

# **Working Implementation Agreements for Open Systems Interconnection Protocols: Part 12 - OS Security**

Output from the March 1994 Open Systems Environment  
Implementors' Workshop (OIW)

Acting SIG Chair: **Richard Harris, The Boeing Company**  
SIG Editor: **Dr. Mohammad Mirhakkak, MITRE**

## **Foreword**

This part of the Working Implementation Agreements was prepared by the Security Special Interest Group (SECSIG) of the Open Systems Environment Implementors' Workshop (OIW). See Part 1 - Workshop Policies and Procedures in the "Draft Working Implementation Agreements Document" for the workshop charter.

Text in this part has been approved by the Plenary of the above-mentioned Workshop. This part replaces the previously existing chapter on this subject. This part has been reformatted from the previous release.

Future changes and additions to this version of these Implementor Agreements will be published as a new part. Deleted and replaced text will be shown as . New and replacement text will be shown as shaded.

## Table of Contents

<b>Part 12 - Security</b>	.....	1
<b>0</b>	<b>Introduction</b> .....	1
<b>1</b>	<b>Scope</b> .....	1
<b>2</b>	<b>Normative References</b> .....	1
<b>3</b>	<b>Definitions</b> .....	1
<b>4</b>	<b>Symbols and Abbreviations</b> .....	1
<b>5</b>	<b>Application Architectures</b> .....	2
	5.1 Introduction .....	2
	5.2 Application Environments .....	2
	5.3 Security Classes .....	2
	5.4 Guidelines for OIW Application Profile Development .....	2
	5.5 Placement of Security Services .....	2
	5.6 Selection of Mechanisms .....	3
<b>6</b>	<b>Key Management</b> .....	4
<b>7</b>	<b>Security Algorithms</b> .....	4
	7.1 Message Digests .....	4
	7.2 Reversible Public Key Algorithms .....	5
	7.3 Irreversible Public Key Algorithms .....	5
	7.4 Key Exchange .....	5
	7.5 Signature Algorithms .....	5
	7.5.1 RSA Signature With MD2 .....	5
	7.5.2 RSA Signature With MD5 .....	5
	7.6 Symmetric Encryption Algorithms .....	6
	7.6.1 Data Encryption Standard .....	6
	7.6.1.1 Padding Rules for DES .....	6
	7.6.1.1.1 RFC 1423 Mechanism .....	6
	7.7 ASN.1 .....	6
	7.8 Security Attributes .....	6
	7.8.1 Liability Limitation .....	6
	7.8.2 Binding Information .....	7
	7.8.3 Certificate Purpose .....	9
	7.8.4 Signature Purpose .....	9
	7.8.5 Role Name .....	9
	7.8.6 Agent Name .....	10
	7.8.7 Document Types .....	10
	7.8.8 Trusted Third Party .....	10

7.8.9	Cosignature Requirements	11
7.8.10	Relative Identity	11
7.8.11	Trust Specification	12
7.8.12	Transaction Limit	13
7.8.13	Transaction Type	13
7.8.14	Time Of Day	13
7.8.15	Location	14
7.8.16	Authorized Signatory	14
7.8.17	Pre-approved Counter Party	15
7.8.18	Delegation Controls	15
<b>8</b>	<b>Lower Layers Security</b>	<b>15</b>
<b>9</b>	<b>Upper Layers Security</b>	<b>16</b>
9.1	Security Mechanisms	16
9.1.1	Peer Entity Authentication	16
9.1.1.1	Simple-Strong Authentication	16
9.1.1.2	External Authentication Mechanisms	17
9.1.1.2.1	Kerberos Version 5	17
9.1.1.2.2	Kerberos Version 4	17
9.1.2	Integrity/Data Origin Authentication Transformation	17
<b>10</b>	<b>Message Handling System (MHS) Security</b>	<b>17</b>
<b>11</b>	<b>Directory Services Security</b>	<b>18</b>
<b>12</b>	<b>Network Management Security</b>	<b>18</b>
12.1	Threats	18
12.2	Security Services	18
12.3	Security Mechanisms	18
12.3.1	Peer-Entity Authentication	18
12.3.2	Connectionless Integrity	18
12.3.3	Data Origination Authentication	18
12.3.4	Connectionless Confidentiality	19
<b>Annex A (normative)</b>		
<b>ISPICS Requirements List</b>		<b>20</b>
<b>Annex B (normative)</b>		
<b>Errata</b>		<b>21</b>
<b>Annex C (normative)</b>		
<b>Security Labels</b>		<b>23</b>
<b>Annex D (normative)</b>		

- Security Algorithms and Attributes** ..... 24
- Annex E** (normative)
- References for Security Algorithms** ..... 28
- Annex F** (informative)
- Bibliography** ..... 29
- Annex G** (normative)
- EI Gamal** ..... 30
- Annex H** (informative)
- STATUS** ..... 31
- Annex I** (informative)
- Security-SIG Management Plan** ..... 32
- Annex J** (informative)
- Key Management** ..... 33
  - J.1 Definition of Key Management ..... 33
  - J.2 Tutorial on Key Management ..... 33
    - J.2.1 Requirements of Key Management ..... 33
    - J.2.2 Key Administration ..... 34
      - J.2.2.1 Generation ..... 34
      - J.2.2.2 Validation ..... 34
      - J.2.2.3 Expiration ..... 34
      - J.2.2.4 Audit ..... 35
      - J.2.2.5 Authorization/Authentication ..... 35
    - J.2.3 Approaches to Key Distribution ..... 35
      - J.2.3.1 Symmetric ..... 35
        - J.2.3.1.1 Certificate ..... 35
        - J.2.3.1.2 Symmetric Generation ..... 35
        - J.2.3.1.3 Centralized ..... 35
        - J.2.3.1.4 External ..... 36
      - J.2.3.2 Asymmetric ..... 36
        - J.2.3.2.1 Certificate ..... 36
        - J.2.3.2.2 Centralized ..... 36
        - J.2.3.2.3 External ..... 36
  - J.3 Key Management Architectures ..... 36
    - J.3.1 Existing Systems ..... 36
      - J.3.1.1 SDNS ..... 36
      - J.3.1.2 SILS ..... 36

