope of this document.

The Transport layer will use ISO TP class 0 when direct dial PSTN connections are required. For SDNS key management support, TP(0) will only be used in conjunction with the "public domain" X.PC specification. The Transport layer requires ISO TP class 4 when the ISO Connectionless Network Protocol (CLNP) is employed for SDNS key management support. The DoD's TCP must be used for the transport layer when the DoD Internet Protocol (IP) is employed at the connectionless network layer. The use of DoD's TCP is dependent upon a standard mapping convention from the ISO Session layer, as stated above.



Notes:

ISO Session Layer to DoD Transport Layer (TCP) "protocol bridge" is not a standard.

ISO Presentation, Session, and Transport Layers may be supplemented by GOSIP for U.S. Government components.

ISO Transport Layer classes 1, 2, and 3 are not used for SDNS Key Management.

Figure 1. THE KMAP PROTOCOL STACK ALTERNATIVES

At the Network layer, the ISO CLNP standard is employed when ISO TP(4) is utilized at the Transport layer. CLNP may interface to X.25 layer 3, or to the Data Link layer for broadcast LANs (IEEE 802 standards). The DoD IP connectionless network protocol is used when DoD TCP is utilized at the Transport layer. IP may interconnect to an underlying X.25 layer 3, or to the data link layer for LANs using IEEE 802 standards.

At the Data Link and Physical layers, CCITT's X.25 and/or DDN X.25, or IEEE 802 standards may be employed. For direct dial, the X.PC "public domain" protocol shall be used. X.PC is specified in publication number NPD-269, available, without charge, from McDonnell Douglas.

> (Note that it is also viable for ISO TP(4) to use ISO CONS without a connectionless network layer. It is also viable, in some parts of the world, to have X.25 virtual circuit LANs. Both of these are prominent implementations and terminology used in Europe, primarily as a result of PTT controls. If or when SDNS devices are available for export to European countries, this is an important consideration for SDNS implementors.)

2.1. Application-Control-Service-Elements (ACSE)

ACSE is specified in ISO 8650/2. All Specific Application Service Elements (SASE) - i.e., what the KMASE is in the ISO application layer, use ACSE to establish, release, and abort connections with peer SASE. All ACSE Protocol Data Unit (APDU) types and service primitives specified in the ISO standard are required for all SDNS key management applications.

Cooperating SDNS implementations shall support all mandatory parameters specified in the ACSE protocol standard. Unsupported parameter options shall not result in a protocol error.

2.2. Presentation-Protocol (PP)

Connection oriented Presentation protocols (PP) exist to ensure that the information content of APDUs and KPDUs are preserved during their transit between application entities. Functions performed by PP include negotiation of transfer syntaxes and transformation to/from transfer syntax. Services include connection establishment and release, context management, and information transfer. Cooperating SDNS devices shall use a single transfer syntax for KPDUs.

2.3. Session-Protocol (SP)

Connection oriented Session Protocol provides the means for the organized exchange of data between cooperating application entities. Services provided by the protocol include the establishment and release of connections and the exchange of data in an orderly manner.

Cooperating SDNS implementations shall support all SPDU types and primitives used by the session kernel, duplex, and expedited data functional units. All mandatory parameters shall be supported for each of the functional units. Details of the functional units and SPDU parameters are defined in ISO 8327. Unsupported parameter options shall not result in a protocol error or PDU discard.

Where DoD TCP is the underlying Transport layer protocol, there shall exist a mapping of the ISO Session Protocol and the DoD TCP. Such mapping shall be defined in an appropriate DoD standard. Until such a standard is available, implementors are discouraged from implementing this form of protocol stack.

2.4. Transport Layer Protocols

2.4.1 Transport-Protocol (TP) Class 4

Connection oriented Transport Protocol Class 4 provides reliable transparent transfer of data between cooperating KMAPs which operate over the connectionless-mode network service. Services provided by TP4 include the means to:

- establish transport connections and negotiate Quality of Service (QOS) parameters;
- transfer Transfer Service Data Units (TSDU) with or without concatenation and separation, segmenting and separating, and splitting and recombining;
- control TSDU rate (flow control);
- transfer expedited TSDUs subject to separate flow control; and,
- release the transport connection.

TP(4) shall be implemented when the underlying network protocol is the ISO Connectionless Network Protocol (CLNP). Cooperating TP4 implementations shall support all Transport PDU (TPDU) types and primitives used by the Session service to transfer KMP data between cooperating SDNS devices. All mandatory parameters for TPDUs shall be implemented, as defined in ISO 8073 and DIS 8073/AD2. Unsupported parameter options shall not result in a protocol error or PDU discard.

2.4.2. Transport-Protocol (TP) Class 0

Connection oriented Transport Protocol Class 0 provides transparent transfer of data between cooperating KMAPs operating over ISO connection oriented network service protocols. Services provided by TP0 include the means to:

- establish transport connection;
- transfer TSDUs; and,
- release transport connections.

TP(0) is required to support all SDNS key management exchanges in the "dial-up" access mode. This access mode requires the underlying services of the X.PC "public domain" protocol. Details of TP0 TPDUs and Session Protocol primitives are defined in ISO 8073. Cooperating TP0 implementations shall support all TPDUs and Session Protocol primitives, and all mandatory parameters. Unsupported optional parameters shall not result in a protocol error or PDU discard.

2.4.3. DoD Transmission-Control-Protocol (TCP)

TCP is required to support SDNS key management exchanges that are compelled to operate within internets that already exist with a significant number of IP routers. This kind of implementation is not encouraged, but may be necessary in some cases to keep the cost of deploying SDNS components to a minimum; for example, where IP routers are the only available routers on networks where SDNS devices are to be

deployed. Consistent implementations of this type may not be feasible until ISO Session layer to DoD transport layer protocol mappings are standardized.

Details of TCP TPDUs and primitives are defined in Military Standard (MILSTD) 1778. Cooperating TCP implementations shall support all TPDUs and Session Protocol (mapped) primitives (To Be Determined) and all mandatory parameters. Unsupported optional parameters shall not result in a protocol error or PDU discard.

2.5. Network Layer Protocols

2.5.1. ISO Connectionless-mode-Network-Protocol (CLNP)

The ISO CLNP shall be used when the ISO TP Class 4 transport protocol is employed. Details of CLNP NPDUs and primitives are defined in ISO 8473. CLNP NPDU primitives and all mandatory parameters shall be implemented in accordance with ISO 8473. Unsupported optional parameters shall not result in a protocol error or PDU discard.

2.5.2. DoD Connectionless-mode Internet Protocol (IP)

The DoD IP shall be used when the DoD TCP transport protocol is employed. Details of IP NPDUs and primitives are defined in MILSTD 1777. IP, including the Internet Control Message Protocol (ICMP), primitives and all mandatory parameters shall be implemented In accordance with MILSTD 1777. Unsupported optional parameters shall not result in a protocol error or PDU discard.

2.5.3. CCITT X.25 Packet Layer Protocol (PLP)

The X.25, layer 3, shall be used for SDNS key management exchanges that use either TCP/IP or TP(4)/CNLP for X.25 network connectivity. Details of X.5 PLP NPDUs and primitives are defined in ISO 8348. X.25 PLP NPDU primitives and all mandatory parameters shall be implemented in accordance with ISO 8348. Unsupported optional parameters shall not result in protocol error or PDU discard.

> (Note: Where DDN and/or BLACKER X.25 layer 3 implementation are required, variances from ISO 8348 may be required.)

2.6. Data Link and Physical Layer Protocols

2.6.1. Direct Dial Public Switched Telephone Network Data Link and Physical Layers

The X.PC "public domain" protocol shall be used for all key management exchanges that are required over direct dial PSTNs. Details of the X.PC protocol are available from McDonnell Douglas, under the publication title NPD-269. For SDNS key management exchange support, the X.PC protocol shall operate in the packet mode only; i.e., not the character mode.

> (Note: X.PC also has some network layer functionality, (i.e., addressing, packet assembly/disassembly, and multiple virtual circuit support - even through the underlying connection is strictly circuit switched [point-to-point].))

2.6.2. IEEE 802.X Data Link and Physical Layer Protocols

IEEE 802 protocol standards shall be employed for key management exchanges that are required over LANs. Details of 802.2, 802.3, 802.4, and 802.5 are available from the IEEE. It is envisioned that there may be some diversion from the specifics of IEEE 802.2 and 802.3 standards for various Ethernet (a trademark of Xerox Corporation) implementations.

2.6.3. CCITT X.25 Data Link and Physical Layer Protocols

Implementors that employ X.25 protocols shall do so in accordance with CCITT X.25 protocol specifications at layers 2 and 1.

SDNS

Secure Data Network System

Key Management Protocol

Definition of Services provided by the Key Management Application Service Element

Source: SDNS Protocol and Signaling Working Group
Key Management Sub-Group

Introductory note :

This document provides the framework for the SDNS Key Management Protocol Specification. It is subject to change after the development phase of SDNS.

Table of Contents

0. Introduction

This section defines the key management services as provided by the Key Management Application Service Element (KMASE). The services defined are of the category Specific Application Service Element (SASE). It provides facilities to support the key management activities required by and in support of cooperating Key Management Application Processes (KMAP) including the establishment of traffic encryption keys (TEK), the updating of TEKs, the distribution of Certificate Revocation Lists (CRL), and the interactive and staged rekey with the Key Management System (KMS).

1. Scope and field of application

This specification defines in an abstract way the key management services within the OSI Application Layer in terms of:

a) the primitive actions and events of the service;

b) the parameter data associated with each primitive action and event; and

c) the relationship between and the valid sequences of these actions and events.

This standard only defines the services in terms of an abstract model. It implies neither a particular implementation of the services, nor does it imply a particular representation of the service primitives. As this is a service definition, there are no conformance testing requirements.

2. References

ISO IS 7498/1 Information Processing Systems - Open Systems Interconnection - Basic Reference Model.

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

ISO DIS 8649 Information Processing Systems- Open Systems Interconnection- Association Control Service Element Service Definition.

ISO TR8509 Information Processing Systems - Open Systems Interconnection - Service Conventions.

SDN.801 SDNS Access Control Concept Document.

SDN.802 SDNS Access Control Specification.

SDN.903 SDNS Key Management Protocol: Specification of the protocol for services provided by the Key Management Application Service Element.

SDN.906 SDNS Traffic Key Attribute Negotiation.

3. Definitions

This document uses definitions contained in the Reference Model for Open Systems Interconnection (ISO IS 7498/1) including the Security Architecture (ISO IS DIS 7498/2).

In addition the following definitions are used:

Credentials -- information exchanged between SDNS components which authenticates each component and used to form a TEK between the components.

Certificate Revocation List -- the list of key material identifiers which have been invalidated.

key material -- information supplied by the KMS that contains a component's credentials and private information.

Initiator -- the KMAP which initiates the establishment of an application association.

Rekey -- the process by which new key material is obtained from the KMS either directly or via a rekey agent.

Rekey Agent -- a SDNS component authorized to act as an intermediary between the KMS and other SDNS components for the delivery of key material.

Responder -- the KMAP which responds to a request for establishment of the application association.

Signed -- a method used to provide authentication of the originator of a message.

Universal-ID -- identifies the universal to which a credential belongs.

4. Abbreviations

ACSE	Association Control Service Element
ASE	Application Service Element
CRL	Certificate Revocation List
K-MIB	Key Management Information Base
ICV	Integrity Check Value
KID	Key Identifier
KMS	Key Management System
KMAE	Key Management Application Entity
KMAP	Key Management Application Process
KMASE	Key Management Application Service Element
PDU	Protocol Data Unit
SDNS	Secure Data Network System
TEK	Traffic Encryption Key
TEK-MIB	Traffic Encryption Key Management Information Base

5. Conventions

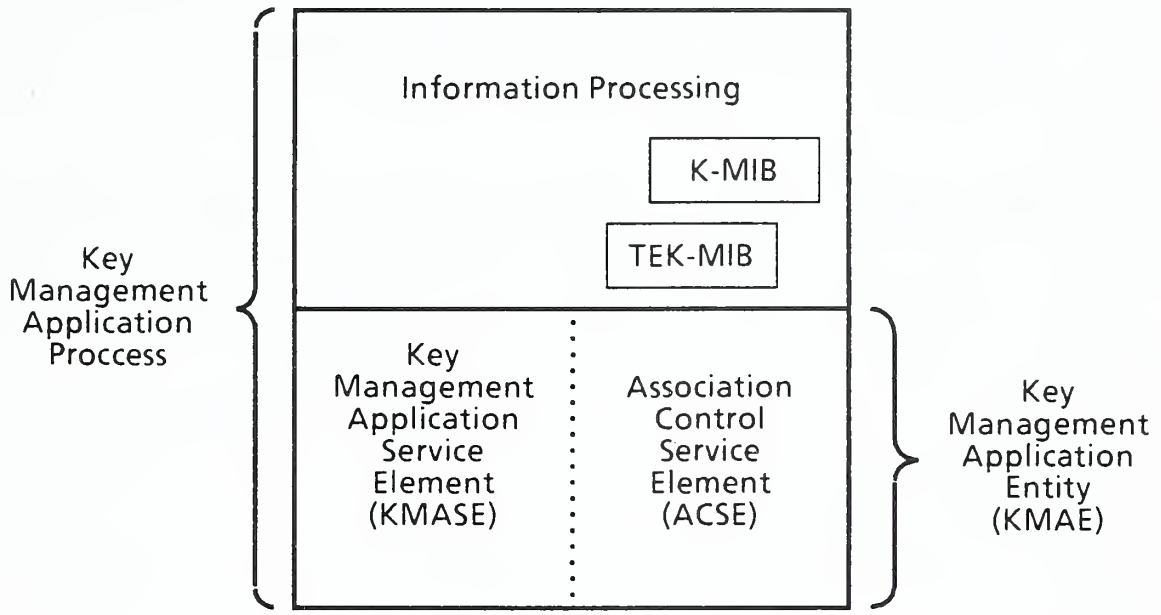
This part of the KMP specification uses the descriptive conventions in the OSI Service Conventions in ISO/TR 8509. In Section 9, the definition of each KMASE service includes a table that lists the parameters of its primitives. For a given primitive, the presence of each parameter is described by one of the following values :

blank	not applicable
M	presence mandatory
U	presence is a user option

In addition, the notation (=) indicates that a parameter value is semantically equal to the value to its left in the table.

6. Model and Basic Concept

This service definition uses the abstract model for a service defined in the OSI Service Conventions in ISO TR 8509. The model defines the interactions between two cooperating KMAPs. The KMAP consists of two parts: the information processing part, and the communication part, which is represented by the KMAE. When communication is required between two KMAPs, an application association is established between invocations of the processes' KMAEs. The KMAE represents an addressable set of communication capabilities which are defined by the KMASE and the ACSE.



KMP Model

The KMASE is driven by service request and response primitives from the KMAP and by indication primitives from the presentation service provider. The KMASE, in

turn, issues indication and confirm primitives to the KMAP, and issues request primitives to the ACSE and the presentation service provider.

The KMAP administers the Management Information Bases (K-MIB and the TEK-MIB) in the sense of updating data in the MIBs upon receipt of indications and confirms, and retrieving parameters from the MIBs when issuing requests and responses. The K-MIB is a security MIB used to store key material. The TEK-MIB is a security MIB used to store the TEK, KID, and attributes regarding the use of the TEK.

These services require information exchange with other SDNS KMAPs and with the KMS KMAP. The following services are provided:

a) Exchange TEK and security service information with other SDNS components.

b) Update TEKs.

c) Perform rekey operation with KMS or with a rekey agent. Seed key conversion is a special case of the rekey operation.

d) Obtain the Certificate Revocation List (CRL).

e) Report cryptographic key usage errors and routing information.

7. Overview of Service

This standard defines the following services for key management:

a) K-EXCH-CRED

b) K-TRY-ME

c) K-NO-KEY

d) K-TEK-ATTRIBUTES

e) K-UPDATE-TEK

f) K-IREKEY

g) K-REQUEST-SREKEY

h) K-DELIVER-SREKEY

i) K-GET-CRL

The K-EXCH-CRED service enables a KMAP to exchange credentials with a cooperating KMAP for the purpose of mutual authentication and subsequent formation of a TEK.

The K-TRY-ME service enables a KMAP to inform a cooperating KMAP that it is authorized to provide security services for a specific network service access point.

