4851903 0660873 627

INTERNATIONAL STANDARD

ISO/IEC 11586-1

First edition
1996-06-01

Information technology — Open Systems Interconnection — Generic upper layers security: Overview, models and notation

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — Sécurité des couches supérieures génériques: Présentation, modèles et notation

Reference number
ISO/IEC 11586-1:1996(E)

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:06 1999

Contents

		Page
1	Scope	1
2	Normative references	1
2.1	Identical Recommendations I International Standards	2
2.2	Paired Recommendations I International Standards equivalent in technical content	2
3	Definitions	2
4	Abbreviations	4
5	General overview	4
6	Security exchanges	5
6.1	Security exchange model	5
6.2	Notation for specifying security exchanges	6
7	Security transformations	7
7.1	Security transformation model	7
7.2	Notation for specifying security transformations	11
8	Abstract syntax notation for selective field protection	12
8.1	Basic notation	12
8.2	Notation with transformation qualifier	14
8.3	Mapping protection requirements to security transformations	15
8.4	Notation for specifying protection mappings	15
9	Conformance	16
Annex A – ASN.1 definitions		17
Annex B – Registration of security exchanges and security transformations		22
Annex C – Security exchange specifications		23
Annex D – Security transformation specifications		27
Annex E – Protection mapping specifications		38
Annex F – Object identifier usage		41
Annex G – Guidelines for the use of generic upper layers security facilities		42
Annex H – Relationship to other standards		47
Annex I – Examples of use of the generic upper layers security facilities		50
Annex J – Bibliography		54

© ISO/IEC 1996

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:18 1999

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 11586-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 21, *Open systems interconnection, data management and open distributed processing*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.830.

ISO/IEC 11586 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — Generic upper layers security*:

— *Part 1: Overview, models and notation*

— *Part 2: Security Exchange Service Element (SESE) service definition*

— *Part 3: Security Exchange Service Element (SESE) protocol specification*

— *Part 4: Protecting transfer syntax specification*

— *Part 5: Security Exchange Service Element Protocol Implementation Conformance Statement (PICS) proforma*

— *Part 6: Protecting transfer syntax Protocol Implementation Conformance Statement (PICS) proforma*

Annexes A to F form an integral part of this part of ISO/IEC 11586. Annexes G to J are for information only.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:19 1999

Introduction

This Recommendation | International Standard forms part of a series of Recommendations | multi-part International Standards, which provide(s) a set of facilities to aid the construction of Upper Layers protocols which support the provision of security services. The parts are as follows:

- Part 1: Overview, Models and Notation;
- Part 2: Security Exchange Service Element Service Definition;
- Part 3: Security Exchange Service Element Protocol Specification;
- Part 4: Protecting Transfer Syntax Specification;
- Part 5: Security Exchange Service Element PICS Proforma;
- Part 6: Protecting Transfer Syntax PICS Proforma.

This Recommendation | International Standard constitutes Part 1 of this series.

For informative guidelines on the application of all facilities described in this series, see Annex G.

It is important to note that these generic security facilities do not in themselves provide security services; they are simply construction tools for security-related protocols. Furthermore, these facilities do not necessarily provide a stand-alone solution to all security communications requirements of applications. Application standards may still need to incorporate security features within their specifications, to work in conjunction with generic security services supported by the Generic Upper Layers Security facilities.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:20 1999

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – GENERIC UPPER LAYERS SECURITY: OVERVIEW, MODELS AND NOTATION

1 Scope

1.1 This series of Recommendations | International Standards defines a set of generic facilities to assist in the provision of security services in OSI applications. These include:

a) a set of notational tools to support the specification of selective field protection requirements in an abstract syntax specification, and to support the specification of security exchanges and security transformations;

b) a service definition, protocol specification and PICS proforma for an application-service-element (ASE) to support the provision of security services within the Application Layer of OSI;

c) a specification and PICS proforma for a security transfer syntax, associated with Presentation Layer support for security services in the Application Layer.