**PART 12 - Security**

**March 1994 (Working)**

	J.3.1.3	ANSI X9.17	36
	J.3.1.4	Kerberos	36
	J.3.2	OSI	37
J.4		Current Issues	37
J.5		Related Organizations	37
	J.5.1	ANSI X.9	37
	J.5.2	SC21	37
	J.5.3	SC27	37
	J.5.4	IEEE 802.10	37
J.6		References	37
<b>Annex K (informative)</b>			
		<b>Base Environment Threats</b>	38

## **List of Figures**

**List of Tables**

Table 2 - Base Security Services/Mechanisms . . . . .	3
Table 3 - Distributed Transactions Security Services/Mechanisms . . . . .	3
Table 4 - WIA Part 12 Changes . . . . .	21
Table 5 - ISO Status . . . . .	31
Table 6 - Management Plan . . . . .	32
Table 7 - Threats to Services . . . . .	38



## **Part 12 - Security**

### **0 Introduction**

Refer to clause 0 of the Stable Implementation Agreements.

### **1 Scope**

### **2 Normative References**

Refer to clause 2 of the Stable Implementation Agreements.

### **3 Definitions**

**Editor's Note** - This clause will contain all unique terms used in this part, to be determined.

Refer to ISO 7498/2 for definitions of security relevant terms. This base standard contains detailed descriptions of accepted security terms. Refer to ISO TR-10000 for general ISO definitions used in this part. The following security terms are not defined in ISO 7498/2:

- a) Authentication;
- b) Mechanism;
- c) Profile.

**Editor's Note** - The above terms will be defined as a work item.

### **4 Symbols and Abbreviations**

## 5 Application Architectures

(See Stable Document).

### 5.1 Introduction

(See Stable Document).

### 5.2 Application Environments

(See Stable Document).

### 5.3 Security Classes

(See Stable Document).

### 5.4 Guidelines for OIW Application Profile Development

### 5.5 Placement of Security Services

The following guidelines are provided for other OIW SIGs to use in the preliminary development of their own application specific security profile. It is intended that final completion of the security profiles should be done in a joint manner between the Security SIG and the other OIW SIGs.

**Editor's Note** - Item a of the following paragraph will be considered for deletion at the next Security SIG meeting.

The steps in the guidelines are as follows:

- a) Start with the base Profile (5.3.1);
- b) Perform application specific threat analysis. Map the result of this analysis to security services;
- c) Map security services onto application specific security services (e.g., the threats identified for MHS in X.402 are mapped against MHS specific security services);
- d) Map security services to mechanisms that will be used to provide the services;
- e) Describe the security classes and map them to the defined functional groups.

**Editor's Note** - Steps f and beyond are TBD. It will require further discussion to decide exactly how the application specific security profile is finally determined, how those profiles can be specified (security context, object identifier?) and how we will specify the mechanisms of choice for the implementation of the profile.

Further discussion is needed on Security Policy. This is a priority work item.

### 5.6 Selection of Mechanisms

Table 2 defines the security mechanisms to use in providing security services to protect against the defined threats.

**Table 2 - Base Security Services/Mechanisms**

SECURITY SERVICES	SECURITY MECHANISMS								
	ENCY- PHER	DIG. SIG.	ACC. CTRL.	DATA INTG.	AUTH EX.	TRFF PAD.	RT. CTRL	NOT- IZE.	AUD- IT
AUTHENTICATION	X	X			X				1
ACCESS CONTROL			X		X				1
NON-REPUDIATION		X		X				X	1
DATA INTEGRITY	X			X					1
CONFIDENTIALITY	X					X	X		1

**NOTE -** The security mechanisms of auditing can be used to provide added security to any security service.

Table 3 defines the security mechanisms to use in providing security services to protect against the defined threats.

**Table 3 - Distributed Transactions Security Services/Mechanisms**

SECURITY SERVICES	SECURITY MECHANISMS								
	ENCY- PHER	DIG. SIG.	ACC. CTRL.	DATA INTG.	AUTH EX.	TRFF PAD.	RT. CTRL	NOT- IZE.	AUD- IT
AUTH. DIALOG					X				
ASSOCIATION	X	X			X				1
ACCESS DIALOG			X						
CONTROL ASSOCIATION			X						1
NON-REPUDIATION		X						X	1
DATA INTEGRITY	X	X		X					1
CONF. CL	X						X		1
TRAFFIC FLOW	X					X	X		1

**NOTE** - The security mechanisms of auditing can be used to provide added security to any security service.

## 6 Key Management

Refer to Part 12, clause 6 of the Stable Implementation Agreements.

## 7 Security Algorithms

(See the Stable Document).

### 7.1 Message Digests

(See Stable Document).

## **7.2 Reversible Public Key Algorithms**

(See Stable Document).

## **7.3 Irreversible Public Key Algorithms**

(See the Stable Document).