The K-NO-KEY service enables a KMAP to inform a cooperating KMAP that traffic has been received protected under a key for which the receiver has no valid KID.

The K-TEK-ATTRIBUTES service verifies the cryptographic compatibility of a TEK, negotiates the communication and security attributes associated with the TEK, and exchanges CRL numbers. Additionally, K-TEK-ATTRIBUTES allows a KMAP to associate the attributes of a previous TEK with a new TEK.

The K-UPDATE-TEK service enables a KMAP to update a TEK without exchanging credentials.

The K-IREKEY service enables a KMAP or Rekey Agent KMAP to interactively obtain new key material from the KMS. Multiple sets of key material may be obtained by sequential use of this service.

The K-REQUEST-SREKEY service enables a KMAP to request the staged rekey of key material through a Rekey Agent KMAP. The credentials which are sent to the Rekey Agent with this service are later forwarded to the KMS with the K-IREKEY service.

The K-DELIVER-SREKEY service enables a Rekey Agent to deliver key material to a component which has previously requested a staged rekey by the use of the K-REQUEST-SREKEY service.

The K-GET-CRL service enables a KMAP to obtain a CRL from a cooperating KMAP (e.g. the KMS KMAP). The new CRL is delivered only if the requestor has an older version of the CRL than the responder.

8. Relationship to Other ASEs and Lower Layer Services

The KMASE is a specific application service element. As specified here it does not describe a complete communications service, but instead relies on the Association Control Service Element (ACSE). The KMASE requires an existing application-association controlled by ACSE. The KMASE also assumes the use of the P-DATA service of the Presentation Service for transferring SDNS key management messages between peer SDNS entities.

9 Service Definition

This section contains the definitions of the primitives a KMAP uses to obtain KMASE services.

Table 1 -- KMASE Services

Service	Type
K-EXCH-CRED	Confirmed
K-TRY-ME	Unconfirmed
K-NO-KEY	Unconfirmed
K-TEK-ATTRIBUTES	Confirmed
K-UPDATE-TEK	Confirmed
K-IREKEY	Confirmed
K-REQUEST-SREKEY	Confirmed
K-DELIVER-SREKEY	Confirmed
K-GET-CRL	Confirmed

9.1. K-EXCH-CRED Service

The KMAP uses the K-EXCH-CRED service to request an exchange of credentials with a cooperating KMAP. It is a confirmed service.

9.1.1. K-EXCH-CRED Parameters

Table 2 lists the K-EXCH-CRED parameters.

Table 2 -- K-EXCH-CRED parameters

Parameter name	req	ind	rsp	cnf
univ-id	M	M(=)		
init-cred	M	M(=)		
init-kid	M	M(=)	M(=)	M(=)
resp-cred			M	M(=)
resp-kid			M	M(=)
result			M	M(=)

9.1.1.1. univ-id

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the universal-ID which applies to the initiator's key material.

9.1.1.2. init-cred

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the initiator's credentials which must be exchanged with the peer SDNS entity to form a TEK.

9.1.1.3. init-kid

This parameter is required on the request and response primitives. It is passed on the indication and confirm primitives. It contains the identifier the initiator KMAP will associate with the TEK.

9.1.1.4. resp-cred

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the responder's credentials which must be exchanged with the cooperating KMAP to form a TEK.

9.1.1.5. resp-kid

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the identifier the responder KMAP associates with the TEK.

9.1.1.6. result

This parameter is required on the response primitive. It is passed on the confirm primitive. It indicates the success of the operation.

9.2. K-TRY-ME Service

The KMAP uses the K-TRY-ME service to notify a cooperating KMAP that a specific end-system does not have a SDNS component, but can be reached through this (the K-TRY-ME initiator) SDNS component. It is an unconfirmed service.

9.2.1. K-TRY-ME Parameters

Table 3 lists the K-TRY-ME parameters.

Table 3 -- K-TRY-ME parameters

Parameter name	req	ind
end-sys-addr	M	M(=)

9.2.1.1. end-sys-addr

This parameter must be included on the request primitive and is passed on the indication primitive. It contains the address of the end system which has no SDNS component and which uses this SDNS component as a gateway.

9.3. K-NO-KEY Service

The KMAP uses the K-NO-KEY service to inform a cooperating KMAP that traffic has been received protected under a key for which the receiver has no valid key id. It is an unconfirmed service.

9.3.1. K-NO-KEY Parameters

Table 4 lists the K-NO-KEY parameters.

Table 4 --K-NO-KEY parameters

Parameter name	req	ind
rcvd-kid	M	M(=)
pdu-count	U	U(=)

9.3.1.1. rcvd-kid

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the KID the initiator KMAP has determined to be invalid.

9.3.1.2. pdu-count

This parameter may be included on the request primitive. It is passed on the indication primitive. It contains the number of PDUs the initiator KMAP has

received protected under the invalid key identifier and the period of time and time units during which reception has taken place.

9.4. K-TEK-ATTRIBUTES Service

Cooperating KMAPs use the K-TEK-ATTRIBUTES service to negotiate the communication and security attributes to be assigned a TEK. Cooperating KMAPs may also use this service to assign the attributes of an existing TEK to a new TEK. It is a confirmed service.

9.4.1. K-TEK-ATTRIBUTES Parameters

Table 5 lists the K-TEK-ATTRIBUTES parameters.

Table 5 -- K-TEK-ATTRIBUTES parameters

Parameter name	req	ind	rsp	cnf
init-kid			M	M(=)
resp-kid	M	M(=)		
proposed-options	U	U(=)		
selected-options			M	M(=)
add-info	U	U(=)	U	U(=)
old-resp-kid	U	U(=)		
crl-ver	M	M(=)	M	M(=)
result			M	M(=)

9.4.1.1. init-kid

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the identifier the initiator KMAP associates with the TEK.

9.4.1.2. resp-kid

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the identifier the responder KMAP associates with the TEK.

9.4.1.3. Options

9.4.1.3.1. proposed options

This parameter may be included on the request primitive. It is passed on the indication primitive. It contains the sets of options the requestor will support.

9.4.1.3.2. selected options

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the set of options the responder has selected.

The options which are negotiated by the exchange of the Ssrq and Ssrs KPDUs establish some of the characteristics of the communication between peer entities which share a traffic key. The negotiated options are represented as sets of parameters within the Ssrq/Ssrs KPDUs. Some of the options are dependent on other options.

Note - see SDN.906, SDNS Traffic Key Attribute Negotiation, for further information.

9.4.1.4. add-info

This parameter may be included on the request and response primitives. It is passed on the indication and confirm primitives. It contains additional information to be included as attributes of the key. The additional information may include:

- o auxilary vector information
- o signed lists of networks the SDNS component serves
- o proposed encryptor/decryptor addresses

Note - see SDN.906, SDNS Traffic Key Attribute Negotiation, for further information.

9.4.1.5. old-resp-kid

This parameter may be included on the request primitive. It is passed on the indication primitive. It contains an indicator that the KMAP wishes to associate the attributes of another TEK with the key specified by the init-KID and resp-KID parameters. Only the initiator of the cryptographic association specifies this parameter. If specified, the initiator omits the proposed options.

9.4.1.6. crl-ver

This parameter must be included on the request and response primitives. It is passed on the indication and confirm primitives. This parameter contains the unsigned version number of the current CRL used by the KMAP.

9.4.1.7. result

This parameter is required on the response primitive. It is passed on the confirm primitive. It indicates the success or failure of the operation.

9.5. K-UPDATE-TEK Service

The KMAP uses the K-UPDATE-TEK service to request a traffic key update for a TEK held by the cooperating KMAPs. This operation replaces an existing traffic key with a new traffic key. The traffic key, upon which the update operation was performed, is deleted. The communication and security attributes associated with the new key are the same as those of the previously existing key. It is a confirmed service.

9.5.1. K-UPDATE-TEK Parameters

Table 6 lists the K-UPDATE-TEK parameters.

Table 6 -- K-UPDATE-TEK parameters

Parameter name	req	ind	rsp	cnf
resp-kid	M	M(=)		
new-init-kid	M	M(=)	M(=)	M(=)
new-resp-kid			M	M(=)
result			M	M(=)

9.5.1.1. resp-kid

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the identifier the responder KMAP associates with the existing TEK.

9.5.1.2. new-init-kid

This parameter is required on the request and response primitives. It is passed on the indication and confirm primitives. It contains the identifier the initiator KMAP associates with the new TEK.

9.5.1.3. new-resp-kid

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the identifier the responder KMAP associates with the new TEK.

9.5.1.4. result

This parameter is required on the response primitive. It is passed on the confirm primitive. It indicates the success of the operation.

9.6. K-IREKEY Service (Interactive Rekey)

The KMAP uses the K-IREKEY service to request directly and interactively updated key material with the KMS. It may be used to rekey a single set of key material or sequentially to rekey multiple sets of key material. It is a confirmed service.

9.6.1. K-IREKEY Parameters

Table 7 lists the K-IREKEY parameters.

9.6.1.1. resp-kid

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the identifier the KMS associates with the TEK used for this exchange.

9.6.1.2. univ-id

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the universal-ID associated with the key material and CRL version number.

9.6.1.3. rekey-cred

This parameter may be included on the request primitive. It is passed on the indication primitive. The KMS performs the rekey operation based on these credentials when present.

9.6.1.4. crl-ver

This parameter must be included on the request and response primitives. It is passed on the indication and confirm primitives. On the request and indication primitives, it contains the version number of the CRL held by the KMAP to be rekeyed. On the response and confirm primitives, it contains the version number of the CRL held by the KMS.

Table 7 -- K-IREKEY parameters

Parameter name	req	ind	rsp	cnf
resp-kid	M	M(=)		
univ-id	M	M(=)		
rekey-cred	U	U(=)		
crl-ver	M	M(=)	M	M(=)
ref-num	U	U(=)	U(=)	U(=)
ekr	M	M(=)	M	M(=)
init-kid			M	M(=)
kms-cred-A			M	M(=)
kms-cred-B			M	M(=)
new-k-mat-A			M	M(=)
new-k-mat-B			M	M(=)
new-crl			U	U(=)
new-univ-crl			U	U(=)
no-more			U	U(=)
result			M	M(=)

9.6.1.5. ref-num

This parameter may be included on the request and response primitives. It is passed on the indication and confirm primitives. It is used by the KMAP to associate received key material (new-k-material) with the corresponding outstanding rekey requests.

9.6.1.6. ekr

This parameter must be included on the request and response primitives. It is passed on the indication and confirm primitives. It is used by the initiator KMAP to indicate expectance of an electronic key replacement and by the KMS KMAP to indicate success of the electronic key replacement procedure.

9.6.1.7. init-kid

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the identifier the initiator KMAP associates with the TEK used for this exchange.

9.6.1.8. kms-cred-A and kms-cred-B

These parameters must be included on the response primitive. They are passed on the confirm primitive. They contain the credentials of the KMS.

9.6.1.9. new-k-mat-A and new-k-mat-B

These paramenters are required on the response primitive. They are passed on the confirm primitive. They contain the updated key material.

9.6.1.10. new-crl

This parameter may be included on the response primitive. It is passed on the confirm primitive. It contains the CRL held by the KMS.

9.6.1.11. new-univ-crl

This parameter may be included on the response primitive. It is passed on the confirm primitive. It contains the CRL (held by the KMS) of the new universal.

9.6.1.12. no-more

This parameter may be included on the response primitive. It is passed on the confirm primitive. It is a flag indicating that the KMS will not accept any additional rekey requests at this time.

9.6.1.13. result

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the success or failure of the operation.

9.7. K-REQUEST-SREKEY Service (Staged Rekey)

The initiator KMAP uses the K-REQUEST-SREKEY service to request updated key material from the KMS via a rekey agent. This is useful when direct access to the KMS is unavailable. It may be used to rekey a single set of key material or sequentially to rekey multiple sets of key material. It is a confirmed service.

9.7.1. K-REQUEST-SREKEY Parameters

Table 8 lists the K-REQUEST-SREKEY parameters.

Table 8 -- K-REQUEST-SREKEY parameters

Parameter name	req	ind	rsp	cnf
resp-kid	M	M(=)		
univ-id	M	M(=)		
rekey-cred	M	M(=)		
ref-num			M	M(=)
ekr	M	M(=)		
ckl-ver	M	M(=)		
init-kid			M	M(=)
result			M	M(=)

9.7.1.1. resp-kid

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the identifier the agent associates with the TEK.

9.7.1.2. univ-id

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the universal-ID associated with the key material and CRL version number.

9.7.1.3. rekey-cred

This parameter is required on the request primitive and is passed on the indication primitive. It contains the credentials the KMS uses in the rekey operation.

9.7.1.4. crl-ver

This parameter must be included on the request primitive. It is passed on the indication primitive. It contains the version number of the CRL held by the KMAP to be rekeyed.

9.7.1.5. ref-num

This parameter is required on the response primitive. It is passed on the confirm primitive. The requesting KMAP uses this parameter to associate key material received via the K-DELIVER-REKEY service with the corresponding outstanding

rekey requests. It contains the reference identifier the Rekey Agent KMAP associates with the key material.

9.7.1.6. ekr

This parameter must be included on the request primitive. It is passed on the indication primitive. It indicates whether the initiator KMAP is expecting an electronic key replacement.

9.7.1.7. init-kid

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the identifier the initiator KMAP associates with the TEK used for this exchange.

9.8. K-DELIVER-SREKEY Service

An initiator KMAP uses the K-DELIVER-REKEY service to request delivery of new key material from a Rekey Agent KMAP. It may be used to deliver a single set of key material or sequentially to deliver multiple sets of key material. It is a confirmed service.

9.8.1. K-DELIVER-SREKEY Parameters

Table 9 lists the K-DELIVER-SREKEY parameters.

9.8.1.1. resp-kid

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the identifier the Rekey Agent (the responder) associates with the TEK used for this exchange.

9.8.1.2. ref-num

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the reference identifier the Rekey Agent KMAP associates with the key material.

Table 9 -- K-DELIVER-SREKEY parameters

Parameter name	req	ind	rsp	cnf
resp-kid	M	M(=)		
ref-num	M	M(=)		
ekr			M	M(=)
init-kid			M	M(=)
kms-cred-A			M	M(=)
kms-cred-B			M	M(=)
new-k-mat-A			M	M(=)
new-k-mat-B			M	M(=)
crl-ver			M	M(=)
new-crl			U	U(=)
new-univ-crl			U	U(=)
result			M	M(=)

9.8.1.3. ekr

This parameter must be included on the response primitive. It is passed on the confirm primitive. It indicates whether the initiator KMAP is receiving an electronic key replacement.

9.8.1.4. kms-cred-a and kms-cred-b

These parameters are required on the response primitive and are passed on the confirm primitive. They contain the credentials of the KMS.

9.8.1.5. new-k-mat-a and new-k-mat-b

These parameters are required on the response primitive. They are passed on the confirm primitive. They contain the new key material.

9.8.1.6. crl-ver

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the version number of the CRL held by the KMS.

9.8.1.7. new-crl

This parameter may be included on the response primitive. It is passed on the confirm primitive. It contains the CRL held by the KMS.

9.8.1.10. new-univ-crl

This parameter may be included on the response primitive. It is passed on the confirm primitive. It contains the new CRL (held by the KMS) of the new universal.

9.8.1.11. init-kid

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the identifier the initiator KMAP associates with the TEK used for this exchange.

9.8.1.8. result

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the success or failure of the operation.

9.9. K-GET-CRL Service

The cooperating KMAP uses the K-GET-CRL service to request the CRL from a SDNS entity. It is a confirmed service.

9.9.1. K-GET-CRL Parameters

Table 10 lists the K-GET-CRL parameters.

Table 10 -- K-GET-CRL parameters

Parameter name	req	ind	rsp	cnf
resp-kid	M	M(=)		
crl-ver	M	M(=)		
init-kid			M	M(=)
new-crl*			U	U(=)
currentcrl*			U	U(=)
result			M	M(=)

* Must choose one of these

9.9.1.1. resp-kid

This parameter is required on the request primitive. It is passed on the indication primitive. It contains the identifier the responder KMAP associates with the TEK.

9.9.1.2. crl-ver

This parameter is required on the request primitive. It is passed on the indication primitive. On the request and indication primitives, it contains the CRL version number of the CRL held by the initiator KMAP.

9.9.1.3. init-kid

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the identifier the initiator KMAP associates with the TEK.