1.2 This Recommendation | International Standard defines the following:

a) general models of security exchange protocol functions and security transformations, based on the concepts described in the OSI Upper Layers Security Model (ITU-T Rec. X.803 | ISO/IEC 10745);

b) a set of notational tools to support the specification of selective field protection requirements in an abstract syntax specification, and to support the specification of security exchanges and security transformations;

c) a set of informative guidelines as to the application of the generic upper layers security facilities covered by this series of Recommendations | International Standards.

1.3 This Recommendation | International Standard does not define the following:

a) a complete set of upper layer security facilities which may be required by other Recommendations | International Standards;

b) a complete set of security facilities for specific applications;

c) the mechanisms employed to support security services.

1.4 The security exchange model, and supporting notation, are intended both for use as the basis of defining the security exchange service element in subsequent parts of this series of Recommendations | International Standards, and for use by any other ASE which may import security exchanges into its own specification.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:20 1999

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*
- ITU-T Recommendation X.207 (1993) | ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application Layer structure.*
- ITU-T Recommendation X.214 (1993) | ISO/IEC 8072:1994, *Information technology – Open Systems Interconnection – Transport service definition.*
- ITU-T Recommendation X.216 (1994) | ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition.*
- ITU-T Recommendation X.217 (1995) | ISO/IEC 8649: ...[1], *Information technology – Open Systems Interconnection – Service definition for the Association Control Service.*
- ITU-T Recommendation X.226 (1994) | ISO/IEC 8823-1:1994, *Information technology – Open Systems Interconnection – Connection-oriented presentation protocol: Protocol specification.*
- ITU-T Recommendation X.509 (1993) | ISO/IEC 9594-8:1995, *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*
- ITU-T Recommendation X.511 (1993) | ISO/IEC 9594-3:1995, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition.*
- CCITT Recommendation X.660 (1992) | ISO/IEC 9834-1:1993, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: General procedures.*
- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*
- ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1995, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*
- ITU-T Recommendation X.803 (1994) | ISO/IEC 10745:1995, *Information technology – Open Systems Interconnection – Upper layers security model.*
- ITU-T Recommendation X.811 (1995) | ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework.*
- ITU-T Recommendation X.812[1] | ISO/IEC 10181-3: ...[1], *Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework.*

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.800 (1991), *Security architecture for Open Systems Interconnection for CCITT applications.*

  ISO 7498-2:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture.*

3 Definitions

3.1 The following term is used as defined in ITU-T Rec. X.200 | ISO/IEC 7498-1:

- transfer syntax.

[1] Presently at the stage of draft.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:20 1999

3.2 The following terms are used as defined in CCITT Rec. X.800 | ISO 7498-2:

- access control;
- confidentiality;
- data origin authentication;
- decipherment;
- digital signature;
- encipherment;
- integrity;
- key;
- key management;
- selective field protection.

3.3 The following terms are used as defined in ITU-T Rec. X.216 | ISO/IEC 8822:

- abstract syntax;
- presentation context;
- presentation data value.

3.4 The following terms are used as defined in ITU-T Rec. X.207 | ISO/IEC 9545:

- application-association
- application-context;
- application-service-element (ASE);
- application-service-object-association (ASO-association).

3.5 The following terms are used as defined in ITU-T Rec. X.811 | ISO/IEC 10181-2:

- authentication exchange;
- claimant;
- entity authentication;
- verifier.

3.6 The following term is used as defined in ITU-T Rec. X.812 | ISO/IEC 10181-3:

- access control certificate.

3.7 The following terms are used as defined in ITU-T Rec. X.803 | ISO/IEC 10745:

- security association;
- security communication function (SCF);
- security exchange;
- security exchange item;
- security exchange function;
- security transformation;
- system security object (SSO).

3.8 For the purposes of this Recommendation | International Standard, the following definitions apply:

3.8.1 **presentation-context-bound security association:** A security association which is established in conjunction with the establishment of a protecting presentation context, and which applies to all presentation data values sent in one direction in that protecting presentation context; attributes of the security association are indicated explicitly along with the encoding of the first presentation data value in the protecting presentation context.