## **7.4 Key Exchange**

(See the Stable Document).

## **7.5 Signature Algorithms**

(See the Stable Document).

### **7.5.1 RSA Signature With MD2**

This algorithm uses the RSA Signature algorithm to sign the digest produced by the MD2 hash algorithm. Its object identifier is

```
md2WithRSASignature
    PARAMETER NULL
 ::= { algorithm 24 }
```

### **7.5.2 RSA Signature With MD5**

This algorithm uses the RSA Signature algorithm to sign the digest produced by the MD5 hash algorithm. Its object identifier is

```
md5WithRSASignature
    PARAMETER NULL
 ::= { algorithm 25 }
```

## 7.6 Symmetric Encryption Algorithms

(See the Stable Document).

### 7.6.1 Data Encryption Standard

#### 7.6.1.1 Padding Rules for DES

**Editor's Note** - This clause will be moved to SIA in June 1994. It will be placed between DES-EDE, clause 7.6.1.6, and RC2-CBC, clause 7.6.2

This section describes some useful padding mechanisms for DES in its various modes of operation, for the case where the input is not a multiple of 8 bytes in length.

##### 7.6.1.1.1 RFC 1423 Mechanism

The following padding mechanism from [w] should be used with DES-CBC if the data to be encrypted is octet aligned, unless the security policy dictates otherwise:

The input to the DES CBC encryption process must be padded to a multiple of 8 octet, in the following manner. Let  $n$  be the length in octets of the input. Pad the input by appending  $8 - (n \bmod 8)$  octet to the end of the message, each having the value  $8 - (n \bmod 8)$ , the number of octets being added. In hexadecimal, the possible paddings are: 01, 0202, 030303, 04040404, 0505050505, 060606060606, 07070707070707, and 0808080808080808. All input is padded with 1 to 8 octets to produce a multiple of 8 octets in length. The padding can be removed unambiguously after decryption.

## 7.7 ASN.1

(See the Stable Document).

## 7.8 Security Attributes

This section identifies some useful security attributes which are defined in ANSI X9.30 Part 3, "Certificate Management for DSA."

### 7.8.1 Liability Limitation

```
LiabilityLimitation ::= CHOICE {
    no-liability      [0]    NULL,
    full-liability    [1]    NULL,
    monetary-limit    [2]    MonetaryValue }
MonetaryValue ::= SEQUENCE {
    currency          [0]    PrintableString (SIZE 3),    -- per ISO 4217
    amount            [1]    INTEGER}
```

This attribute defines the limits of a CA's liability in the event of key compromise, etc.

```
liability-limitation ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX LiabilityLimitation
    SINGLE VALUE
::= id-liability-limitation
```

## **7.8.2 Binding Information**

```

BindingInformation ::= SEQUENCE {
    methodOfDelivery      [0] DeliveryMethod,
    methodOfIdentification [1] IdentificationMethod,
    entityType             [2] EntityType }

DeliveryMethod ::= ENUMERATED {
    not-presented-in-person (0),
    presented-in-person (1),
    presented-by-authorized-agent (2),
    split-knowledge (3),
    other (4) }

MethodOfIdentification ::= SEQUENCE {
    IdentificationMethod,
    SEQUENCE OF IdentificationDocument }

IdentificationMethod ::= ENUMERATED {
    reasonable-commercial-practices (0),
    verified-by-trusted-third-party (1),
    dual-control (2),
    other (3) }

IdentificationDocument{ID-Documents} ::= SEQUENCE {
    documentType      ID-DOC.&id({ID-Documents}),
    documentID        ID-DOC.&Type({ID-Documents},
                                {@documentType}) }

ID-DOC ::= TYPE-IDENTIFIER
drivers-license ID-DOC ::= { PrintableString IDENTIFIED BY
    { id-doc-drivers-license } }

passport ID-DOC ::= { PrintableString IDENTIFIED BY
    { id-doc-passport } }

alien-registration ID-DOC ::= { PrintableString IDENTIFIED BY
    { id-doc-alien-registration } }

birth-certificate ID-DOC ::= { PrintableString IDENTIFIED BY
    { id-doc-birth-certificate } }

EntityType ::= ENUMERATED {
    individual (0),
    corporation (1),
    government (2),
    other (3) }

```

This attribute indicates the criteria used to bind the credentials to the identity of the entity being certified.

```

binding-information ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX BindingInformation
    SINGLE VALUE
    ::= id-binding-information

```



### 7.8.3 Certificate Purpose

```
CertificatePurpose ::= ENUMERATED {
    any (0),
    encipherment (1), -- key transport
    signature (2) }
```

This attribute indicates what functions the public key contained in the certificate may be used for.

```
certificate-purpose ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX CertificatePurpose
    SINGLE VALUE
 ::= id-certificate-purpose
```

### 7.8.4 Signature Purpose

The Signature Purpose attribute indicates the purpose of the originator in applying a signature to a document (e.g., authorizing the document, witnessing another signer's signature, etc.).

```
signaturePurpose ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX OBJECT IDENTIFIER
 ::= { attribute 15 }
```

Values for the attribute will be registered at a later date.

### 7.8.5 Role Name

The Role Name attribute type specifies the designated FUNCTION of an object (generally a human) WITHIN the organization.

Example:

Role="Program Manager, X.500 Project"

Role="Principal Investigator, X.520 Anomalies and Defects"

```
roleName ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
    (SIZE (1..ub-common-name))
 ::= { attribute 16 }
```

### 7.8.6 Agent Name

The Agent Name attribute specifies the FUNCTION of an object whose actions have consequences OUTSIDE of the organization, and are authorized in some sense to speak for, commit, or bind the organization.