9.9.1.4. new-crl

This parameter may be included on the response primitive. It is passed on the confirm primitive. It contains the signed CRL held by the responder KMAP.

9.9.1.5 current-crl

This parameter may be included on the response primitive. It is passed on the confirm primitive. It contains the CRL version number held by the responder KMAP.

9.9.1.6. result

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains a notification of the success or failure of the operation.

10. Sequencing Information for KMASE Services

This section defines the interaction among the KMP services.

10.1. K-EXCH-CRED Sequencing Information

10.1.1. Type of Service

K-EXCH-CRED is a confirmed service.

10.1.2. Usage Restrictions

Only the initiator of the application association may request the K-EXCH-CRED service, except after a K-TRY-ME or K-NO-KEY service. After a K-TRY-ME or K-NO-KEY service, only the responder of the application association service may request the K-EXCH-CRED.

The K- EXCH-CRED service must be (*or must have been) invoked prior to the use of the following services:

- o K-TEK-ATTRIBUTES
- o K-IREKEY
- o K-REQUEST-SREKEY
- o K-DELIVER-SREKEY
- o K-GET-CRL
- o K-UPDATE-TEK*

10.1.3. Disrupted Services

The K-EXCH-CRED service does not disrupt any services.

10.1.4. Disrupting Services

The A-ABORT and the A-P-ABORT services disrupt the K-EXCH-CRED service.

10.2. K-TRY-ME Service Sequencing Information

10.2.1. Type of Service

K-TRY-ME is an unconfirmed service.

10.2.2. Usage Restrictions

Only the initiator of the application association may request the K-TRY-ME service.

10.2.3. Disrupted Services

The K-TRY-ME service does not disrupt any services.

10.2.4. Disrupting Services

The A-ABORT and the A-P-ABORT services disrupt the K-TRY-ME service.

10.2.5. Further Sequencing Information

Following a K-TRY-ME request, the requestor must make no further requests on the application association.

10.3. K-NO-KEY Service Sequencing Information

10.3.1. Type of Service

K-NO-KEY is an unconfirmed service.

10.5.2. Usage Restrictions

Only the initiator of the association may request the K-UPDATE-TEK service.

10.5.3. Disrupted Services

The K-UPDATE-TEK service does not disrupt any services.

10.5.4. Disrupting Services

The A-ABORT and the A-P-ABORT services disrupt the K-UPDATE-TEK service.

10.6. K-IREKEY Service Sequencing Information

10.6.1. Type of Service

K-IREKEY is a confirmed service.

10.6.2. Usage Restrictions

Only the requester of the credential exchange may use the K-IREKEY after a credential exchange.

10.6.3. Disrupted Services

The K-IREKEY service does not disrupt any services.

10.6.4. Disrupting Services

The A-ABORT and the A-P-ABORT services disrupt the K-IREKEY service.

10.6.5. Further Sequencing Information

The KMAP may use the K-IREKEY service multiple times following a single credential exchange. Following a K-IREKEY service, the KMAP may use no other key management services on the same application association.

10.7. K-REQUEST-SREKEY Service Sequencing Information

10.7.1. Type of Service

K-REQUEST-SREKEY is a confirmed service.

10.7.2. Usage Restrictions

Only the initiator, after a successful credential exchange, may request the K-REQUEST-SREKEY service.

10.7.3. Disrupted Services

The K-REQUEST-SREKEY service does not disrupt any services.

10.3.2.Usage restrictions

Only the initiator of the application assocation may request the K-NO-KEY service.

10.3.3. Disrupted services

The K-NO-KEY service does not disrupt any services.

10.3.4. Disrupting services

The A-ABORT and the A-P-ABORT services disrupt the K-NO-KEY service.

10.3.5. Further Sequencing Information

Following a K-NO-KEY request, the requestor must make no further requests on the application association.

10.4. K-TEK-ATTRIBUTES Sequencing Information

10.4.1. Type of Service

K-TEK-ATTRIBUTES is a confirmed service.

10.4.2. Usage Restrictions

The KMAP may only use the K-TEK-ATTRIBUTES service after a successful credential exchange. Only the initiator of the application association may request the K-TEK-ATTRIBUTES, except following a K-TRY-ME or K-NO-KEY service. After a K-TRY-ME or K-NO-KEY service, only the responder of the application association may request the K-TEK-ATTRIBUTES service.

10.4.3. Disrupted Services

The K-TEK-ATTRIBUTES service does not disrupt any services.

10.4.4. Disrupting Services

The A-ABORT and the A-P-ABORT services disrupt the K-TEK-ATTRIBUTES service.

10.4.5 Further Sequencing Information

Following the use of a K-TEK-ATTRIBUTES service, no further services (with the exception of the K-UPDATE-TEK, K-TRY-ME, and K-NO-KEY services) may be requested on the application association until a K-EXCH-CRED service is performed.

10.5. K-UPDATE-TEK Sequencing Information

10.5.1. Type of Service

K-UPDATE-TEK is a confirmed service.

10.7.4. Disrupting Services

The A-ABORT and the A-P-ABORT services disrupt the K-REQUEST-SREKEY service.

10.7.5. Further Sequencing Information

The KMAP may use the K-REQUEST-SREKEY service multiple times following a single credential exchange. Following a K-REQUEST-SREKEY service, no other key management services may be used on the same application association.

10.8. K-DELIVER-SREKEY Service Sequencing Information

10.8.1. Type of Service

K-DELIVER-SREKEY is a confirmed service.

10.8.2. Usage Restrictions

Only the initiator, after a successful credential exchange, may request the K-DELIVER-SREKEY service.

10.8.3. Disrupted Services

The K-DELIVER-SREKEY service does not disrupt any services.

10.8.4. Disrupting Services

The A-ABORT and the A-P-ABORT services disrupt the K-DELIVER-SREKEY service.

10.8.5. Further Sequencing Information

The KMAP may use the K-DELIVERY-SREKEY service multiple times following a single credential exchange. Following a K-DELIVER-SREKEY service, no other key management services may be used on the same application association.

10.9. K-GET-CRL Service Sequencing Information

10.9.1. Type of Service

K-GET-CRL is a confirmed service.

10.9.2. Usage Restrictions

Only the initiator, after a successful credential exchange, may request the K-GET-CRL service.

10.9.3. Disrupted Services

The K-GET-CRL service does not disrupt any services.

10.9.4. Disrupting Services

The A-ABORT and the A-P-ABORT services disrupt the K-GET-CRL service.

11. Sequencing Information for ACSE Services

In addition to the disrupted services defined in ISO DIS 8649, the A-ABORT and A-P-ABORT services disrupt all KMP services.

SDNS

Secure Data Network System

Key Management Protocol

Specification of the protocol for services provided by the Key Management Application Service Element

Source: SDNS Protocol and Signaling Working Group
Key Management Sub-Group

Introductory note :

This document provides the framework for the SDNS Key Management Protocol Specification. It is subject to change after the development phase of SDNS.

Table of Contents

0. Introduction

This document specifies the protocol for the services provided to the Key Management Application Process (KMAP) by an application service element - the Key Management Application Service Element (KMASE) - to support applications in a distributed open systems environment.

Key management provides for the generation, distribution, and updating of traffic encryption keys (TEKs). Some management capabilites for authentication and access control are provided by the KMAP.

1. Scope and Field of Application

This document specifies the protocol and procedures for the Key Management Application Service Element. The KMASE services are provided in conjunction with the Association Control Service Element (ACSE) and the presentation service.

The KMASE procedures are defined in terms of:

a) the interactions between cooperating KMASE protocol machines through the use of the presentation service; and

b) the interactions between the KMASE protocol machine and its service user.

This document specifies conformance requirements for systems implementing these procedures.

2. References

ISO 7498/1	Information Processing Systems - Open Systems Interconnection - Basic Reference Model.
ISO 7498/2	Information Processing Systems - Open Systems Interconnection - Security Architecture.
ISO DIS 8649	Information Processing Systems - Open Systems Interconnection - Association Control Element Service Definition.
ISO DIS 8650	Information Processing Systems - Open Systems Interconnection - Association Control Protocol Specificaton.
ISO 8348	Information Processing Systems - Data communications - Network Service Definition.
ISO DIS 8822	Information Processing Systems - Open Systems Interconnection - Connection oriented presentation service definition.
ISO DIS 8824	Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).
ISO DIS 8825	Information Processing Systems - Open Systems Interconnection - Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1).

SDN.801	SDNS Access Control Concept Document.
SDN.802	SDNS Access Control Specification.
SDN.902	SDNS Key Management Protocol: Definition of Services provided by the Key Management Application Service Element.
SDN.906	SDNS Traffic Key Attribute Negotiation.

3. Definitions

This document uses definitions contained in the Reference Model for Open Systems Interconnection (IS 7498/1) including the Security Architecture (IS 7498/2).

In addition the following definitions are used:

Credentials -- information exchanged between SDNS component which authenticates the component and is used to form a traffic encryption key between the components.

Certificate Revocation List -- the list of key material identifiers which have been invalidated.

key material -- information supplied by the KMS that contains a component's credentials and private information.

Initiator -- the KMAP which initiates the establishment of the application association.

Rekey -- the process by which new key material is obtained from the KMS either directly or via a rekey agent.

Rekey Agent -- a SDNS component authorized to act as an intermediary between the KMS and other SDNS components for the delivery of key material.

Responder -- the KMAP which responds to a request for establishment of the application association.

Signed -- a method used to provide authentication of the originator of a message.

Universal-ID -- identifies the universal to which a credential belongs.

4. Abbreviations

4.1. Data Units

KPDU Key Management Protocol Data Unit

4.2. Types of Key Management Protocol data units

Nkrq	New Key Request KPDU
Nkrs	New Key Response KPDU
Tryme	Try Me addressing KPDU
Nokey	No Valid Key KPDU
EKpdu	Encrypted KPDU
Ssrq	Security Service Request KPDU
Ssrs	Security Service Response KPDU
Rkrq	Interactive ReKey Request KPDU
Rkrs	Interactive ReKey Response KPDU
Srkrq	Staged ReKey Request KPDU
Srkrs	Staged ReKey Response KPDU
Srdrq	Staged Rekey Delivery Request KPDU
Srdrs	Staged Rekey Delivery Response KPDU
Crrq	CRL Exchange Request KPDU
Crrs	CRL Exchange Response KPDU
Kurq	Key Update Request KPDU
Kurs	Key Update Response KPDU
Estat	Encrypted Status KPDU

4.3. Other Abbreviations

ACSE	Association Control Service Element
ASE	Application Service Element
CRL	Certificate Revocation List
K-MIB	Key Management Information Base
ICV	Integrity Check Value
KID	Key Identifier
KMS	Key Management Center
KMAE	Key Management Application Entity
KMAP	Key Management Application Process
KMASE	Key Management Application Service Element
KMP	Key Management Protocol
PDU	Protocol Data Unit
SDNS	Secure Data Network System
TEK	Traffic Encryption Key
TEK-MIB	Traffic Encryption Key Management Information Base

5. Conventions

Each procedure in this specification lists the KPDUs and the parameters used by the procedure. The structure of each KMASE KPDU is specified in clause 9 using the abstract syntax notation of ISO DIS 8824.

6. Overview of the Protocol

6.1 Service Provision

The protocol specified in this document provides the KMASE services defined in the SDNS Key Management Protocol: Definition of Services provided by the Key Management Application Service Element (SDN.902).

These services are listed in Table 1.

Table 1 -- KMASE Services

Service	Type
K-EXCH-CRED	Confirmed
K-TRY-ME	Unconfirmed
K-NO-KEY	Unconfirmed
K-TEK-ATTRIBUTES	Confirmed
K-UPDATE-TEK	Confirmed
K-IREKEY	Confirmed
K-REQUEST-SREKEY	Confirmed
K-DELIVERY-SREKEY	Confirmed
K-GET-CRL	Confirmed

6.2. Use of Services

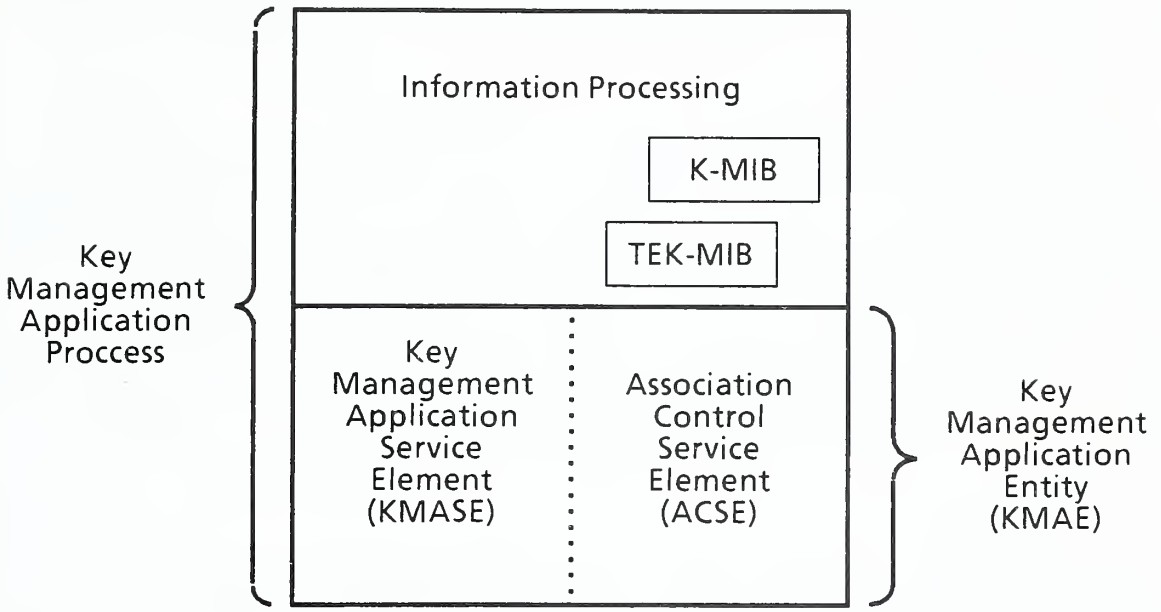
The Key Management Protocol needs a transfer service to pass information in the form of KMP KPDUs between cooperating KMAEs. This service is provided by the KMASE. This specification assumes an existing application association, established and released by means of ACSE. The Key Management Protocol defined in this specification assumes the use of the connection-oriented presentation services (ISO DIS 8822) and the ACSE services (ISO DIS 8649).

6.2.1. ACSE Services

The KMASE requires access to the A-ABORT and A-RELEASE services.

6.2.2. Use of the Presentation-service

The KMASE requires access to the P-DATA service.



KMP Model

6.3. Model

The KMAP comprises two parts: the information processor part and the communication part represented by the KMAE. When communication is required between two KMAPs, an application association is established between invocations of the processes' KMAEs. The KMAE represents an addressable set of communication capabilities which are defined by the KMASE and the ACSE.

During the use of the KMASE services, the existence of an application association between cooperating KMAEs is presumed. The nature and purpose of communication over the application association are determined by the information processing context and the application context which are established for it.

Note that each application association may be identified in an end system by an internal, implementation dependent, mechanism so that the KMASE service user, the KMASE, and the ACSE can refer to it.

The KMASE is driven by service request and response primitives from the KMAP and by indication primitives from the presentation service provider. The KMASE, in turn, issues indication and confirm primitives to the KMAP and request primitives to the ACSE and the presentation service provider.

The KMAP administers the K-MIB and the TEK-MIB in the sense of updating data in the IB upon receipt of indications and confirms and retrieving parameters from the IB when issuing requests and responses. K-MIB is a security MIB used to store key material. TEK-MIB is a security MIB used to store TEK, key identifiers (KIDs) and attributes regarding the use of the TEK.

7. Elements of Procedure

This section contains the elements of procedure of the Key Management Protocol. The procedures define the transfer of KPDUs whose abstract syntax is defined in Section 9.

7.1. Exchange Credentials

7.1.1. Purpose

This procedure is used to exchange the credentials necessary to form a traffic encryption key (TEK) for communications between SDNS components.

7.1.2. PDUs and Parameters

This procedure uses the following KPDUs and parameters:

- o Nkrq
  - **init-kid**
  - **univ-id**
  - **init-cred**

- o Nkrs
  - **init-kid**
  - **resp-kid**
  - **resp-cred**

7.1.3. Procedure

The credentials required to establish a TEK are exchanged by one KMASE (the initiator) transmitting a Nkrq KPDU to the other KMASE (the responder), which replies with a Nkrs KPDU.