3.8.2 **single-item-bound security association:** A security association applying to a single independently-protected presentation data value which is not associated with a presentation context; attributes of the security association are indicated explicitly along with the presentation data value encoding.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:21 1999

3.8.3 externally-established security association: A security association which is established independently of instances of its use, and which has a globally-unique identifier enabling it to be referenced at the time of use.

3.8.4 initial encoding rules: The ASN.1 encoding rules used to generate an unprotected bit string from a value of an ASN.1 type, when that value is to be protected using a security transformation.

3.8.5 protecting presentation context: A presentation context which associates a protecting transfer syntax with an abstract syntax.

3.8.6 protecting transfer syntax: A transfer syntax which employs a security transformation.

3.8.7 protection mapping: A specification which relates a protection requirement, identified by name in an abstract syntax specification, to a specific security transformation to be used to satisfy that requirement.

4 Abbreviations

ACSE	Association Control Service Element
ASE	application-service-element
ASO	application-service-object
GULS	Generic Upper Layers Security
OSI	Open Systems Interconnection
PDU	protocol-data-unit
PDV	Presentation Data Value
PICS	Protocol Implementation Conformance Statement
SCF	Security Communication Function
SEI	Security Exchange Item
SESE	Security Exchange Service Element
SSO	System Security Object

5 General overview

The Generic Upper Layers Security (GULS) Standards define a set of protocol-construction tools and protocol components to support the provision of security protection for applications. These facilities support the provision of security services in the OSI Upper Layers (Application Layer, sometimes with Presentation Layer support).

NOTE – Security services for OSI applications may be provided using security mechanisms at either the upper or lower layers. In the latter case, the protection is obtained by specifying an appropriate protection quality-of-service (as defined in ITU-T Rec. X.214 | ISO/IEC 8072) to ACSE when establishing an application-association. This protection quality-of-service is passed transparently through the Presentation and Session Layers to the transport service. Provision of security services in the lower layers is outside the scope of this Recommendation | International Standard.

The facilities provided in the GULS Standards include:

- a general means of building Application Layer protocol components to support the exchange of security-related information between a pair of communicating application-entity-invocations (the security exchange concept, which is supported by the SESE); these facilities are described in clause 6;
- a general approach to using Presentation Layer facilities to perform security-related transformations on information items in order to protect these items (the security transformation concept, which is supported by a generic protecting transfer syntax); these facilities are described in clause 7;
- abstract syntax notation tools to assist an application protocol designer in specifying that security protection is to apply to selected fields of this protocol (a PROTECTED parameterized type, and the PROTECTED-Q variant of this type); these facilities are described in clause 8.

Security exchanges are used for such purposes as entity authentication and key management. Security transformations (and the generic protecting transfer syntax and/or PROTECTED parameterized type or its variations) are used for such purposes as integrity, confidentiality, data origin authentication and/or non-repudiation.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:21 1999

The Upper Layers Security Model (ITU-T Rec. X.803 | ISO/IEC 10745) provides the architectural model for the GULS specifications. It describes the roles of security exchange functions and security transformations.

Security exchange functions provide the means for communicating security information between application-entity-invocations as part of the operation of a security mechanism, i.e. they generate and process application-protocol-control-information with a security-related purpose. Security exchanges may be specified using the notation in this Recommendation | International Standard, then imported into any abstract syntax specification. The Security Exchange Service Element (SESE) is an application-service-element (ASE) defined in ITU-T Rec. X.831 | ISO/IEC 11586-2 and ITU-T Rec. X.832 | ISO/IEC 11586-3. The SESE provides a way of conveying security exchanges, which supports the goal of making application-specific ASEs independent of security mechanisms used. However, some aspects of the specification of an application which directly incorporates security provisions will be mechanism dependent.

Security transformations can be supported by the generic protecting transfer syntax described in ITU-T Rec. X.833 | ISO/IEC 11586-4.