Example:

```
Agent="Chief Financial Officer"
Agent="Purchasing Agent"
Agent="Corporate Spokesperson"
```

```
agentName ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
  (SIZE (1..ub-common-name))
 ::= { attribute 17 }
```

### 7.8.7 Document Types

Document type OIDs are needed for the binding information attributes. Associated data types for the OIDs can be found in Appendix E of ANSI X9.30 Part 3.

```
doc-type ID ::= { iso(1) identified-organization(3) oiw(14)
secsig(3) doc-type(6) }
id-doc-drivers-license ID ::= { doc-type 1 }
id-doc-passport ID ::= { doc-type 2 }
id-doc-alien-registration ID ::= { doc-type 3 }
```

### 7.8.8 Trusted Third Party

```
TrustedThirdParty ::= SEQUENCE {
  partyType      ThirdPartyType,
  thirdParty     Name } -- or SubjectName?

ThirdPartyType ::= ENUMERATED { notary(0), witness(1),
  guardian(2), legal-custodian(3) }
```

This attribute is used when the identification process uses such an entity, e.g. to present identification documents. This allows a complete trail to be constructed from the top-level CA through all involved parties to the certificate subject.

```
trusted-third-party ATTRIBUTE
  WITH ATTRIBUTE-SYNTAX Name
 ::= id-trusted-third-party
```

### 7.8.9 Cosignature Requirements

This attribute defines any additional signatures required on a certificated signed by the CA to which the attribute refers. It is used to enforce the multiple signature requirement for high-risk applications.

```

CosignatureRequirements ::= SEQUENCE {
    quorum      [0]  INTEGER { allMembers(0) },
    members    [1]  SEQUENCE OF CosignerEntry }

CosignerEntry ::= CHOICE {
    single      [0]  Cosigner,
    list       [1]  CosignatureRequirements }

Cosigner ::= SEQUENCE {
    cosigner   CosignerID,
    weight     INTEGER DEFAULT 1 }

CosignerID ::= CHOICE {
    name       [0]  CosignerName,
    issuerSerial [1]  IssuerSerial }

CosignerName ::= SEQUENCE {
    name       Name,
    uniqueID   BIT STRING OPTIONAL }

IssuerSerial ::= SEQUENCE {
    issuer     Name,
    serial     CertificateSerialNumber }

cosignature-requirements ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX CosignatureRequirements
    SINGLE VALUE
 ::= id-cosignature-requirements

```

### 7.8.10 Relative Identity

A CA may wish to certify only a portion of a name of an individual in a normal business setting. E.g., the CA may wish to disclaim liability for correctness of an individual's personal name, since the user's signature is binding on the organization in any event. In such a case, the CA would only vouch for the correctness of the organizational part of the user's distinguished name.

```

RelativeIdentity ::= INTEGER
-- number of certified RDNs in the DN

relative-identity ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX RelativeIdentity
    SINGLE VALUE
 ::= id-relative-identity

```

### 7.8.11 Trust Specification

One can specify the trust in a given CA with the following ASN.1 type.

```
Trust ::= SIGNED SET {      -- signed by the user
    cross      [0] CrossCertify OPTIONAL,
    users      [1] Users OPTIONAL }
```

The two components answer the questions:

- a) Which CAs may the CA cross certify?, and
- b) Which users may the CA certify?

```
CrossCertify ::= SEQUENCE OF CrossCertifyEntry
```

```
CrossCertifyEntry ::= SEQUENCE {
    crossSpec      CrossSpecification,
    trustLevel     [0] INTEGER OPTIONAL,
    transitive     [1] INTEGER DEFAULT 0 }
```

```
CrossSpecification ::= CHOICE {
    superior      [0] NULL,      -- my immediate superior
    ancestors     [1] NULL,      -- all superiors
    subordinates  [2] NULL,      -- normal CA hierarchy
    descendants   [3] NULL,      -- any descendant CA
    name          [4] Name,      -- individual CAs
    group         [5] Name,      -- group of names (of CAs)
    subtree       [6] Subtree }  -- all CAs in a subtree
```

The list of CAs which may be cross certified may include CA names, directory subtrees (possibly containing a hierarchy of CAs), group names where the (non hierarchical) group is a list of CA names, and various CAs whose names bear a relationship to the name of the CA in question:

- a) the immediate superior CA or all superior CAs (up to the TLCA);
- b) all immediately subordinate CAs; and
- c) all subordinate CAs at any depth.

An explicit level of trust may be specified, as well as an indication of whether cross-certification applies transitively, i.e. if certificates in a domain which is cross-certified by the CA named in the entry will be trusted. Transitivity is indicated by specifying the number of cross certificates which may be in a chain rooted on the specified CA, i.e. the number of domain boundaries crossed.