The initiator KMASE uses the parameters from the K-EXCH-CRED request primitive to build a Nkrq KPDU. This KPDU contains the initiator's KID for the new TEK (**init-kid**), the universal-ID (**univ-id**), and the initiator's credentials (**init-cred**). This Nkrq KPDU is sent to the responder KMASE.

On receipt of an Nkrq, the responder KMASE issues a K-EXCH-CRED indication primitive to the KMAP. The responder KMASE now expects a K-EXCH-CRED response primitive from the KMAP containing the responder KID and the responder's credentials.

The responder KMASE uses the parameters from the K-EXCH-CRED response primitive to prepare a Nkrs KPDU, when the parameters indicate a successful exchange. This KPDU contains the initiator's KID for the new TEK (**init-kid**), the responder's KID for the new TEK (**resp-kid**), and the responder's credentials (**resp-cred**). The Nkrs KPDU is sent to the initiator KMASE. If the K-EXCH-CRED response primitive indicates a failure result, the responder KMASE issues an A-ABORT request primitive to the ACSE.

When the initiator KMASE receives the Nkrs KPDU, it issues a K-EXCH-CRED confirm primitive to the KMAP indicating the success of the exchange credentials procedure.

7.2. Try Me Notification

7.2.1. Purpose

The Try Me procedure is a notification to a SDNS KMASE that, to reach a specified end-system, the KMASE may establish a key with the SDNS component sending this message.

7.2.2. PDUs and Parameters

This procedure uses the following KPDUs and parameters:

- **Tryme**
  - **end-sys-addr**

7.2.3. Procedure

The initiator KMASE uses the parameters from the K-TRY-ME request primitive to prepare a Tryme KPDU. This KPDU contains the end-system address served by the initiator KMASE (**end-sys-addr**). The initiator KMASE sends the Tryme KPDU to the cooperating KMASE. The initiator KMASE does not expect any response.

When the responder KMASE receives a Tryme KPDU, the responder KMASE issues a K-TRY-ME indication primitive to the KMAP. The responder KMASE does not expect a K-TRY-ME response primitive.

7.3. No Key Notification

7.3.1. Purpose

The No Key procedure is a notification to a SDNS KMASE that traffic has been received which is protected under an invalid key.

7.3.2. PDUs and Parameters

This procedure uses the following KPDUs and parameters:

- **Nokey**
  - **rcvd-kid**
  - **pdu-count**

7.3.3 Procedure

The initiator KMASE uses the parameters from the K-NO-KEY request primitive to prepare a Nokey KPDU. This KPDU contains the invalid KID which has been received (**rcvd-kid**) and the number of received KPDUs protected under that key during a certain time period (**pdu-count**). The Nokey KPDU is sent to the cooperating KMASE. The initiator KMASE does not expect a response.

When the responder KMASE receives a Nokey KPDU, it issues a K-NO-KEY indication primitive to the responder KMAP. The responder KMASE does not expect a response primitive.

7.4. Encrypted KPDU Procedure

7.4.1. Purpose

The Encrypted KPDU Procedure is used by either the initiator or responder KMASE to encrypt and transfer KPDUs.

7.4.2. PDUs and Parameters

This procedure uses the following KPDUs and parameters:

- **EKpdu**
  - **remote-kid***
  - **contents****
- **PlainTextKpdu**
  - **plaintext*****
  - **icv**
  - **padding**

***NOTE:** The remote-kid is either the init-kid or the resp-kid depending on context of the KPDU.

****NOTE:** The contents parameter is the result of the encryption function.

*****NOTE:** The plaintext parameter consists of one of the following KPDUs:

- Ssrq
- Ssrs
- Rkrq
- Rkrs
- Srkrq
- Srkrs
- Srdrq
- Srdrs
- Crrq
- Crrs
- Kurq
- Kurs
- Estat

7.4.3. Procedure

The PlainText Kpdu comprises the plaintext, the icv and optional padding. The plaintext is one of the following KPDUs:

- o Ssrq
- o Ssrs
- o Rkrq
- o Rkrs
- o Srkrq
- o Srkrs
- o Srdrq
- o Srdrs
- o Crrq
- o Crrs
- o Kurq
- o Kurs
- o Estat

The KMASE produces the icv as a function of the parameter plaintext. The KMASE appends any optional padding that is required. The KMASE encrypts the PlainText Kpdu and the result of encryption is the parameter contents.

The EKpdu comprises the KID (**remote-kid**) and contents. The KMASE forms the EKpdu by prepending the remote-kid to the contents, and then the formation of the EKpdu is complete.

When a KMASE receives an EKpdu, it uses the remote-kid to select the correct TEK to decrypt the contents. After decryption, the result is a PlainText Kpdu, which is one of the above possible KPDUs and an icv. The KMASE then verifies the icv.

If the receiving KMASE does not possess the TEK identified by the remote-kid, the receiving KMASE issues an A-ABORT. Also, if decryption fails or if the icv fails verification, the KMASE issues an A-ABORT.

NOTE: The decryption of the EKpdu verifies the cryptographic compatibility of the TEK established by the Nkrq and Nkrs exchange.

NOTE: This procedure uses the basic encoding rules defined in ISO DIS 8825 to form the PlainText Kpdu prior to encryption and the EKpdu after encryption.

7.5. Attribute Negotiation

7.5.1. Purpose

The Ssrq and Ssrs KPDU exchange is used to verify the cryptographic compatibility of the TEK established by the Nkrq and Nkrs exchange, to negotiate and establish the communication and security services associated with the TEK, and to exchange Certificate Revocation List (CRL) version numbers.

7.5.2. PDUs and parameters

This procedure uses the following KPDUs and parameters:

- o Ssrq
  - **old-resp-kid**
  - **crl-ver**
  - **proposed-options**
  - **add-info**
- o Ssrs
  - **crl-ver**
  - **selected-options**
  - **add-info**
- o Estat
  - **estatus**
  - **doNotTryBefore**

7.5.3. Procedure

Security and communication attributes are negotiated by means of one KMASE (the initiator) transmitting a Ssrq KPDU to the other KMASE (the responder), which replies with a Ssrs KPDU. The initiator specifies a list of sets of negotiable communication and security parameter options in the Ssrq KPDU. The responder specifies the selected set of options in the Ssrs KPDU.

The initiator KMASE uses the parameters from the K-TEK-ATTRIBUTES request primitive to prepare a Ssrq KPDU. The Ssrq KPDU contains the old responder KID.(**old-resp-kid**), the initiator's CRL version number (**crl-ver**), option sets (**proposed-options**), and additional information (**add-info**). Option sets contain all sets of options that the initiator will allow. Option set preference is determined by first-to-last precedence in the ordering. After forming the Ssrq KPDU, the initiator KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, and encodes the result into the EKpdu format. The cryptographic key used for both the icv and the encryption is the previously formed TEK. The KMASE sends the EKpdu to the responder KMASE.

On receipt of a EKpdu, the responder KMASE performs the EKpdu decryption and icv verification. If either operation fails, the KMASE issues an A-ABORT request primitive to the ACSE. If both decryption and icv verification are successful, the KMASE issues a K-TEK-ATTRIBUTES indication primitive to the responder KMAP. The KMASE now expects a K-TEK-ATTRIBUTES response primitive.

The responder KMASE uses the parameters from the K-TEK-ATTRIBUTES response primitive to prepare a Ssrs KPDU when the parameters indicate successful attribute negotiation. The Ssrs KPDU contains the responder's CRL version number (**crl-ver**), option set (**selected-options**), and additional information (**add-info**). The Option Set contains the set of communication and security option parameters chosen from the initiator's sets in the Ssrq. After forming the Ssrs KPDU, the KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE. The cryptographic key used for both the icv and the encryption is the previously formed TEK. If the parameters of the K-TEK-ATTRIBUTES response primitive indicate a

failure of the attribute negotiation operation, the responder KMASE prepares an Estat KPDU instead of the Ssrs KPDU. The Estat KPDU contains a status (**estatus**). When the failure is service-not-available-at-this-time, the earliest time the initiator KMAP should attempt the request again is included (**doNotTryBefore**). After forming the Estat KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format, and sends it to the initiator KMASE.

When the initiator KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the initiator KMASE issues an A-ABORT request primitive to the ACSE. If both the decryption and icv verification are successful, the KMASE issues a K-TEK-ATTRIBUTES confirm primitive to the initiator KMAP indicating the success or failure of the attribute negotiation operation.

7.5.4. Option Sets

The options which are negotiated by the exchange of the Ssrq and Ssrs KPDUs establish some of the characteristics of the communication between peer entities which share a traffic key. The negotiated options are represented as sets of parameters within the Ssrq/Ssrs KPDUs. Some of the options are dependent on other options.

Note - see SDN.906, SDNS Attribute Negotiation, for further information.

7.5.5. add-info

In addition to the negotiation of options, the Ssrq and Ssrs exchange may provide the communicating entities with some additional information. This information is optional and its use by the recipient is a local matter. Some of the information may be signed by the KMS or an authorized directory service. The following sections describe some of the possible information which may appear in the Ssrq and Ssrs exchanges.

Note - see SDN.906, SDNS Traffic Key Attribute Negotiation, for further information.

7.5.6. Other PDU Fields

Also included in the Ssrq and Ssrs are fields containing information that is exchanged. These are:

Old Responder KID (old-resp-kid) - notify the responder that the initiator wishes to associate the options of a current TEK with the newly formed TEK and use them on the existing cryptographic association. The proposed options field in the Ssrq is absent. The original TEK is then zeroized. This parameter is present in the Ssrq and is set only by the Initiator of the TEK. If the responder KMAP is successful in associating the correct set of attributes with the newly formed TEK, those attributes are represented in the selected-options parameter of the Ssrs.

CRL version number (crl-ver) - contains notification of the initiator's CRL version in the Ssrq and the responder's CRL version in the Ssrs.

7.6. Traffic Key Update

7.6.1. Purpose

The Traffic Key Update procedure is used to obtain a new common traffic encryption key from an existing traffic key. All TEK attributes associated with the old TEK are assigned to the new key. The new key replaces the existing key.

7.6.2. PDUs and parameters

This procedure uses the following KPDUs and parameters:

- o **Kurq**
  - **new-init-kid**

- o **Kurs**
  - **new-resp-kid**

7.6.3. Procedure

A new traffic key is generated by means of one KMASE (the initiator) transmitting a Kurq KPDU to the other KMASE (the responder), which replies with a Kurs KPDU. The TEK attributes associated with the original traffic key are associated with the new TEK.

The initiator KMASE uses the parameters from the K-UPDATE-TEK request primitive to prepare a Kurq KPDU for the responder KMASE. The Kurq KPDU contains the KID the initiator has assigned to the new key (**new-init-kid**). After forming the Kurq KPDU, the KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format. The cryptographic key used for both the icv and the encryption is the existing TEK. The KMASE sends the EKpdu to the responder KMASE.

When the responder receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the responder KMASE issues an A-ABORT request primitive to the ACSE. If both the decryption and icv verification operations are successful, the responder KMASE issues a K-UPDATE-TEK indication primitive to the KMAP. The responder KMASE now expects a K-UPDATE-TEK response primitive from the KMAP.
The responder KMASE uses the parameters from the K-UPDATE-TEK response primitive to prepare a Kurs KPDU, when the K-UPDATE-TEK response primitive indicates that the key update operation is successful. The Kurs KPDU contains the KID the responder has assigned to the updated TEK (**new-resp-kid**). After forming the Kurs KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE. The cryptographic key used for both the icv and the encryption is the newly formed TEK. If the K-UPDATE-TEK response primitive indicates a failure of the key update operation, the responder KMASE issues an A-ABORT request primitive to the ACSE.

When the initiator KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the initiator KMASE issues an A-ABORT request primitive to the ACSE. If both the decryption and icv verification are successful, the KMASE issues a K-UPDATE-TEK confirm primitive to the initiator KMAP indicating the success of the key update operation.

7.7. Interactive Rekey

7.7.1. Purpose

The Interactive Rekey procedure is used by the KMASE of a SDNS component to directly and interactively exchange rekey information with the KMASE in a KMS.

7.7.2. PDUs and parameters

This procedure uses the following KPDUs and parameters:

- **o Rkrq**
  - **- univ-id**
  - **- rekey-cred**
  - **- ekr**
  - **- crl-ver**
  - **- ref-num**

- **o Rkrs**
  - **- kms-cred-A**
  - **- kms-cred-B**
  - **- new-k-mat-A**
  - **- new-k-mat-B**
  - **- ekr**
  - **- crl-ver**
  - **- ref-num**
  - **- new-crl**
  - **- new-univ-crl**
  - **- no-more**

- **o Estat**
  - **- estatus**
  - **-doNotTryBefore**

7.7.3. Procedure

A SDNS component obtains new key material by means of the KMASE (the initiator) transmitting a Rkrq KPDU to the KMS KMASE (the responder), which replies with a Rkrs KPDU.

The initiator KMASE uses the parameters from the K-IREKEY request primitive to prepare a Rkrq KPDU. This KPDU contains the universal-ID associated with the key material to be replaced (**univ-id**), and the current CRL version number held by the initiator SDNS component and associated with the universal-ID (**crl-ver**). If the initiator is a rekey agent, the Rkrq KPDU may also contain the credentials to be rekeyed (**rekey-cred**). If the component is expecting an electronic key replacement, the ekr parameter (**ekr**) is set to TRUE . Additionally, if this is the second part of a staged rekey procedure, the Rkrq KPDU contains the reference number (**ref-num**). After forming the Rkrq KPDU, the initiator KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the KMS KMASE.

When the KMS KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the KMS KMASE issues an A-ABORT request primitive to the ACSE. If both decryption and icv verification are successful, the KMS KMASE issues a K-IREKEY indication primitive to the KMS KMAP. The KMS KMASE now expects a K-IREKEY response primitive.

The KMS KMASE uses the parameters from the K-IREKEY response primitive to prepare a Rkrs KPDU, when the rekey operation is successful. This KPDU contains the KMS credentials (**kms-cred-A and kms-cred-B**), the new key material (**new-k-mat-A and new-k-mat-B**), and the current KMS CRL version number associated with the universal-ID under which this rekey was performed (**crl-ver**). If the KMS CRL version is more recent than the initiator's CRL version, the Rkrs also contains the new CRL signed by the KMS (**new-crl**). If the component is also receiving an electronic key replacement, the ekr parameter (**ekr**) is set to TRUE. Additionally, if this is the second part of a staged rekey procedure, the Rkrs KPDU contains the reference identifier (**ref-num**). If this rekey process is a universal rekey, the Rkrs KPDU also contains the new universal key material in the new-k-mat fields (**new-k-mat-A and new-k-mat-B**) and CRL of the new universal (**new-univ-crl**). Optionally, the Rkrs KPDU may contain a flag to indicate that the KMS cannot accept any further rekey requests on this association at this time (**no-more**). After forming the Rkrs KPDU, the KMS KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE. If the K-IREKEY response primitive indicates a failure of the interactive rekey operation, the KMS KMASE prepares a Estat KPDU instead of a Rkrs KPDU. The Estat KPDU contains a status (**estatus**). When the failure is service-not-available-at-this-time, the earliest time the initiator KMAP should attempt the request again is included (**doNotTryBefore**). After forming the Estat KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE.

When the initiator KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the initiator KMASE issues an A-ABORT request primitive to the ACSE. If both the decryption and icv verification are successful, the KMASE issues a K-IREKEY confirm primitive to the initiator KMAP indicating the success or failure of the rekey operation.

7.8. Staged Rekey

7.8.1. Staged Rekey Request

7.8.1.1. Purpose

The Staged Rekey Request procedure is used by the KMASE of a SDNS component to pass rekey information to the KMASE of a Rekey Agent. This is useful in an environment where direct access to the KMS is unavailable.

> **Note:** The staged rekey procedure is defined only for an exchange between a rekey agent and a "terminal" SDNS component. Exchanges between two rekey agents are for further study.

7.8.1.2. PDUs and parameters

This procedure uses the following KPDUs and parameters:

- **Srkrq**
  - **univ-id**
  - **rekey-cred**
  - **crl-ver**
  - **ekr**
- **Srkrs**
  - **ref-num**
- **Estat**
  - **estatus**
  - **doNotTryBefore**

7.8.1.3. Procedure

A SDNS component may request new key material from a Rekey Agent by the KMASE (the initiator) transmitting a Srkrq KPDU to the Rekey Agent KMASE (the responder), which replies with a Srkrs KPDU.