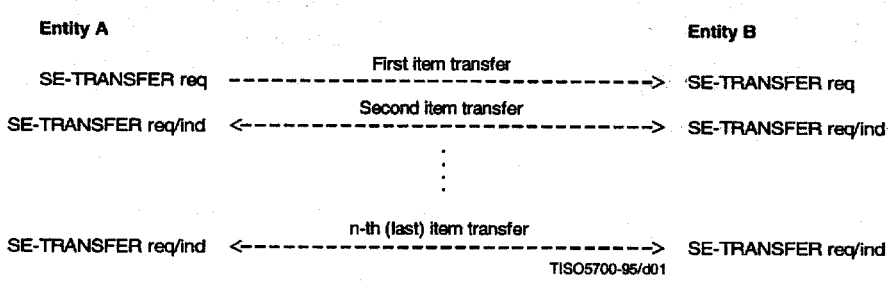
6 Security exchanges

6.1 Security exchange model

This Recommendation | International Standard defines the procedural model for a security exchange, as introduced in ITU-T Rec. X.803 | ISO/IEC 10745.

A security exchange occurs between two entities A and B. It consists of the transfer of one Security Exchange Item (SEI) from A to B, possibly followed by a sequence of one or more transfers of SEIs in either direction between A and B. The number of transfers depends upon the particular security exchange. Each SEI may comprise an arbitrarily complex data structure, representable by any ASN.1 type. It may include components which are individually protected using the PROTECTED notation described in clause 8.

The time-sequence diagram in Figure 1 illustrates the sequence of SEI transfers for an n-way security exchange, and the corresponding SESE service primitive invocations, as defined in ITU-T Rec. X.831 | ISO/IEC 11586-2.



Note – The double-sided arrow indicates that the transfer could be send by either A or B.

Figure 1 – Security exchange model

There are two classes of exchange:

- *Alternating* – Successive item transfers occur in alternating directions and only one transfer is active at any time;
- *Arbitrary* – There are no constraints upon direction of any transfer and transfers in the two directions may be active simultaneously.

While a security exchange is in progress, other information transfers may occur and other security exchanges may be in progress on the same application-association. However, application-context rules will usually constrain such overlapping activities. Presentation data values conveying SEIs may be concatenated with, interleaved with, or embedded in other presentation data values.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:22 1999

6.2 Notation for specifying security exchanges

A specification of a security exchange includes a specification of the types of SEIs that may be exchanged, a statement of any ordering constraints applying to transfers of these SEIs, a statement of error conditions that may result from the transfer of each SEI, and a statement of (or reference to) the associated semantics.

A security exchange definition includes:

a) assignment of a global object identifier or a local integer value to the security exchange, to allow its use to be unambiguously identified in protocol;

b) a specification of the abstract syntax of the SEIs and error notifications transferred in the security exchange.

To support specification of this information in a form which can be used by the SESE protocol, three ASN.1 information object class definitions (see ITU-T Rec. 681 | ISO/IEC 8824-2) are provided:

- the SECURITY-EXCHANGE information object class is used in specifying a particular security exchange; an information object of this class contains one or more SEC-EXCHG-ITEM information objects;
- the SEC-EXCHG-ITEM information object class is used to define one SEI; an information object of this class may contain one or more ERROR information objects;
- the SE-ERROR information object class is used to define an error condition which may result from the transfer of a SEI.

NOTE – Annex G provides guidelines showing how these information object classes are used in the definition of a complete Application Context.

```
SECURITY-EXCHANGE ::= CLASS
-- This information object class definition is for use when
-- specifying a particular instance of a security exchange.
{
  &SE-Items        SEC-EXCHG-ITEM,
  -- This is an ASN.1 information object set, comprising a set
  -- of security exchange items
  &sE-Identifier   Identifier        UNIQUE
  -- A local or global identifier for the particular security exchange
}
WITH SYNTAX
-- The following syntax is used to specify a particular security exchange.
{
  SE-ITEMS         &SE-Items
  IDENTIFIER       &sE-Identifier
}
Identifier ::= CHOICE
{
  local            INTEGER,
  global           OBJECT IDENTIFIER
}

SEC-EXCHG-ITEM ::= CLASS
{
  &ItemType,
      -- ASN.1 type for this exchange item
  &itemId          INTEGER,
      -- Identifier for this item, e.g. 1, 2, 3, ..
  &Errors          SE-ERROR          OPTIONAL
      -- Optional list of errors which may result from transfer of this item
}
WITH SYNTAX
{
  ITEM-TYPE        &ItemType
  ITEM-ID          &itemId
  [ERRORS          &Errors]
}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:22 1999