The Subtree is defined in X.501 (1993).

```
Users ::= SEQUENCE OF UserEntry

UserEntry ::= SEQUENCE {
    userSpec      UserSpecification,
    trustLevel    [0] INTEGER OPTIONAL,
    transitive     [1] INTEGER DEFAULT 0 }

UserSpecification ::= CHOICE {
    subordinates  [0] NULL,      -- my subordinates
    name          [1] Name,      -- individuals
    group         [2] Name,      -- group of names
    subtree       [3] Subtree } -- whole subtree
```

The users which a CA may certify may include (all) subordinates (a very common case), as well as individual names, names of groups (Directory entries which contain lists of user names), and subtrees as defined above.

```
trust-specification
    WITH ATTRIBUTE-SYNTAX TrustSpecification
    SINGLE VALUE
::= id-trust-specification
```

### **7.8.12 Transaction Limit**

This attribute represents the maximum monetary value of a message (transaction) which the entity may sign.

```
TransactionLimit ::= MonetaryValue

transaction-limit ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX TransactionLimit
::= id-transaction-limit
```

### **7.8.13 Transaction Type**

This attribute represents a transaction type which the entity may sign. (Multiple values of the attribute may be present.)

```
TransactionType ::= OBJECT IDENTIFIER

transaction-type ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX TransactionType
::= id-transaction-type
```

### **7.8.14 Time Of Day**

This attribute describes the time periods during which signatures from this entity are considered valid.

```
TimeConstraint ::= SEQUENCE {
    daysOfWeek      BIT STRING { sunday(0), monday(1),
                                tuesday(2), wednesday(3), thursday(4),
                                friday(5), saturday(6) },
    intervalsOfDay  IntervalsOfDay }

IntervalsOfDay ::= SET OF SEQUENCE {
    intervalStart  Time24,
    intervalEnd    Time24 }

Time24 ::= SEQUENCE {
    hour           INTEGER (0..23),
    minute         INTEGER (0..59) }

time-of-day ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX TimeConstraint
::= id-time-of-day
```

### 7.8.15 Location

This attribute indicates the valid location(s) a transaction may be submitted from.

```
Location ::= CHOICE {
    [0] PresentationAddress,
    [1] IPAddress,
    [2] X121Address }

IPAddress ::= OCTET STRING (SIZE 6)
```

PresentationAddress and X121Address are defined in X.520.

```
location ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX Location
::= id-location
```

### 7.8.16 Authorized Signatory

This attribute may be used in the attribute certificate of an organizational entity to formally indicate the identities of individuals authorized to sign for the organization.

```
AuthorizedSignatory ::= Name

authorized-signatory ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX AuthorizedSignatory
::= id-authorized-signatory
```

### 7.8.17 Pre-approved Counter Party

This attribute may be used to indicate entities with which the certified entity is pre-authorized to conduct financial transactions (e.g., customers or suppliers).

```
PreApprovedCounterParty ::= Name

preapproved-counterparty ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX PreApprovedCounterParty
::= id-preapproved-counterparty
```

### 7.8.18 Delegation Controls

This attribute may be used to indicate the amount of "authority" an entity may delegate to another entity when issuing an attribute certificate.

```
DelegationControl ::= SEQUENCE {
    delegation      Delegation,
    limit           TransactionLimit,
    types           SET OF TransactionType }

delegation ::= ENUMERATED {
    exercise (0), -- may not delegate
    deputy (1),  -- may delegate exercise of authority
    officer (2), -- may subdelegate up to deputy
    master (3) } -- may delegate anything

delegation-control ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX DelegationControl
::= id-delegation-control
```

## 8 Lower Layers Security

## 9 Upper Layers Security

Refer to Part 12, clause 9 of the Stable Agreements Document.

### 9.1 Security Mechanisms

#### 9.1.1 Peer Entity Authentication

ACSE authentication extensions [ISO8649][ISO8650/1] support two-way authentication through the definition of a new functional unit. When this functional unit is employed, additional parameters are provided by the ASSOCIATE service to indicate this requirement and convey authentication information between entities. The ASN.1 definition for this information is given below:

from [ISO8650/1]:

```
Mechanism-name ::= OBJECT IDENTIFIER
  --This field shall be present if authentication-value is of type ANY.
Authentication-value := CHOICE {
  charstring          [0] IMPLICIT GraphicString,
  bitstring           [1] IMPLICIT BIT STRING,
  external            [2] IMPLICIT EXTERNAL,
  other               [3] IMPLICIT SEQUENCE {

      other-mechanism-name  Mechanism-name,
      other-mechanism-value ANY DEFINED BY
        other-mechanism-name }}

  -- The abstract syntax of authentication-value is determined by the
  -- authentication-mechanism used during association establishment. The
  -- authentication-mechanism is either explicitly denoted by the OBJECT IDENTIFIER
  -- value for Mechanism-name, or it is know implicitly by prior agreement between
  -- the communicating partners. If "other" is chosen, then "Mechanism-name"
  -- must be present in accordance with ISO 8824.
```

These agreements define the following mechanisms for use with this ACSE functional unit:

simple-strong authentication mechanism.

##### 9.1.1.1 Simple-Strong Authentication





## **11 Directory Services Security**

## **12 Network Management Security**

### **12.1 Threats**

Refer to clause 12.1 of the Stable Implementation Agreements.

### **12.2 Security Services**

Refer to clause 12.2 of the Stable Implementation Agreements.

### **12.3 Security Mechanisms**

#### **12.3.1 Peer-Entity Authentication**

Refer to 12.3.1 of the Stable Implementation Agreements.

#### **12.3.2 Connectionless Integrity**

In order to identify whether changes to a data unit have occurred it is proposed that an integrity check value (ICV) be computed over the entire data unit and included in the protocol control information for that data unit. The specification and location for conveying this information is left for further study. Because of the envisaged relationship between the underlying mechanisms employed for data origination authentication and connectionless integrity, they are to be considered jointly.