The initiator KMASE uses the parameters from the K-REQUEST-SREKEY request primitive to prepare a Srkrq KPDU. The Srkrq KPDU contains the universal-ID associated with the key material to be replaced (**univ-id**) and the current CRL version number held by the initiator SDNS component and associated with the universal-ID (**crl-ver**). Additionally, the Srkrq KPDU contains the credentials that the KMS should use to perform the rekey operation (**rekey-cred**) and a notification of whether the initiator expects an electronic key replacement (**ekr**). After forming the Srkrq KPDU, the initiator KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the responder KMASE.

When the Rekey Agent KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the Rekey Agent KMASE issues an A-ABORT request primitive to the ACSE. If both decryption and icv verification are successful, the rekey agent KMASE issues a K-REQUEST-SREKEY-

indication primitive to the Rekey Agent KMAP. The Rekey Agent KMASE now expects a K-REQUEST-SREKEY response primitive from the KMAP.

The Rekey Agent KMASE uses the parameters from the K-REQUEST-SREKEY reponse primitive to prepare a Srkrs KPDU, when the K-REQUEST-SREKEY response primitive indicates a successful operation. The Srkrs KPDU contains the ref-num generated by the Rekey Agent (**ref-num**). After forming the Srkrs KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE.

If the K-REQUEST-SREKEY response primitive indicates a failure, the Rekey Agent KMASE prepares a Estat KPDU instead of a Srkrs KPDU. The Estat KPDU contains a status (**estatus**). When the failure is service-not-available-at-this-time, the earliest time the initiator KMAP should attempt the request again is included (**doNotTryBefore**). After forming the Estat KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE.

When the initiator KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the initiator KMASE issues an A-ABORT request primitive to the ACSE. If both the decryption and icv verifications are sucessful, the KMASE issues a K-REQUEST-SREKEY confirm primitive to the initiator KMAP indicating the success or failure of the Staged Rekey Request operation.

7.8.2. Staged Rekey Delivery

7.8.2.1. Purpose

The Staged Rekey Delivery procedure is used by the KMASE of a SDNS component to take delivery of rekey information from a Rekey Agent.

7.8.2.2. PDUs and parameters

This procedure uses the following KPDUs and parameters:

- o Srdrq
  - **ref-num**
- o Srdrs
  - **kms-cred-A**
  - **kms-cred-B**
  - **new-k-mat-A**
  - **new-k-mat-B**
  - **ekr**
  - **crl-ver**
  - **new-crl**
  - **new-univ-crl**
- o Estat
  - **estatus**
  - **doNotTryBefore**

7.8.2.3 Procedure

Delivery of new key material to a SDNS component from a Rekey Agent is accomplished by means of the initiator KMASE sending a Srdrq KPDU to the Rekey Agent KMASE (the responder), which replies with a Srkrs KPDU.

The initiator KMASE uses the parameters from the K-DELIVER-SREKEY request primitive to prepare a Srdrq KPDU. The Srdrq KPDU contains the reference number as generated by the Rekey Agent (**ref-num**). After forming the Srdrq KPDU, the initiator KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the responder KMASE.

When the Rekey Agent KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the Rekey Agent KMASE issues an A-ABORT request primitive to the ACSE. If both decryption and icv verification are successful, the Rekey Agent KMASE issues a K-DELIVERY-SREKEY indication primitive to the Rekey Agent KMAP. The Rekey Agent KMASE now expects a K-DELIVER-SREKEY response primitive.

The Rekey Agent KMASE uses the parameters from the K-DELIVER-SREKEY reponse primitive to prepare a Srdrs KPDU, when the K-DELIVER-SREKEY response primitive indicates a successful operation. The Srdrs KPDU contains the KMS credentials (**kms-cred-A and kms-cred-B**), the new key material (**new-k-mat-A and new-k-mat-B**). Also, the Srdrs KPDU contains the current crl version number associated with the universal-ID at the time the KMS performed the rekey operation (**crl-ver**). The new CRL (**new-crl**) may also be present. Additionally, the Srdrs KPDU contains a notification as to whether the initiator received an electronic key replacement (**ekr**). If this rekey process is a universal rekey, the Rkrs KPDU also contains the new universal key material in the new-k-mat fields (**new-k-mat-A and new-k-mat-B**) and CRL of the new universal (**new-univ-crl**). After forming the Srdrs KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE.

If the K-DELIVER-SREKEY response primitive indicates a failure, the Rekey Agent KMASE prepares a Estat KPDU instead of a Srdrs KPDU. The Estat KPDU contains a status (**estatus**). When the failure is service-not-available-at-this-time, the earliest time the initiator KMAP should attempt the request again is included (**doNotTryBefore**). After forming the Estat KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE.

When the initiator KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the initiator KMASE issues an A-ABORT request primitive to the ACSE. If both the decryption and icv verifications are sucessful, the initiator KMASE issues a K-DELIVER-SREKEY confirm primitive to the initiator KMAP indicating the success or failure of the deliver rekey operation.

7.9. CRL Retrieval

7.9.1. Purpose

The Crrq and Crrs KPDU exchange is used to request and send a copy of the CRL signed by the KMS.

7.9.2. PDUs and Parameters

This procedure uses the following KPDUs and parameters:

- o Crrq
  - crl-ver
- o Crrs
  - currentcrl
  - new-crl
- o Estat
  - estatus
  - doNotTryBefore

7.9.3. Procedure

The CRL list is retrieved by means of one KMASE (the initiator) transmitting a Crrq KPDU to the other KMASE (the responder), which replies with a Crrs KPDU.

The initiator KMASE uses the parameters from the K-GET-CRL request primitive to prepare a Crrq KPDU. The Crrq KPDU contains the version number of the initiator's crl (**crl-ver**). After forming the Crrq KPDU, the initiator KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the responder KMASE. The cryptographic key used for both the icv and the encryption is the previously formed TEK.

When the responder KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the responder KMASE issues an A-ABORT request primitive to the ACSE. If both decryption and icv verification are successful, the KMASE issues a K-GET-CRL indication primitive to the responder KMAP. The responder KMASE now expects a K-GET-CRL response primitive.

The responder KMASE uses the parameters from the K-GET-CRL response primitive to prepare a Crrs KPDU, when the K-GET-CRL parameters indicate a successful operation. The Crrs KPDU contains either the current signed CRL (for the universal associated with the current TEK) held by the responder SDNS component (**new-crl**) or the CRL version number (**currentcrl**) if the initiator's CRL version is current. After forming the Crrs KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format, and sends it to the initiator KMASE. If the K-GET-CRL response primitive indicates a failure of the get CRL operation (e.g. the responder cannot supply the CRL at this time, or does not support CRL exchanges), the responder KMASE prepares a Estat KPDU instead of a Crrs KPDU. The Estat KPDU contains a status (**estatus**). When the failure is service-not-available-at-this-time, the

earliest time the initiator KMAP should attempt the request again is included (**doNotTryBefore**). After forming the Estat KPDU, the responder KMASE encodes the KPDU into the PlainText Kpdu format, encrypts the PlainText Kpdu, encodes the result into the EKpdu format and sends it to the initiator KMASE.

When the initiator KMASE receives the EKpdu, it performs the EKpdu decryption and icv verification. If either operation fails, the initiator KMASE issues an A-ABORT request primitive to the ACSE. If both the decryption and icv verification are successful, the initiator KMASE issues a K-GET-CRL confirm primitive to the initiator KMAP indicating the success or failure of the get CRL operation.

7.10. Encrypted Status Procedure

7.10.1. Purpose

The respondor KMASE uses the Encrypted Status procedure when the KMAP issues a response primitve with a failure result.

7.10.2. PDU and Parameters

- **o Estat**
  - **estatus**
  - **doNotTryBefore**

7.10.3. Procedure

When the responder KMAP issues a response primitive with a failure result, the KMASE generates an Estat. The parameter estatus contains the result value, and the value represents a failure when it is greater than zero. The following codes represent the various failure responses:

```
service-not-available                  (8), -- No further details
service-not-available-CRL              (9),
service-not-available-Interactive-rekey (10),
service-not-available-Staged-rekey     (11),
service-not-available-at-this-time     (15), -- Specify doNotTryBefore
incorrect-value-in-protected-field     (16), -- No further details
incorrect-value-PDU-type               (17),
incorrect-value-old-resp-kid           (18),
incorrect-value-crl-ver                (19),
incorrect-value-add-info               (20),
incorrect-value-univ-id                (21),
incorrect-value-ref-num                (22),
incorrect-value-proposed-options       (23),
rekey-not-complete                     (24),
delete-this-TEK                        (126), -- Locally definable
negotiation-failure                    (127), -- No further details
failure                                (255)
```

When the KMASE receives an Estat KPDU, it delivers a confirm primitive with estatus, and optionally doNotTryBefore, as the result parameter.

The KMASE includes the doNotTryBefore parameter in the Estat when the failure is service-not-available-at-this-time. The parameter doNotTryBefore is UTC Time and is the earliest time the initiator KMAP should attempt the request again.

8. Mapping to Used Services

This clause defines how the presentation-service primitives described in ISO 8822 are used by the KMASE. Table 2 defines the mapping of the KMASE service primitives and KPDUs to the presentation-service primitives.

Table 2 -- Presentation-service Mapping Overview

KMP Service	KPDU	Presentation Service
K-EXCH-CRED req/ind	Nkrq	P-DATA request/indication
K-EXCH-CRED rsp/cnf	Nkrs	P-DATA request/indication
K-TRY-ME req/ind	Tryme	P-DATA request/indication
K-NO-KEY req/ind	Nokey	P-DATA.request/indication
K-TEK-ATTRIBUTES req/ind	Ssrq	P-DATA request/indication
K-TEK-ATTRIBUTES rsp/cnf	Ssrs	P-DATA request/indication
K-TEK-ATTRIBUTES rsp-/cnf-	Estat	P-DATA request/indication
K-UPDATE-TEK req/ind	Kurq	P-DATA request/indication
K-UPDATE-TEK rsp/cnf	Kurs	P-DATA request/indication
K-IREKEY req/ind	Rkrq	P-DATA request/indication
K-IREKEY rsp/cnf	Rkrs	P-DATA request/indication
K-IREKEY rsp-/cnf-	Estat	P-DATA request/indication
K-REQUEST-SREKEY req/ind	Srkrq	P-DATA request/indication
K-REQUEST-SREKEY rsp/cnf	Srkrs	P-DATA request/indication
K-REQUEST-SREKEY rsp-/cnf-	Estat	P-DATA request/indication
K-DELIVER-SREKEY req/ind	Srdrq	P-DATA request/indication
K-DELIVER-SREKEY rsp/cnf	Srdrs	P-DATA request/indication
K-DELIVER-SREKEY rsp-/cnf-	Estat	P-DATA request/indication
K-GET-CRL req/ind	Crrq	P-DATA request/indication
K-GET-CRL rsp/cnf	Crrs	P-DATA request/indication
K-GET-CRL rsp-/cnf-	Estat	P-DATA request/indication

Each KPDU is transferred as user data of the P-DATA service. The P-DATA service is an unconfirmed service.

The use of the P-DATA request and P-DATA indication primitive parameters is as follows:

User Data	The KPDU to be transferred. Its maximum size is not restricted in this mapping.

9. Abstract Syntax Definition of KPDUs

This section uses the Abstract Syntax Notation One specified in DIS 8824, Specification of Abstract Syntax Notation One (ASN.1), to define the PDUs exchanged by the key management protocol. This is done without determining the way an instance of this type is to be represented (by a sequence of octets) during transfer. Encoding rules are used to produce a transfer syntax for the types defined in the abstract syntax. The Encoding Rules to be used for the transfer syntax of the following Abstract Syntax are specified in DIS 8825, Specification of Basic Encoding Rules for ASN.1.

```
KMP DEFINITIONS :: =
BEGIN

IMPORTS
        AttributeSet    AttributeSet
        AddInfo         AddInfo

-- The main entry point.  All encrypted PDU's look alike
-- This notation assumes default EXPLICIT tagging

Kpdu :: = CHOICE {
  [0] IMPLICIT Nkrq,
  [1] IMPLICIT Nkrs,
  [2] IMPLICIT Tryme,
  [3] IMPLICIT Nokey,
  [4] IMPLICIT EKpdu
}

-- Various "simple" types.  More detailed definition is for further study

Crl-Ver                 :: = [APPLICATION 20] IMPLICIT INTEGER
IntegrityCheckValue     :: = [APPLICATION 21] IMPLICIT OCTET STRING
    -- IntegrityCheckValue should be encoded as primitive
Credentials             :: = [APPLICATION 22] IMPLICIT OCTET STRING
KMaterials              :: = [APPLICATION 23] IMPLICIT OCTET STRING
KeyId                   :: = [APPLICATION 24] IMPLICIT OCTET STRING
    -- KeyId is limited to four octets in length
KeyList                 :: = [APPLICATION 25] IMPLICIT OCTET STRING
Universal-id            :: = [APPLICATION 26] IMPLICIT INTEGER
SecurityLevel           :: = [APPLICATION 27] IMPLICIT INTEGER {
```

```
    -- SecurityLevel continued
    -- Encoding depends on defining authority.  The following values are defined for
        DoD
    unclassified(85),-- 0101 0101
    confidential(122),  -- 0111 1010
    secret(173),        -- 1010 1101
    top-secret(222)     -- 1101 1110
}

-- Misc supporting types

PduCount :: = SEQUENCE {
  pdus-rcvd  INTEGER,
  time       INTEGER,
  time-unit  INTEGER { seconds(0), minutes(1) }
}

-- The unencrypted PDU formats

-- New Key Request
Nkrq :: = SEQUENCE {
  init-cred   Credentials,
  univ-id     Universal-id,
  init-kid    KeyId
}

-- New Key Response
Nkrs :: = SEQUENCE {
  resp-cred     Credentials,
  init-kid      KeyId,
  resp-kid      KeyId
}

-- Try Me
Tryme :: = SET { end-sys-addr       OCTET STRING }


-- No Key
Nokey :: =  SET {
  rcvd-kid      KeyId,
  pdu-count     PduCount OPTIONAL
}
```

```
-- Encrypted PDU's.  Each has a KID in plaintext

EKpdu :: = SEQUENCE {
  remote-kid KeyId,
-- remote-kid is either init-kid or resp-kid depending on decrypted contents
  contents OCTET STRING  -- ENCRYPTED PlainText Kpdu
}

--The contents of an encrypted PDU.

PlainTextKpdu :: = SEQUENCE {
-- must not be encoded with indefinite length
  plaintext CHOICE {
              [0] IMPLICIT Ssrq,
              [1] IMPLICIT Ssrs,
result        [2] IMPLICIT Estat,
              [3] IMPLICIT Kurq,
              [4] IMPLICIT Kurs,
              [5] IMPLICIT Rkrq,
              [6] IMPLICIT Rkrs,
              [7] IMPLICIT Srkrq,
              [8] IMPLICIT Srkrs,
              [9] IMPLICIT Srdrq,
              [10] IMPLICIT Srdrs,
              [11] IMPLICIT Crrq,
              [12]          Crrs
  },
  padding OCTET STRING OPTIONAL,
  icv IntegrityCheckValue
} --Note icv is guaranteed to be last.
  --Note Indefinite length encoding of icv is prohibited.

-- Security Services Request

Ssrq :: =  SET {
  crl-ver               Crl-Ver,
                        NewOrOldOptions,
  add-info              AddInfo OPTIONAL
},
NewOrOldOptions :: =  CHOICE {
  old-resp-kid          KeyId ,
  proposed-options      SEQUENCE OF  AttributeSet ,
                            -- ordered by preference
} --Note AttributeSet and AddInfo will be imported.  See SDN.906, SDNS Traffic Key Attribute Negotiation.

-- Security Services Response

Ssrs :: = SET {
  crl-ver                   Crl-Ver,
  selected-options      [0] AttributeSet,
  add-info              [1] IMPLICIT AddInfo OPTIONAL
}
```

```
-- Encrypted Status

Estat :: = SET {
 estatus INTEGER
    {
  service-not-available                (8),  -- No further details
  service-not-available-CRL            (9),
  service-not-available-Interactive-rekey  (10),
  service-not-available-Staged-rekey   (11),
  service-not-available-at-this-time   (15), -- Specify doNotTryBefore
  incorrect-value-in-protected-field   (16), -- No further details
  incorrect-value-PDU-type             (17),
  incorrect-value-old-resp-kid         (18),
  incorrect-value-crl-ver              (19),
  incorrect-value-add-info             (20),
  incorrect-value-univ-id              (21),
  incorrect-value-ref-num              (22),
  incorrect-value-proposed-options     (23),
  rekey-not-complete                   (24),
  delete-this-TEK                      (126), -- Locally definable
  negotiation-failure                  (127), -- No further details
  failure                              (255) -- No further details
 },
 doNotTryBefore  UTCTime  OPTIONAL -- used during temporary problems
}

-- Key Update Request

Kurq :: = SET { new-init-kid  KeyId }

-- Key Update Response

Kurs :: = SET { new-resp-kid  KeyId }

-- Rekey Request (interactive)

Rkrq :: = SET {
  rekey-cred     Credentials OPTIONAL,
                     --if absent, use credentials associated with current TEK
  univ-id        Universal-id,
  crl-ver        Crl-Ver,
  ref-num        INTEGER OPTIONAL, --used if rekey-cred used
  ekr            BOOLEAN
}
```

```
-- Rekey Response (interactive)

Rkrs :: = SET {
   ref-num                INTEGER OPTIONAL,
   kms-cred-A         [0] IMPLICIT Credentials,
   new-k-mat-A        [1] IMPLICIT KMaterials,
   kms-cred-B         [2] IMPLICIT Credentials,
   new-k-mat-B        [3] IMPLICIT KMaterials,
   crl-ver                Crl-Ver,
   new-crl            [4] IMPLICIT KeyList OPTIONAL,
   new-univ-crl       [5] IMPLICIT KeyList OPTIONAL,
   no-more            [6] IMPLICIT BOOLEAN OPTIONAL, -- True means send no more
   ekr                    BOOLEAN
}

-- Staged Rekey Request

Srkrq :: = SET {
   rekey-cred     Credentials,
   univ-id        Universal-id,
   crl-ver        Crl-Ver,
   ekr            BOOLEAN
}

-- Staged Rekey Response

Srkrs :: = SET { ref-num  INTEGER }


-- Staged Rekey Delivery Request

Srdrq :: = SET { ref-num INTEGER }

-- Staged Rekey Delivery Response

Srdrs :: = SET {
   kms-cred-A         [0] IMPLICIT Credentials,
   new-k-mat-A        [1] IMPLICIT KMaterials,
   kms-cred-B         [2] IMPLICIT Credentials,
   new-k-mat-B        [3] IMPLICIT KMaterials,
   crl-ver                Ckl-Ver,
   new-crl            [4] IMPLICIT KeyList OPTIONAL,
   new-univ-crl       [5] IMPLICIT KeyList OPTIONAL,
   ekr                    BOOLEAN

}
```

```
-- Certificate Revocation List Request

Crrq :: = SET { crl-ver   Crl-Ver }

-- Certificate Revocation List Response

Crrs :: =   CHOICE {
  currentcrl       Crl-Ver,
  new-crl      KeyList
}


END   -- of KMP
```