#### **12.3.3 Data Origination Authentication**

The proposed security mechanism for data origination authentication is encipherment and intended to protect the ICV computed for connectionless integrity. Successful peer authentication results in the establishment of a cryptographic association between network management entities. The association allows the originator of a data unit to encrypt it or portions of it, and have the peer recipient verify origination through decryption. In order to minimize computational effort, it is proposed that only the integrity check value be enciphered (i.e., a signature) rather than the entire data unit.

This approach implies that data origination authentication information resides with the integrity check value, and that an according ASN.1 definition reflect any requirements of the signing algorithm or choice of algorithm. However, there appears to be no appropriate location in the application layer protocols employed by network management

to convey such data origination authentication information. This issue is left for further study.

#### **12.3.4 Connectionless Confidentiality**

---

**Annex A** (normative)

---

**ISPICS Requirements List**

---

**Annex B** (normative)

---

**Errata****Table 4 - WIA Part 12 Changes**

<b>NO. OF ERRATA</b>	<b>TYPE</b>	<b>REFERENCED DOCUMENT</b>	<b>CLAUSE</b>	<b>NOTES</b>
	TECHNICAL	WIA PART - 13	12	ADDED NEW
	TECHNICAL	WIA PART - 13	11	ADDED NEW
	TECHNICAL	WIA PART - 13	5.2/3	ADDED NEW
	TECHNICAL	WIA PART - 13	8	ADDED NEW
	TECHNICAL	SIA PART - 12	0..12	ADD OUTLINE 2ND LEVEL
	TECHNICAL	SIA PART - 12	9	ADD TEXT
	TECHNICAL	SIA PART - 12	12.1.2	ADD TEXT
	TECHNICAL	SIA PART - 12	12.2.2	ADD TEXT
	TECHNICAL	SIA PART - 12	12.4.1/.2	ADD TEXT

---

**Annex C** (normative)

---

**Security Labels**

**Editor's Note** - Agreements about security labels is a future work item.

---

**Annex D (normative)**


---

**Security Algorithms and Attributes**

```

OIWSECSIGAlgorithmObjectIdentifiers { iso(1) identified-organization(3)
                                      oiw(14) secsig(3)
                                      oIWSECSIGAlgorithmObjectIdentifiers(1) }

DEFINITIONS =
BEGIN

EXPORTS
-- to be determined

IMPORTS
-- none

-- category of information object
-- defining our own here; perhaps the definition should be imported from
-- { joint-iso-ccitt ds(5) modules(1) usefulDefinitions(0) }
-- This annex contains OIW registrations only; refer to section 7 algorithm
-- descriptions algorithms IDs.

algorithm      OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
                                      oiw(14) secsig(3) algorithm(2) }

-- macros

-- taken from { joint-iso-ccitt ds(5) modules(1) authenticationFramework(7) }
ALGORITHM MACRO      ::=
BEGIN
TYPE NOTATION        ::= "PARAMETER" type
VALUE NOTATION       ::= value(VALUE OBJECT IDENTIFIER)
END -- of ALGORITHM

-- algorithms

md4WithRSA ALGORITHM
  PARAMETER NULL
  ::= { algorithm 2 }

md5WithRSA ALGORITHM
  PARAMETER NULL
  ::= { algorithm 3 }

md4WithRSAAEncryption ALGORITHM
  PARAMETER NULL
  ::= { algorithm 4 }

```

```
desECB ALGORITHM
  PARAMETER NULL
  ::= { algorithm 6 }

desCBC ALGORITHM
  PARAMETER CBCParameter
  ::= { algorithm 7 }

CBCParameter ::= IV

desOFB ALGORITHM
  PARAMETER FBParameter
  ::= { algorithm 8 }

desCFB ALGORITHM
  PARAMETER FBParameter
  ::= { algorithm 9 }

FBParameter ::= SEQUENCE {
  iv IV,
  numberOfBits NumberOfBits}

NumberOfBits ::= INTEGER    -- Number of feedback bits (1 to 64 bits)
```

**Editor's Note** - Check FIPS PUB 81 for allowed ranges of feedback bits and specify ranges here as a comment.

```
IV ::= OCTET STRING    -- 8 octets

desMAC ALGORITHM
  PARAMETER MACParameter
  ::= { algorithm 10 }

MACParameter ::= INTEGER    -- Length of MAC (16, 24, 32, 40, 40 or 64 bits)
```

**Editor's Note** - Check FIPS PUB 113 for allowed

```
rsaSignature ALGORITHM
  PARAMETER NULL
  ::= { algorithm 11 }

dsa ALGORITHM
  PARAMETER DSAParameters
  ::= { algorithm 12 }

dsaWithSHA ALGORITHM
  PARAMETER DSAParameters
  ::= { algorithm 13}

mdc2WithRSASignature
  PARAMETER NULL
  ::= { algorithm 14 }
```