10. Conformance

An implementation claiming conformance to SDNS Specification SDN.903 shall comply with the requirements in SECTIONS 10.1 through 10.3.

10.1. Statement Requirements

An implementor shall state the following:

a) the application for which conformance is claimed.

10.2. Static Requirements

The system shall:

a) conform to the abstract syntax definition of KPDU's defined in section 9.

10.3. Dynamic Requirements

The system shall:

a) conform to the elements of procedure defined in section 7,

b) conform to the mappings to the used services, for which conformance is claimed, as defined in section 8.

Annex A. KMP State Machine Description

This annex describes the key management protocol in terms of state tables. The state tables show how the protocol moves from one state to another according to the incoming events.

A.1. Conventions

The intersection of an incoming event (row) and a state (column) forms a cell.

In the state table, a blank cell represents the combination of an incoming event and a state that is not defined for the KMASE. Columns or rows which would be blank are omitted from the state tables. Additionally, the association release and abort events are omitted from the state tables because the actions to be taken on receipt of these events are the same regardless of the state.

A non-blank cell represents an incoming event and a state that is defined for the KMASE. Such a cell contains one or more action lists. An action list may be either mandatory or conditional. If a cell contains a mandatory action list, it is the only action list in the cell.

A mandatory action list contains:

a) optionally one or more outgoing events, and

b) a resultant state.

A conditional action list contains:

a) a predicate expression comprising predicates and Boolean operators (* represents the Boolean NOT), and

b) a mandatory action list. (This mandatory action list is used only if the predicate expression is true).

A.2. Actions to be taken by the KMASE

The KMASE state table defines the action to be taken by the KMASE in terms of optional outgoing events and the resultant state of the application-association.

A.2.1 Invalid Intersections

Blank cells indicate an invalid intersection of an incoming event and state. If such an intersection occurs, one of the following actions is taken:

a) If the incoming event comes from the KMP-user, any action taken by the KMASE is a local matter.

b) If the incoming event is related to a received APDU, PS- provider, or ACSE, the KMASE issues an A-ABORT request primitive to ACSE.

A.2.2 Valid Intersections

If the intersection of the state and incoming event is valid, one of the following actions is taken:

a) If the cell contains a mandatory action list, the KMASE takes the actions specified.

b) If a cell contains one or more conditional action lists, for each predicate expression that is true, the KMASE takes the actions specified. If none of the predicate expressions are true, the KMASE takes one of the actions defined in section A.2.1.

A.3. Incoming Events

Incoming Events

Abbreviated Name	Name and Description
EXCreq	K-EXCH-CRED request
EXCrsp+	K-EXCH-CRED response success
EXCrsp-	K-EXCH-CRED response failure
ATTreq	K-TEK-ATTRIBUTES request
ATTrsp+	K-TEK-ATTRIBUTES response success
ATTrsp-	K-TEK-ATTRIBUTES response failure
UPDreq	K-UPDATE-TEK request
UPDrsp+	K-UPDATE-TEK response success
UPDrsp-	K-UPDATE-TEK response failure
IREKEYreq	K-IREKEY request
IREKEYrsp+	K-IREKEY response success
IREKEYrsp-	K-IREKEY response failure
SREKEYreq	K-REQUEST-SREKEY request
SREKEYrsp+	K-REQUEST-SREKEY response success
SREKEYrsp-	K-REQUEST-SREKEY response failure
SDLIVRreq	K-DELIVER-SREKEY request
SDLIVRrsp+	K-DELIVER-SREKEY response success
SDLIVRrsp-	K-DELIVER-SREKEY response failure
CRLreq	K-GET-CRL request
CRLrsp+	K-GET-CRL response success
CRLrsp-	K-GET-CRL response failure
TRYMEreq	K-TRYME request
ASSOC_INIT	Notification of A-ASSOCIATE confirm success
ASSOC_RESP	Notification of A-ASSOCIATE response success
ASSOC_TER	Notification of A-RELEASE, A-ABORT,or A-P-ABORT request or indication
NKRQ	New key request KPDU
NKRS	New key response KPDU
SSRQ	Security services request KPDU
SSRS	Security services response KPDU
RKRQ	Rekey request KPDU
RKRS	Rekey response KPDU
SRKRQ	Request Staged ReKey request KPDU
SRKRS	Request Staged ReKey response KPDU
SRDRQ	Deliver Staged Rekey request KPDU
SRDRS	Deliver Staged Rekey response KPDU
CRRQ	Certificate Revocation List request KPDU
CRRS	Certificate Revocation List response KPDU
KURQ	Key update request KPDU
KURS	Key update response KDPU
ESTAT	Encrypted Status KPDU
TRYME	Tryme Addressing KPDU
NOKEY	No Valid Key KPDU

A.4. Outgoing Events

Outgoing Events

Abbreviated Name	Name and Description
EXCind	K-EXCH-CRED indication
EXCcnf+	K-EXCH-CRED confirm success
ATTind	K-TEK-ATTRIBUTES indication
ATTcnf+	K-TEK-ATTRIBUTES confirm success
ATTcnf-	K-TEK-ATTRIBUTES confirm failure
UPDind	K-UPDATE-TEK indication
UPDcnf+	K-UPDATE-TEK confirm success
UPDcnf-	K-UPDATE-TEK confirm failure
IREKEYind	K-IREKEY indication
IREKEYcnf+	K-IREKEY confirm success
IREKEYcnf-	K-IREKEY confirm failure
SREKEYind	K-REQUEST-SREKEY indication
SREKEYcnf+	K-REQUEST-SREKEY confirm success
SREKEYcnf-	K-REQUEST-SREKEY confirm failure
SDLIVRind	K-DELIVER-SREKEY indication
SDLIVRcnf+	K-DELIVER-SREKEY confirm success
SDLIVRcnf-	K-DELIVER-SREKEY confirm failure
CRLind	K-GET-CRL indication
CRLcnf+	K-GET-CRL confirm success
CRLcnf-	K-GET-CRL confirm failure
NOKEYind	K-NOKEY indication
TRYMEind	K-TRYME indication
NKRQ	New key request KPDU
NKRS	New key response KPDU
SSRQ	Security services request KPDU
SSRS	Security services response KPDU
RKRQ	Rekey request KPDU
RKRS	Rekey response KPDU
SRKRQ	Staged ReKey Request KPDU
SRKRS	Staged ReKey Response KPDU
SRDRQ	Deliver Rekey Request KPDU
SRDRS	Deliver Rekey Reponse KPDU
CRRQ	Certificate Revocation List KPDU
CRRS	Certificate Revocation List KPDU
KURQ	Key update request KPDU
KURS	Key update response KDPU
ESTAT	Encrypted Status KPDU
TRYME	Tryme Addressing KPDU
NOKEY	No Valid Key KPDU
A-ABORT	ACSE A-ABORT service request
A-RELEASE	ACSE A-RELEASE service request

A.5 Key Management Protocol Machine States

State Name	Description
IDLE	Unassociated, no application association in place
SLAVE_ASSOC	Associated, responder, application association in place
MSTR_ASSOC	Associated, initiator, application association in place
SLAVE_KEYED	Responder, successful credential exchange completed
MSTR_KEYED	Initiator, successful credential exchange completed
MSTR_REKEY	Initiator, successful credential exchange completed and only interactive rekeys may be performed
MSTR_SRKEY	Initiator, successful credential exchange completed and only staged rekeys may be performed.
WF_EXCrsp	Waiting for K_EXCH_CRED response primitive
WF_ATTrsp	Waiting for K_TEK_ATTRIBUTES response primitive
WF_CRLrsp	Waiting for K_GET_CRL response primitive
WF_UPDrsp	Waiting for K_UPDATE_TEK response primitive
WF_IREKEYrsp	Waiting for K_IREKEY response primitive
WF_NKRS	Waiting for NKRS KPDU
WF_SSRS	Waiting for SSRS KPDU
WF_CRRS	Waiting for CRRS KPDU
WF_KURS	Waiting for KURS KPDU
WF_RKRS	Waiting for RKRS KPDU
WF_SRKRS	Waiting for SRKRS KPDU
WF_SRDRS	Waiting for SRDRS KPDU
WF_SREKEYrsp	Waiting for K_SREKEY response primitive
WF_SDLIVRrsp	Waiting for K_DELIVER_REKEY response primitive

A.6 State Tables

	IDLE	MSTR_ASSOC	WF_NKRS	MSTR_KEYED	MSTR_REKEY	WF_SSRS	WF_CRRS	WF_KURS	WF_RKRS
ASSOC_INIT	*MSTR_ASSOC*								
EXCreq		NKRQ *WF_NKRS*							
NKRS			EXCcnf + *MSTR_KEYED*						
ATTreq				SSRQ *WF_SSRS*					
SSRS						ATTcnf + *MSTR_ASSOC*			
CRLreq				CRRQ *WF_CRRS*					
CRRS							CRLcnf + *MSTR_KEYED*		
UPDreq		KURQ *WF_KURS*							
KURS								UPDcnf + *MSTR_ASSOC*	
IREKEYreq				RKRQ *WF_RKRS*	RKRQ *WF_RKRS*				
RKRS									IREKEYcnf + *MSTR_REKEY*
ESTAT						ATTcnf- *MSTR_ASSOC*	CRLcnf- *MSTR_KEYED*		IREKEYcnf- *MSTR_REKEY*
TRYMEreq		TRYME *SLAVE_ASSOC*							
NOKEYreq		NOKEY *SLAVE_ASSOC*							
ASSOC_TERM	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*

KMP USER STATE TABLE (SLAVE)

	IDLE	SLAVE_ASSOC	WF_EXCrsp	SLAVE_KEYED	WF_ATTrsp	WF_CRLrsp	WF_UPDrsp
ASSOC_RESP	*SLAVE_ASSOC*						
NKRQ		EXCind *WF_EXCrsp*					
EXCrsp +			NKRS *SLAVE_KEYED*				
EXCrsp-			A-ABORT *IDLE*				
SSRQ				ATTind *WF_ATTrsp*			
ATTrsp +					SSRS *SLAVE_ASSOC*		
ATTrsp-					ESTAT *SLAVE_ASSOC*		
CRRQ				CRLind *WF_CRLrsp*			
CRLrsp +						CRRS *SLAVE_KEYED*	
CRLrsp-						ESTAT *SLAVE_KEYED*	
KURQ		UPDind *WF_UPDrsp*					
UPDrsp +							KURS *SLAVE_ASSOC*
UPDrsp-							A-ABORT *IDLE*
TRYME		TRYMEind *MSTR_ASSOC*					
NOKEY		NOKEYind *MSTR_ASSOC*					
ASSOC_TERM	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*

KMP KMS STATE TABLE (SLAVE)

	IDLE	SLAVE_ASSOC	WF_EXCrsp	SLAVE_KEYED	WF_CRLrsp	WF_IREKEYrsp
ASSOC_RESP	*SLAVE_ASSOC*					
NKRQ		EXCind *WF_EXCrsp*				
EXCrsp +			NKRS *SLAVE_KEYED*			
EXCrsp-			A-ABORT *IDLE*			
CRRQ				CRLind *WF_CRLrsp*		
CRLrsp +					CRRS *SLAVE_KEYED*	
CRLrsp-					ESTAT *SLAVE_KEYED*	
RKRQ				IREKEYind *WF_IREKEYrsp*		
IREKEYrsp +						RKRS *SLAVE_KEYED*
IREKEYrsp-						ESTAT *SLAVE_KEYED*
ASSOC_TERM	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*

KMP STAGED REKEY STATE TABLE (USER)

	IDLE	MSTR_ASSOC	WF_NKRS	MSTR_SRKEY	WF_SRKRS	WF_CRRS	WF_SRDRS
ASSOC_INIT	*MSTR_ASSOC*						
EXCreq		NKRQ *WF_NKRS*					
NKRS			EXCcnf + *MSTR_SRKEY*				
SREKEYreq				SRKRQ *(note 1)* *WF_SRKRS*			
SRKRS					SREKEYcnf + *MSTR_SRKEY*		
CRLreq				CRRQ *WF_CRRS*			
CRRS						CRLcnf + *MSTR_SRKEY*	
SDLIVRreq				SRDRQ *(note 1)* *WF_SRDRS*			
SRDRS							SDLIVRcnf + *MSTR_SRKEY*
ESTAT					SREKEYcnf- *MSTR_SRKEY*	CRLcnf- *MSTR_SRKEY*	SDLIVRcnf- *MSTR_SRKEY*
ASSOC_TER	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*

Note 1 - See SDN.902, sections 10.7 and 10.8 for sequencing information.