## PART 12 - Security

March 1994 (Working)

```
shaWithRSASignature
  PARAMETER NULL
  ::= { algorithm 15 }

dhWithCommonModulus ALGORITHM
  PARAMETER NULL
  ::= { algorithm 16 }

desEDE ALGORITHM
  PARAMETER NULL
  ::= { algorithm 17 }

sha ALGORITHM
  PARAMETER NULL
  ::= { algorithm 18 }

mdc-2 ALGORITHM
  PARAMETER NULL
  ::= { algorithm 19 }

dsaCommon ALGORITHM
  PARAMETER NULL
  ::= { algorithm 20 }

dsaCommonWithSHA ALGORITHM
  PARAMETER NULL
  ::= { algorithm 21)

rsaKeyTransport ALGORITHM
  PARAMETER NULL
  ::= { algorithm 22 }

keyed-hash-seal ALGORITHM
  PARAMETER NULL
  ::= { algorithm 23 }

md2WithRSASignature
  PARAMETER NULL
  ::= { algorithm 24 }

md5WithRSASignature
  PARAMETER NULL
  ::= { algorithm 25 }
```

```
authentication-mechanism OBJECT IDENTIFIER ::=
  { iso(1) identified-organization(3) oiw(14) secsig(3) auth-mechanism(3) }

attribute OBJECT IDENTIFIER ::=
  { iso(1) identified-organization(3) oiw(14) secsig(3) attribute(4) }
id-liability-limitation OBJECT IDENTIFIER ::= { attribute 1 }
id-binding-information OBJECT IDENTIFIER ::= { attribute 2 }
id-trusted-third-party OBJECT IDENTIFIER ::= { attribute 3 }
id-cosignature-requirements OBJECT IDENTIFIER ::= { attribute 4 }
id-certificate-purpose OBJECT IDENTIFIER ::= { attribute 5 }
id-relative-identity OBJECT IDENTIFIER ::= { attribute 6 }
id-trust-specification OBJECT IDENTIFIER ::= { attribute 7 }
id-transaction-limit OBJECT IDENTIFIER ::= { attribute 8 }
id-transaction-type OBJECT IDENTIFIER ::= { attribute 9 }
id-location OBJECT IDENTIFIER ::= { attribute 10 }
id-time-of-day OBJECT IDENTIFIER ::= { attribute 11 }
id-authorized-signatory OBJECT IDENTIFIER ::= { attribute 12 }
id-preapproved-counterparty OBJECT IDENTIFIER ::= { attribute 13 }
id-delegation-control OBJECT IDENTIFIER ::= { attribute 14 }

doc-type OBJECT IDENTIFIER ::=
  { iso(1) identified-organization(3) oiw(14) secsig(3)
    doc-type(5) }

id-doc-drivers-license OBJECT IDENTIFIER ::= { doc-type 1 }
id-doc-passport OBJECT IDENTIFIER ::= { doc-type 2 }
id-doc-alien-registration OBJECT IDENTIFIER ::= { doc-type 3 }
id-doc-birth-certificate OBJECT IDENTIFIER ::= { doc-type 4 }

module OBJECT IDENTIFIER ::=
  { iso(1) identified-organization(3) oiw(14) secsig(3) module (6) }
x9f1-certmgmt OBJECT IDENTIFIER ::= { module 1 }

END -- of Algorithm Object Identifier Definitions
```

---

**Annex E** (normative)

---

**References for Security Algorithms**

(See the Stable Document).

---

**Annex F** (informative)

---

**Bibliography**

(See the Stable Document).

---

**Annex G** (normative)

---

**El Gamal**

---

**Annex H** (informative)

---

**STATUS****Table 5 - ISO Status**

<b>DOCUMENT</b>	<b>WD</b>	<b>CD</b>	<b>DIS</b>	<b>IS</b>
ISO/IEC JTC1 SC21/WG1 N5044	X	X	X	6/91
NETWORK LAYER ISO/IEC JTC1 SC6	X	7/91		
TRANSPORT LAYER ISO/IEC JTC1 SC6	X	X	7/91	
LOWER LAYER ISO/IEC JTC1 SC6 6227	X			

**NOTE** - This table was not included in any motion presented to the Plenary in December 1990.

## Annex I (informative)

## Security-SIG Management Plan

Table 6 - Management Plan

Document	Next Milestone	Date
ISO/IEC JTC1 SC21 N3614		
ISO/IEC DP 9796		
SDN-601/NIST IR 90-4262	COMPLETED	
SDN-701/NIST IR 90-4250	COMPLETED	
SDN-702/NIST IR 90-4250	COMPLETED	
ISO/IEC JTC1 SC21/WG1 N5002		
SDN-902/NIST IR 90-4262	COMPLETED	
SDN-903/NIST IR 90-4262	COMPLETED	
ISO/IEC JTC1 SC21/WG1 N4110		
SDN-301/NIST IR 90-4250	COMPLETED	
SDN-401/NIST IR 90-4250	COMPLETED	
SDN-906/NIST IR 90-4262	COMPLETED	
ISO/IEC JTC1 SC21/WG1 N5001		
ISO/IEC JTC1 SC21/WG1 F29 N5045		
ISO/IEC JTC1 SC21/WG1 F30		
ISO/IEC JTC1 SC21/WG1 F31 N5047		
ISO/IEC JTC1 SC21/WG1 F32 N5046		
ISO/IEC JTC1 SC21/WG4 N3775		
ISO/IEC JTC1 SC21/WG1 N4110		
ISO/IEC JTC1 SC21/WG7 N4022		
ISO/IEC JTC1 SC21/WG1 N5048		
ISO/IEC JTC1 SC21/WG1 N5049		
ISO/IEC JTC1 SC21/WG1 N5044	IS	6/91
NETWORK LAYER ISO/IEC JTC1 SC6	CD	7/91
TRANSPORT LAYER ISO/IEC JTC1 SC6 6285	DIS	7/91
LOWER LAYER ISO/IEC JTC1 SC6 6227	WD	N/A

---

## Annex J (informative)

---

### Key Management

Many of the security services defined for use within OSI protocols and applications are provided by the use of cryptographic techniques. The use of these techniques requires that cryptographic keys are available.

Key management is the generic name covering the process required to ensure such availability. A definition of the objective for key management is thus:

- a) To provide suitable cryptographic keys to security services that require such keys in a secure and timely manner.

This area has been studied for a number of years and specific solutions produced to address needs in well defined situations; the defense and banking communities are examples.

The general problem of key management in a nonspecific OSI environment has not, however, been addressed. And hence OSI key management standards do not exist; whilst responsibility for them has been assigned, work to produce such standards is only just starting.

#### J.1 Definition of Key Management

Key management is the collection of procedures and services that support the generation, storage, transport, and destruction of cryptographic key material. Specifically, with respect to OIW agreements, key management supports the security services specified in the OIW protocol ISPs.

#### J.2 Tutorial on Key Management

This tutorial provides information on the role of key management within an overall security architecture. It addresses the requirements OSI security services place on key management. It describes the issues that arise specifically with regard to the administration of keying material, approaches to key distribution, and the relationship of these approaches to the requirements and concerns referred to above.

##### J.2.1 Requirements of Key Management

This section identifies the generic and specific requirement that security services and protocols place on key management.

- a) Symmetric (private, single key);

All parties belong to the same cryptographic network and hold the same private key which is known only to the members of that network. This one key is used by all members for both encryption and decryption. The network can



be as small as 2, or as large as thousands. However, to minimize damage in the event the key is broken, the network size is kept small.

b) Asymmetric (public, two key);

There is no cryptographic network as in the sense of symmetric keying. Each user holds two keys: an encrypt and a decrypt. The decrypt key is private and known only to the holder. Each user's encrypt key is also placed at a point of public access where all other users can obtain it. A user who wishes to send encrypted information to another user would RETRIEVE the intended recipient's public encrypt key from the common storage area and use it to encrypt the information to be sent to the recipient. The recipient would then use his own private decrypt key to decrypt the information.

c) Intermediary

This key scheme is one in which each user holds his own private key known only to himself and to a trusted intermediary. The users encrypt information to be sent to the intended recipient using his private key and then sends it to the intermediary. The intermediary decrypts the information using the user's private key, re-encrypts the information using the intended recipient's private key, and sends the information to the intended recipient. The intended recipient then decrypts the information using its own private key.

## **J.2.2 Key Administration**

One of the primary tasks of key management is the administration of keying material. There are several general issues which arise in this context.

### **J.2.2.1 Generation**

Key management is responsible for ensuring the availability of keys when required. The provision of cryptographic keys may be by a process internal to [the] key management [system] or by an external process.

Generated keys must be suitable for use by the key requestor. This suitability is determined by the cryptographic algorithm to be used by the requestor.

### **J.2.2.2 Validation**

TBD

### **J.2.2.3 Expiration**

Key management must have provision for expiring keys, including time limit expiration and expiration due to compromise.

**J.2.2.4      Audit**

Key management must maintain an audit trail of its activities. There must be capabilities for reporting this information in an appropriate fashion.

**J.2.2.5      Authorization/Authentication**

Key management may require the requesting security service authentication itself to key management to determine the validity of the request.

**J.2.3        Approaches to Key Distribution**

There are several extant approaches to key management. These include public key and certificate methods, symmetric key techniques, and proposals to use network management for key manager.

**J.2.3.1      Symmetric**

Network management provides an alternate view of key management. The basic approach here is to treat keying material as management information to be manipulated.

There are two ways to structure this. The security services could generate a "key management event" and the key management service could respond with a keying material. There are difficulties with this because of the difficulty in assuring event delivery.

Alternatively, the security services could be seen as the manager generating get and put commands to enable the communication of keying material.

The largest concern with this approach is that unless combined with one of the others, one merely re-introduces the key-management problem in order to provide peer-entity authentication, integrity, and confidentiality for the key exchange.

**J.2.3.1.1      Certificate**

**J.2.3.1.2      Symmetric Generation**

**J.2.3.1.3      Centralized**

**J.2.3.1.4 External**

**J.2.3.2 Asymmetric**

Public key techniques are mostly commonly used for authentication and data integrity where the amount of information being protected is relatively small. These can also be used as an underlying mechanism to implement a symmetric private key exchange.

Public key technology can also be coupled with certificates or other methods of relating public keys to identifies. Doing this provides peer entity authentication based on the strength of the relationship between keys and identities. Directory stored certificated (possibly with local caching) are an example of a method of this type.

**J.2.3.2.1 Certificate**

**J.2.3.2.2 Centralized**

**J.2.3.2.3 External**

**J.3 Key Management Architectures**

**J.3.1 Existing Systems**

**J.3.1.1 SDNS**

**J.3.1.2 SILS**

**J.3.1.3 ANSI X9.17**

**J.3.1.4 Kerberos**

**J.3.2**    **OSI**  
TBD

**J.4**    **Current Issues**

**J.5**    **Related Organizations**

**J.5.1**    **ANSI X.9**

**J.5.2**    **SC21**

**J.5.3**    **SC27**

**J.5.4**    **IEEE 802.10**

**J.6**    **References**

---

**Annex K (informative)**


---

**Base Environment Threats**

Table 7 defines the services required to protect against various threats in a Base Environment.

Each X in the table identifies a security service which offers protection against the corresponding threat.

**Table 7 - Threats to Sevices**

THREAT	SECURITY SERVICES				
	AUTH.	ACC. CTRL	DATA CONF.	DATA INTEG.	NON- REPUD.
MASQUERADE	X				
REPLAY			X	X	
MODIFICATION OF MESSAGE		X		X	
DENIAL OF SERVICE		X		X	
TRAP DOOR		X		X	
TROJAN HORSE		X	X	X	
DISCLOSURE		X	X		
REPUDIATION					X