	IDLE	SLAVE_ASSOC	WF_EXCrsp	SLAVE_KEYED	WF_CRLrsp	WF_SREKEYrsp	WF_SDLIVRrsp
ASSOC_RESP	*SLAVE_ASSOC*						
NKRQ		EXCind *WF_EXCrsp*					
EXCrsp +			NKRS *SLAVE_KEYED*				
EXCrsp-			A-ABORT *IDLE*				
CRRQ				CRLind *WF_CRLrsp*			
CRLrsp +					CRRS *SLAVE_KEYED*		
CRLrsp-					ESTAT *SLAVE_KEYED*		
SRKRQ				SREKEYind *WF_SREKEYrsp*			
SREKEYrsp +						SRKRS *SLAVE_KEYED*	
SREKEYrsp-						ESTAT *SLAVE_KEYED*	
SRDRQ				SDLIVRind *WF_SDLIVRrsp*			
SDLIVRrsp +							SRDRS *SLAVE_KEYED*
SDLIVRrsp-							ESTAT *SLAVE_KEYED*
ASSOC_TERM	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*	*IDLE*

SDNS
Secure Data
Network System

Key Management Protocol

SDNS Traffic Key Attribute Negotiation

Source: SDNS Protocol and Signaling Working Group
Key Management Sub-Group

Introductory note:

This document specifies the framework of the SDNS Key Management Protocol Security Attribute Negotiation service. It supplements SDN.902 and SDN.903 where traffic key security service attributes negotiation is specified. This document is being circulated for comment and approval. It is subject to change during the development phase of SDNS.

Table of Contents

0. Introduction

This document defines the SDNS key management attribute negotiation service provided by the Key Management Application Service Element (KMASE). The attribute negotiation service provides facilities to support optional security services that are required by Key Management Application Processes (KMAPs) in support of both supplemental access control requirements, and the specific security protocols accommodated by the cooperating SDNS components (i.e., SP2, SP3, SP4, SPN). The attribute negotiation services and Key Protocol Data Unit (KPDU) definitions associated with these services are inherently part of the Key Management Protocol (KMP). This document, therefore, is a direct supplement to the SDNS Key Management Service Definition and Protocol Specifications (SDN.902 and SDN.903, respectively).

1. Scope and field of application

This document applies to those SDNS key management applications which utilize the SDNS attribute negotiation process to apply specific security and communication attributes to a Traffic Encryption Key (TEK) immediately following TEK formation by the KMASE initiator and responder.

2. References

IS 8824 Information Processing Systems - Open Systems Interconnection Specification of Abstract Syntax Notation One (ASN.1)

IS 8825 Information Processing Systems - Open Systems Interconnection Specification of Basic Encoding Rules (BER) for ASN.1

SDN.802 SDNS Access Control Specification

SDN.902 SDNS Key Management Protocol: Definition of Services provided by the Key Management Application Service Element

SDN.903 SDNS Key Management Protocol: Specification of the Protocol for services provided by the Key Management Application Service Element.

3. Definitions

This document uses the definitions contained in the Reference Model for Open Systems Interconnection (IS 7498/1), including the Security Architecture (IS 7498/2). Additionally, the document uses definitions contained in section 3.0 (Definitions) of both SDN.902 and SDN.903.

4. Abbreviations

This document uses abbreviations contained in section 4.0 (Abbreviations) of both SDN.902 and SDN.903.

5. Conventions

The structure of the attribute negotiation parameters are specified using Abstract Syntax Notation One (ASN.1), as defined in ISO 8824. The ASN.1 specified attributes defined in this document are intended to be "exported" to the KMASE ASN.1 specifications, provided in SDN.903, where "AttributeSet" and "Add-Info" are called out as the specific placeholders. [Reference: SDN.903, pp 23]

6. Basic Concept

The basic concept of the attribute negotiation portion of the KMASE is to establish the security and communication service attributes and conditions which are mutually acceptable to both the initiating and responding parties of the Traffic Encryption Key (TEK) generation processes of both KMASE. The KMASE TEK generation process is specified in SDN.902 and SDN.903.

Once a collaborative TEK has been generated by both parties, via the New Key Request (Nkrq) and New Key Response (Nkrs) KPDU exchange process of the KMASE, security service options are negotiated by the initiating KMASE by sending a Security Services Request (Ssrq) KPDU to the receiving KMASE, and the receiving KMASE responding to the Ssrq with a Security Services Response (Ssrs) KPDU. The Ssrq and Ssrs KPDUs are both encrypted using the TEK generated as a successful result of the preceding Nkrq and Nkrs exchange. Therefore, the Ssrs and Ssrq KPDU exchange process not only provides security service attribute negotiation, the exchange also serves as a mutual "liveness check" of the freshly generated TEK; i.e., bilateral authentication of the peer KMASEs.

The Ssrq KPDU sent by the initiating KMASE may contain a set of service options that it supports, called proposed-options, as well as (optionally) additional information, that is domain specific, called add-info. The proposed-options Attribute Set contains the set of those security services and attributes that are standard - i.e., universally understood by all KMASE, and are those specifically supported by the initiating KMASE. Those standard security services and attributes understood by all KMASE, but not supported by the initiating KMASE, are simply not included in the initiating KMASE's proposed-options set. The optional add-info contains domain specific attributes that may be negotiated in addition to the standard negotiated attributes.

In turn, the Ssrs KPDU, sent to the initiating KMASE by the responding KMASE, contains a set of options that it has selected, from the Ssrq KPDU's proposed-options (given the Ssrq KPDU contained these attributes -i.e., it does not have to), called selected-options, and, optionally, add-info data. If, however, the proposed-options or optional add-info attributes sent to the responding KMASE in the Ssrq KPDU are unacceptable, then the responding KMASE may respond with an Estat KPDU, indicating that the Ssrq KPDU or its service options are in error or fail to meet the requirements of the responding KMASE.

If the Ssrs KPDU contains a set of (valid) selected-options, and optionally an add-info set of attributes, and the initiating KMASE accepts, and is in harmony with the selected-options and optionally the add-info set of attributes, then those selected options and optional add-info attributes are used to initialize the selected peer entity security protocols in each component, and are (in most instances) associated directly with the TEK that has been generated.

Subsequent secure communication between the selected peer security protocols (SP) is directly bound to the TEK and regulated by the specific services agreed upon in association with the TEK.

If the response to the Ssrq KPDU is an Estat KPDU, or if the initiator KMASE can not accept the Ssrs KPDU options or optionally add-info attributes, and it then subsequently sends an Estat KPDU to the responder, then the connection is terminated and the TEK generated is discarded by both parties.

7. Attribute Negotiation Service

The KMASE K-TEK-ATTRIBUTES service provides the attribute negotiation activities. The K-TEK-ATTRIBUTES service verifies the cryptographic compatibility with the TEK, negotiates the communication and security attributes associated with the TEK, and exchanges Compromised Key List (CRL) numbers. Additionally, K-TEK-ATTRIBUTES allows a KMAP to associate the attributes of a previous TEK with a new TEK. The K-TEK-ATTRIBUTES parameters are illustrated in Table 1.

Table 1 -- K-TEK-ATTRIBUTES parameters

Parameter Name	Req	ind	rsp	cnf
init-kid			M	M(=)
resp-kid	M	M(=)		
proposed-options	U	U(=)		
selected-options			M	M(=)
add-info	U	U(=)	U	U(=)
old-resp-kid	U	U(=)		
crl-ver	M	M(=)		
result			M	M(=)

Legend:
blank -- not applicable
M -- presence is mandatory
U -- presence is user option
= -- p value is semantically equal to value to its left in the table

7.1 init-kid

This K-TEK-ATTRIBUTES parameter is required on the response primitive. It is passed on the confirm primitive. It contains the identifier the initiator KMAP associates with the TEK. This service parameter is specified in SDN.902, and its KPDU ASN.1 specification described in SDN.903.

7.2 resp-kid

This K-TEK-ATTRIBUTES parameter is required on the request primitive, and passed on the indicate primitive. It contains the identifier the responder KMAP associates with the TEK. This service parameter is specified in SDN.902, and its KPDU ASN.1 specification described in SDN.903.

7.3 proposed-options

This K-TEK-ATTRIBUTES parameter is optional on the request primitive, and, if used, is passed on the indicate primitive. It contains the set of options the initiating KMASE will support. The Attributes Set that constitute the proposed-options, and selected-options are not detailed in SDN.902, and not specified in ASN.1 in SDN.903. Definition of the Attributes Set is the major motivation for this document. The list of Attribute Set options may indicate any or all of the following:

- Confidentiality Service Only Option Indicator
  Choice of:

  - SP2 Option Indicator
    Parameters:
    -- Confidentiality algorithm identifier
    -- Single security level identifier (optional)
    -- SP2 version number

  - SPN* Option Indicator
    Parameters:
    -- Confidentiality algorithm identifier
    -- Single security level identifier (optional)
    -- SPN version number

  - SP4C Option Indicator
    Parameters:
    -- Confidentiality algorithm identifier
    -- Single security level identifier (optional)
    -- SP4C version number

- Both Confidentiality and Integrity Services Option Indicator
  Choice of:

  - SP2 Option Indicator
    Parameters:
    -- Confidentiality and Integrity algorithm identity
    -- Single security level identifier (optional)
    -- Per PDU security label (optional)
    -- SP2 version number

- SP3 A, I, or D Option Indicators
  Parameters:
  -- Confidentiality and Integrity algorithm identity
  -- Single security level identifier (optional)
  -- Per PDU security label (optional)
  -- SP3 (A, I, or D) version number

- SPN* Option Indicator
  Parameters:
  -- Confidentiality and Integrity algorithm identity
  -- Single security level identifier (optional)
  -- Per PDU security label (optional)
  -- SPN version number

- SP4C Option Indicator
  Parameters:
  -- Confidentiality and Integrity algorithm identity
  -- Single security level identifier (optional)
  -- Per PDU security label (optional)
  -- Final sequence number
  -- SP4C version number

● Integrity Only Option Indicator
  Choice of:

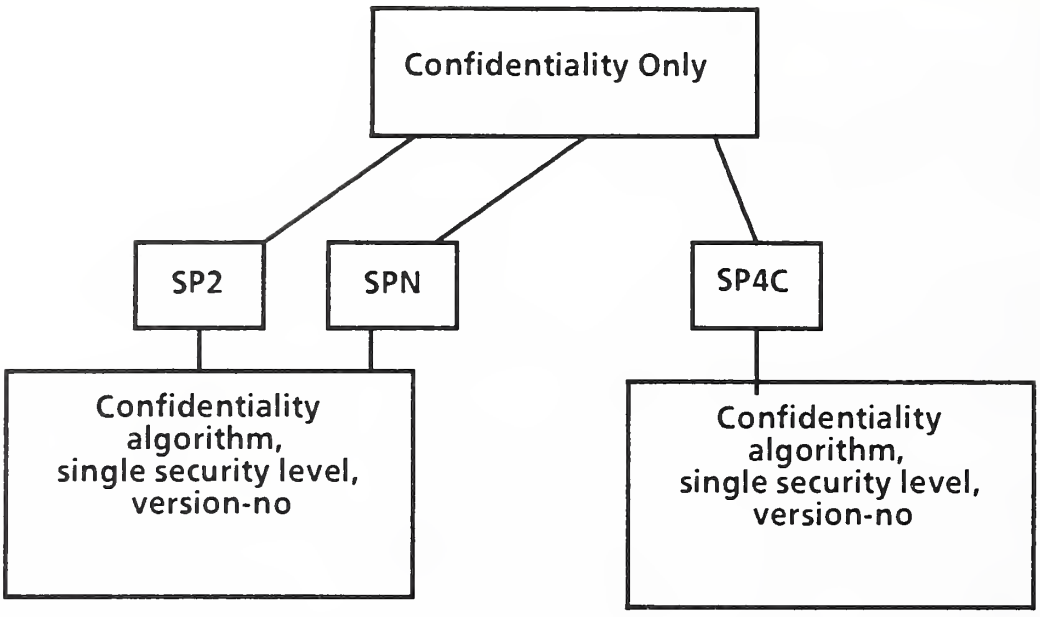
- SP3 A, I, or D Option Indicators
  Parameters:
  -- Integrity algorithm identity
  -- Single security level identifier (optional)
  -- Per PDU security label (optional)
  -- SP3 A, I, or D version number

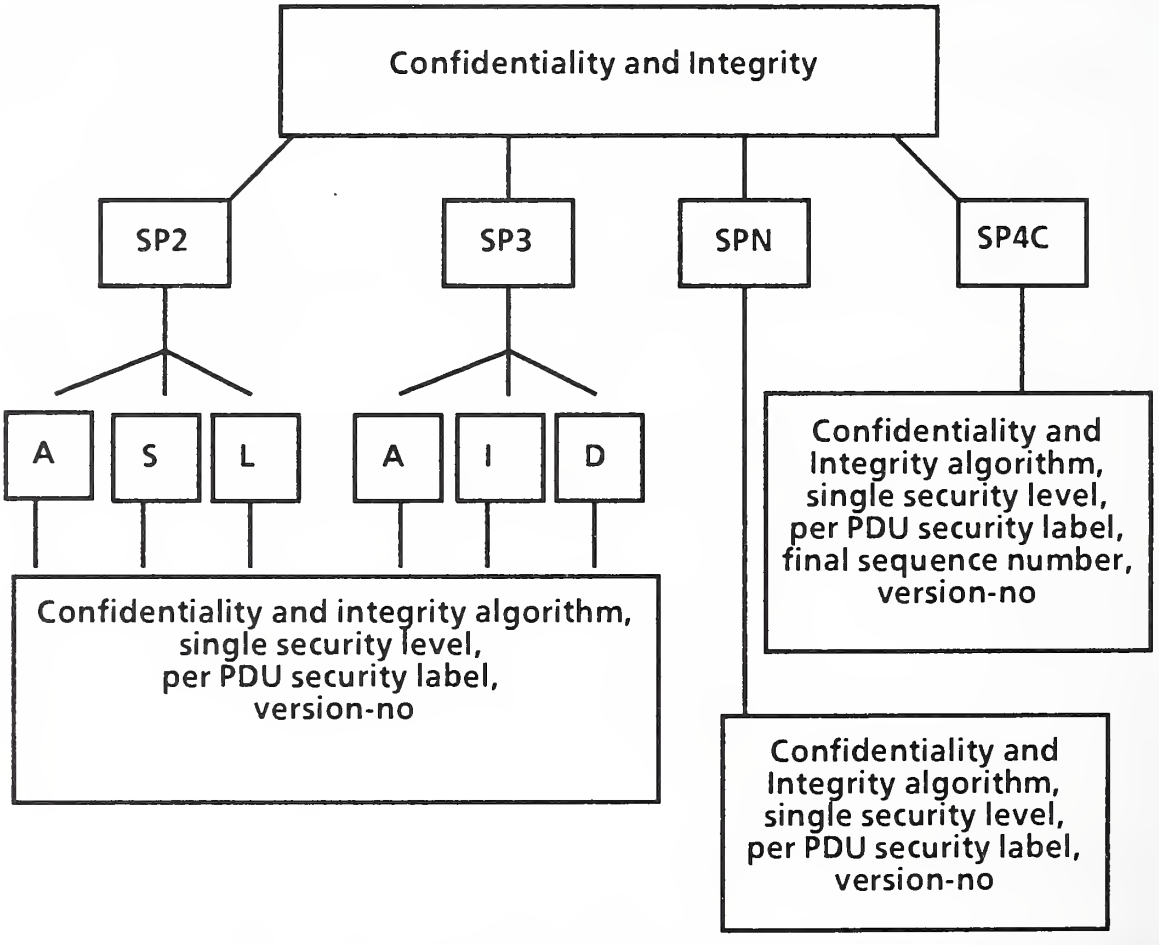
- SPN Option Indicator
  Parameters:
  -- Integrity algorithm identity
  -- Single security level identifier (optional)
  -- Per PDU security label, (optional)
  -- SPN version number

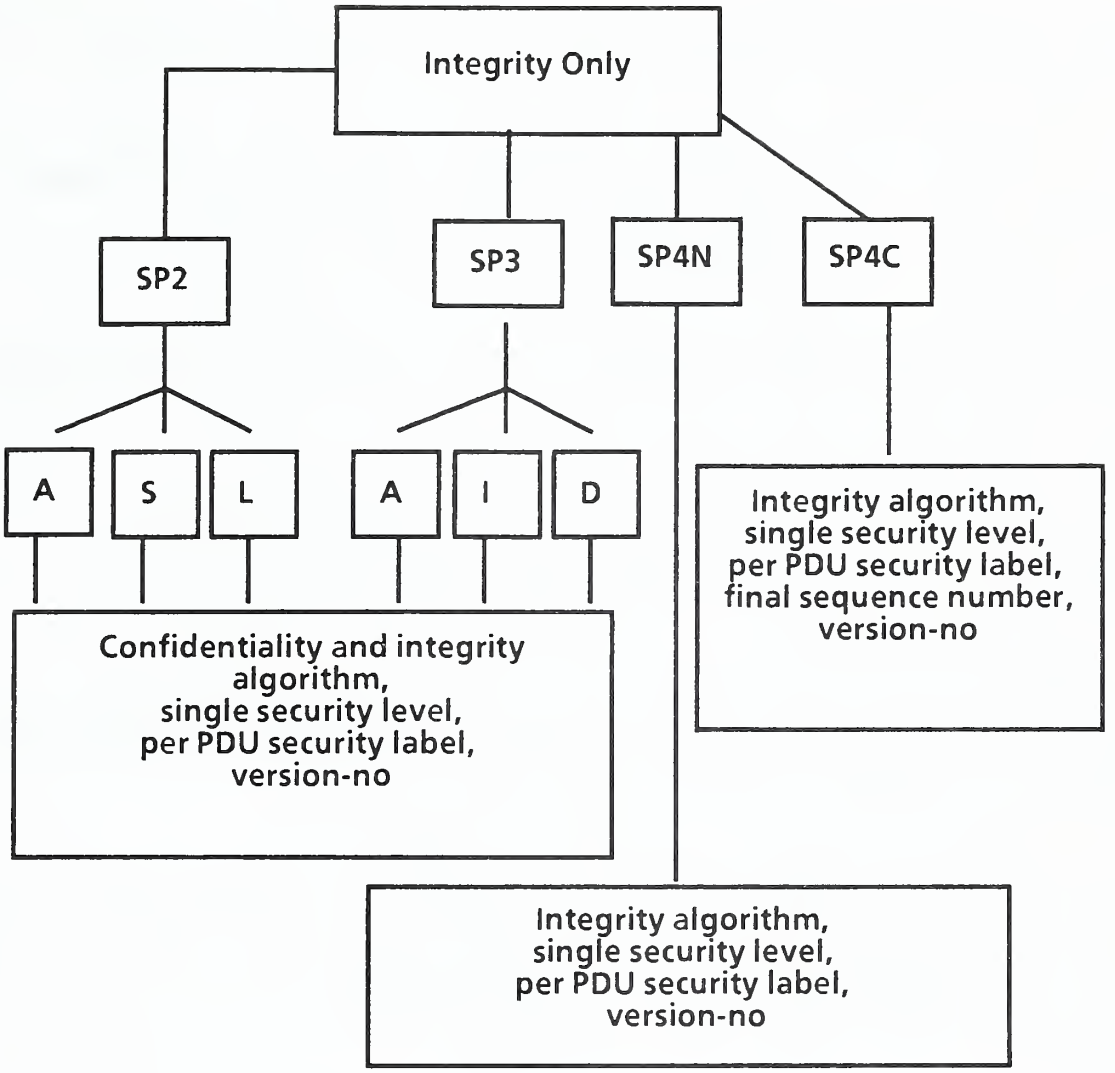
- SP4C Option Indicator
  Parameters:
  -- Integrity algorithm identity
  -- Single security level identifier (optional)
  -- Per PDU security label
  -- Final sequence number
  -- SP4C version number

* SPN is equivalent to SP4E and SP3N

Hence, the list of proposed-options has a tree structure, where some options are dependent on other options. These dependencies are illustrated in the following trees.


Confidentiality Only
SP2
SPN
SP4C
Confidentiality algorithm, single security level, version-no
Confidentiality algorithm, single security level, version-no


Confidentiality and Integrity
SP2
SP3
SPN
SP4C
A
S
L
A
I
D
Confidentiality and integrity algorithm, single security level, per PDU security label, version-no
Confidentiality and Integrity algorithm, single security level, per PDU security label, final sequence number, version-no
Confidentiality and Integrity algorithm, single security level, per PDU security label, version-no



At the first tree level, the confidentiality/integrity services to be used are requested. The allowable options are:

confidentiality only
confidentiality and integrity
integrity

The next tree level specifies the security protocol and its mode to be used with the key. The allowable values and the mnemonics are:

SP2 - SP2 protocol
SP3 - SP3 protocol using either address or header protection
SPN - Intersection of SP3N and SP4E
SP4C - Connection-oriented SP4

When SP3 is the specified protocol, the protected header format is chosen at the next level of the tree; i.e., the mode. The allowable values and mnemonics are:

A - SP3 mode which includes source and destination NSAP addresses in a protected header
I - SP3 mode which includes the ISO ConnectionLess Network Protocol (CLNP) and a protected CLNP header
D - SP3 mode which includes the DoD Internet Protocol (IP) and a protected IP header.

The tree leaves contain a list of the remaining options. Illustrations, above, highlight which options are allowable for the security services and protocol modes selected. The remainder of this section describes each of the remaining negotiation options.

7.3.1 Confidentiality Algorithm

The confidentiality algorithm option is used for negotiating which symmetric algorithm(s) to use for the data confidentiality service of the security protocol and mode chosen. This algorithm is different than that used to form the TEK.

7.3.2 Integrity Algorithm

The integrity algorithm option is used for negotiating use of integrity algorithms, other than that used to form the TEK. This option includes the negotiation of the icv length. The icv length can be any of the following:

icv-len2 Integrity with a 2 byte icv length,
icv-len4 Integrity with a 4 byte icv length,
icv-len6 Integrity with a 6 byte icv length,
icv-len8 Integrity with a 8 byte icv length,
icv-len10 Integrity with a 10 byte icv length, or
icv-len12 Integrity with a 12 byte icv length.

7.3.3 Confidentiality and Integrity Algorithm

The confidentiality and integrity algorithm option is used for negotiating use of algorithms other than that used to form the TEK for data security. This option includes the negotiation of the algorithm mode and the icv length, as defined above.

7.3.4 Per KPDU Security Label and Format

This option is used to negotiate the existence and format of an explicit security label on every PDU exchanged while using the TEK identified. The choices for this option and the mnemonics are:

ppl abs Per PDU security label absent
ppl DoD DoD security label per PDU
ppl xxx xxx security label per PDU

NOTE: xxx is yet to be defined.

7.3.5 Single Security Level

This option is used to negotiate a single security level for the TEK identified.

7.3.6 Final Sequence Numbers The final sequence numbers option is used to negotiate connection truncation protection. The choices and mnemonics are:

true - Final sequence numbers are exchanged at the closing of transport connections when using this TEK,
false - Final sequence numbers are not exchanged at the closing of the transport connections when using this TEK.

7.3.7 SP Version Numbers

The SP version numbers are used to negotiate the use of a particular SP version; e.g., there may be n versions of SP3D.

7.4 selected options

This parameter is required on the response primitive. It is passed on the confirm primitive. It contains the set of options the responder has selected from either the proposed-options attributes the initiator has sent in the Ssrq KPDU (if included in the Ssrq KPDU), or the responder has exclusively chosen (if the Ssrq KPDU did not contain proposed-options).

7.5 add-info

This optional parameter may be included on the request and response primitives. It is passed on the indication and confirm primitives. It indicates that additional information is required (and supplied) to further accommodate local domain system management necessities, such as access control and addressing information to be included as (associated) attributes of the traffic key. The additional information may include:

- auxiliary vector information,
- signed lists of networks the SDNS component serves.

7.5.1 Auxillary Vector Information

The use of auxillary vectors (AV) is a local domain matter. The information contained in an AV is used to augment that which is contained in a user's "primary certificate". The information may be added identification, authentication, or access control data. The AV may be signed by the SDNS KMS or an "Auxillary Vector Management System" (AVMS). The reader is referred to the SDNS Access Control Specification, SDN.802 for further information.

7.5.2 Add-Info For Signed List of Network Addresses

An SP3 entity located at an intermediate system may send to its peer entity a list of the end systems reachable via the entity.

7.6 old-resp-kid

This parameter may be included on the request primitive. It is passed on the indication primitive. It contains an indicator that the KMASE wishes to associate the attributes of another TEK with the key specified by the init-kid and resp-kid parameters. Only the initiator of the cryptographic association specifies this parameter. If specified, the initiator omits the proposed options. This service parameter is fully specified in SDN.902 and described in ASN.1 form in SDN.903.

7.7 crl-version

This parameter must be included on the request and response primitives. It is passed on the indication and confirm primitives. This parameter contains the unsigned version number of the current CRL used by the KMAP. This service parameter is fully specified in SDN.902 and described in ASN.1 form in SDN.903.

7.8 result

This parameter is required on the response primitive. It is passed on the confirm primitive. It indicates the success or failure of the operation. This service parameter is fully specified in SDN.902 and described in ASN.1 form in SDN.903.

8. Attribute Negotiation Flow

Security options are negotiated by means of one KMASE (the initiator) transmitting a Ssrq KPDU to the other KMASE (the responder), which replies with a Ssrs KPDU. The initiator specifies a set of preferred and alternative security options in the Ssrq KPDU. The responder specifies the selected set of security options in the Ssrs KPDU.

Using the parameters from the K-TEK-ATTRIBUTES request primitive, the initiator KMASE prepares a Ssrq KPDU for the responder KMASE. The Ssrq KPDU contains a list of preferred and alternative options. Default values are assumed for options which are omitted from the preferred and alternative sets. After forming the Ssrq KPDU, the KMASE computes an icv and then encrypts the KPDU. The cryptographic key used for
both the icv and the encryption is the previously formed traffic key. The KMASE sends the Ssrq KPDU to the peer KMASE.

On receipt of a Ssrq KPDU, the responder KMASE decrypts the preferred and alternative options and verifies the icv. If either of these operations fails, the KMASE sends a STAT KPDU to the peer KMASE indicating the cause of the failure. After decrypting and verifying the icv of the KPDU, the KMASE issues a K-TEK-ATTRIBUTES indication primitive to the responder. The KMASE now expects a K-TEK-ATTRIBUTES response primitive from the responder containing the selected options, and, optionally, add-info data.

Using the parameters from the K-TEK-ATTRIBUTES response primitive, the responder KMASE prepares a Ssrs KPDU. After forming the Ssrs KPDU, the KMASE computes an icv and then encrypts the KPDU. The cryptographic key used for both the icv and the encryption is the previously formed traffic key. The Ssrs KPDU is sent to the initiator KMASE.

If the parameters of the K-TEK-ATTRIBUTES response primitive indicates a failure, the KMASE sends an Estat KPDU instead of a Ssrs KPDU.

On receipt of a Ssrs KPDU the initiator KMASE decrypts the KPDU and verifies the icv. The KMASE issues the K-TEK-ATTRIBUTES confirm primitive to the KMASE user application indicating the success or failure of the operation as well as the selected options, and, optionally, any add-info data.

If the initiator KMASE receives an Estat KPDU while awaiting the Ssrs KPDU, the KMASE issues a K-TEK-ATTRIBUTES confirm response to the KMASE user application indicating the failure of the option negotiation process.

9. Exportable AttributeSet Abstract Syntax Specifications

The following Ssrq and Ssrs KPDU AttributeSet definitions, expressed in ASN.1 form, should be "imported" to the KMASE ASN.1 specifications defined in SDN.903, where the placeholder "AttributeSet" is established.

```
AttributeSet Definitions:: = BEGIN
-- EXPORT To SDN.903 ASN.1 Specs @ AttributeSet Placeholder

Options :: = CHOICE{          -- proposed/selected-options
        [0] SpnWithConf,
        [1] SpnWithInteg,
        [2] SpnWithConfAndInteg,
        [3] Sp4cWithConf,
        [4] Sp4cWithInteg,
        [5] Sp4cWithConfAndInteg,
        [6] Sp3WithInteg,
        [7] Sp3WithConfAndInteg,
        [8] Sp2WithConf,
        [9] Sp2WithConfAndInteg
        }

SpnWithConf :: = SET{
        version-no          INTEGER,
        confAlgorithm       AlgorithmIdentifier,
        singleSecLevel      SecurityLevel OPTIONAL
        }

SpnWithInteg :: = SET{
        version-no          INTEGER,
        integAlgorithm      AlgorithmIdentifier,
        singleSecLevel      SecurityLevel OPTIONAL,
        perPDUSecLabel      [0] INTEGER OPTIONAL{
                            ppl-abs(0),          -- Per PDU security label absent
                            ppl-Dod(1),          -- DoD security label per PDU
                            ppl-xxx(2)           -- xxx security label per PDU
                            },
        icv-length          IMPLICIT Icv-Length
        }

SpnWithConfAndInteg :: = SET{
        version-no                  INTEGER,
        confAndIntegAlgorithm       AlgorithmIdentifier,
        singleSecLevel              SecurityLevel OPTIONAL,
        perPDUSecLabel              [0] INTEGER OPTIONAL{
                                    ppl-abs(0),   -- Per PDU security label absent
                                    ppl-Dod(1),   -- DoD security label per PDU
                                    ppl-xxx(2)    -- xxx security label per PDU
                                    },
        icv-length                  IMPLICIT Icv-Length
        }
```

```
Sp4cWithInteg :: = SET{
        version-no          INTEGER,
        integAlgorithm      AlgorithmIdentifier,
        singleSecLevel      SecurityLevel OPTIONAL,
        perPDUSecLabel      [0] INTEGER OPTIONAL{
                            ppl-abs(0),   -- Per PDU security label absent
                            ppl-Dod(1),   -- DoD security label per PDU
                            ppl-xxx(2)    -- xxx security label per PDU
                            },
        finalSeqNumber      BOOLEAN,
        icv-length          IMPLICIT Icv-Length
        }

Sp4cWithConf :: = SET{
        version-no          INTEGER,
        confAlgorithm       AlgorithmIdentifier,
        singleSecLevel      SecurityLevel OPTIONAL,
        perPDUSecLabel      [0] INTEGER OPTIONAL{
                            ppl-abs(0),   -- Per PDU security label absent
                            ppl-Dod(1),   -- DoD security label per PDU
                            ppl-xxx(2)    -- xxx security label per PDU
                            },
        finalSeqNumber      BOOLEAN
        }

Sp4cWithConfAndInteg :: = SET{
        version-no                  INTEGER,
        confAndintegAlgorithm       AlgorithmIdentifier,
        singleSecLevel              SecurityLevel OPTIONAL,
        perPDUSecLabel              [0] INTEGER OPTIONAL{
                                    ppl-abs(0),   -- Per PDU security label absent
                                    ppl-Dod(1),   -- DoD security label per PDU
                                    ppl-xxx(2)    -- xxx security label per PDU
                                    },
        finalSeqNumber              BOOLEAN,
        icv-length                  IMPLICIT Icv-Length
        }
```

10. Exportable Additional Information Abstract Syntax Specifications

The following Ssrq and Ssrs KPDU AddInfo definitions, expressed in ASN.1 form, should be "imported" to the KMASE ASN.1 specifications defined in SDN.903, where the placeholder "AddInfo" is established.

```
AddInfo Definition:: = BEGIN

 -- Export to SDN.903 @ Add-Info placeholder
          --Add-Info Parameter is Optional in Ssrs and Ssrq KPDUs
          --If used, each possible option is itself optional, as follows:

 Add-Info :: = SEQUENCE{
          aux-vector                  [0] IMPLICIT AuxillaryVector OPTIONAL
                                      (tbd),
          signed-EndSystem-list       [1] IMPLICIT SignedEndSystemList
                                      OPTIONAL (tbd)

          }

END  -- of AddInfo
```

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 90- 4262

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
MARCH 1990

4. TITLE AND SUBTITLE

5. AUTHOR(S)

Charles Dinkel - Editor

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

National Security Agency - Information Security Applications Group
9800 Savage Road, SDNS SPO (C23)
Fort Meade, MD 20755-6000

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The Secure Data Network System project, known as SDNS, implements computer to computer communications security for distributed applications. The internationally accepted Open Systems Interconnection (OSI) computer networking architecture provides the framework for SDNS. SDNS uses the layering principles of OSI to implement secure data transfers between computer nodes of local area and wide area networks. This publication includes four key management documents developed by the National Security Agency (NSA) as output from the SDNS project. SDN.601 supplies a profile for the implementation of SDNS Key Management services in Open Systems. The definition of services provided by the Key Management Application Service Element (KMASE) is provided by SDN.902. The protocol specified in SDN.903 describes the KMASE services provided to the Key Management Application Process (KMAP) to support applications in a distributed open system environment. The framework of the SDNS security attribute negotiation service is described in SDN.906.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

computer security; key management; key management application process; key management protocols; SDNS; security protocols; traffic encryption; traffic key ;network security

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
98

15. PRICE
A05

ELECTRONIC FORM


AMERICAN BINDERY EAST
1993