```
SE-ERROR ::= CLASS
{
  &ParameterType     OPTIONAL,
  -- ASN.1 type of a parameter to accompany the signalling
  -- of the error condition back to the sender of the SEI
  &errorCode         Identifier        UNIQUE
  -- An identifier used in signalling the error condition
  -- back to the sender of the SEI
}
WITH SYNTAX
{
  [PARAMETER         &ParameterType]
  ERROR-CODE         &errorCode
}
```

Examples of the use of this notation are given in Annex C.

7 Security transformations

7.1 Security transformation model

A security transformation is a security function (or combination of security functions) applied to user data to protect that data during communication or storage. A security transformation involves an encoding process applied prior to communication or storage, and a decoding process which may be (but need not always be) applied upon receipt or retrieval. Examples of security transformations are:

a) applying an encipherment process on encoding of data and a corresponding decipherment process on decoding;

b) generating a seal or signature and appending it to data on encoding and checking and removing the appended seal or signature on decoding;

c) combining the functions in a) and b) into one security transformation.

Security transformations defined using the notation in 7.2 are suitable for use by OSI applications (in conjunction with the generic protecting transfer syntax defined in ITU-T Rec. X.833 | ISO/IEC 11586-4) or for other purposes, including off-line protection in local storage and non-OSI communications.

NOTE – Subclause 7.1.5 describes use of security transformations on an OSI presentation connection. Subclause 7.1.6 describes their use independently of the OSI presentation protocol.

Security transformations may constitute the primary means of providing a security service (e.g. confidentiality, integrity, data origin authentication) or they may contribute to the provision of a security service (e.g. entity authentication, access control, non-repudiation).

Figure 2 illustrates the steps involved in protecting a data item for transfer or storage.

At an encoding system, the process of deriving a transformed (protected) representation of an unprotected data item is:

a) if the unprotected item is a value of an ASN.1 type, as specified in an abstract syntax definition, encode to a bit string representation using initial encoding rules; then

b) apply the encoding process of the security transformation to the bit string representation of the unprotected item, possibly using additional local input information, to obtain a transformed item, which is a value of the ASN.1 type XformedDataType (the precise type is specified as part of the security transformation definition); then

c) encode the ASN.1 value resulting from b) (possibly as part of the process of encoding an encompassing ASN.1 value, such as the protecting transfer syntax structure defined in ITU-T Rec. X.833 | ISO/IEC 11586-4).

At a decoding system, the process of recovering the unprotected data item and/or checking for a security compromise is:

d) decode the received or retrieved transformed item, which is an ASN.1 value of type XformedDataType (this decoding process may form part of the decoding of an encompassing ASN.1 value, such as the protecting transfer syntax structure defined in ITU-T Rec. X.833 | ISO/IEC 11586-4); then

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:22 1999

e) apply the decoding process of the security transformation to the received or retrieved value, possibly using additional local input information, and generate outputs according to that decoding process (depending upon the particular transformation, outputs might include a recovered copy of the unprotected item, an indication of success/failure of signature or seal verification, and/or a copy of a signature for storing locally for later use); then

f) if an output of step e) is a recovered copy of the unprotected item, and if that item is a value of an ASN.1 type as specified in an abstract syntax definition, decode that data item using the same initial encoding rules as in step a).

Determination of the initial encoding rules in steps a) and f) is described in 7.1.4. Note that, in general, security transformations may operate upon data items other than values of ASN.1 types (e.g. arbitrary bit strings), so this encoding process is not always needed.

Determination of the encoding rules for steps c) and d) depends upon the storage or communication environment, and is independent of the particular security transformation used.

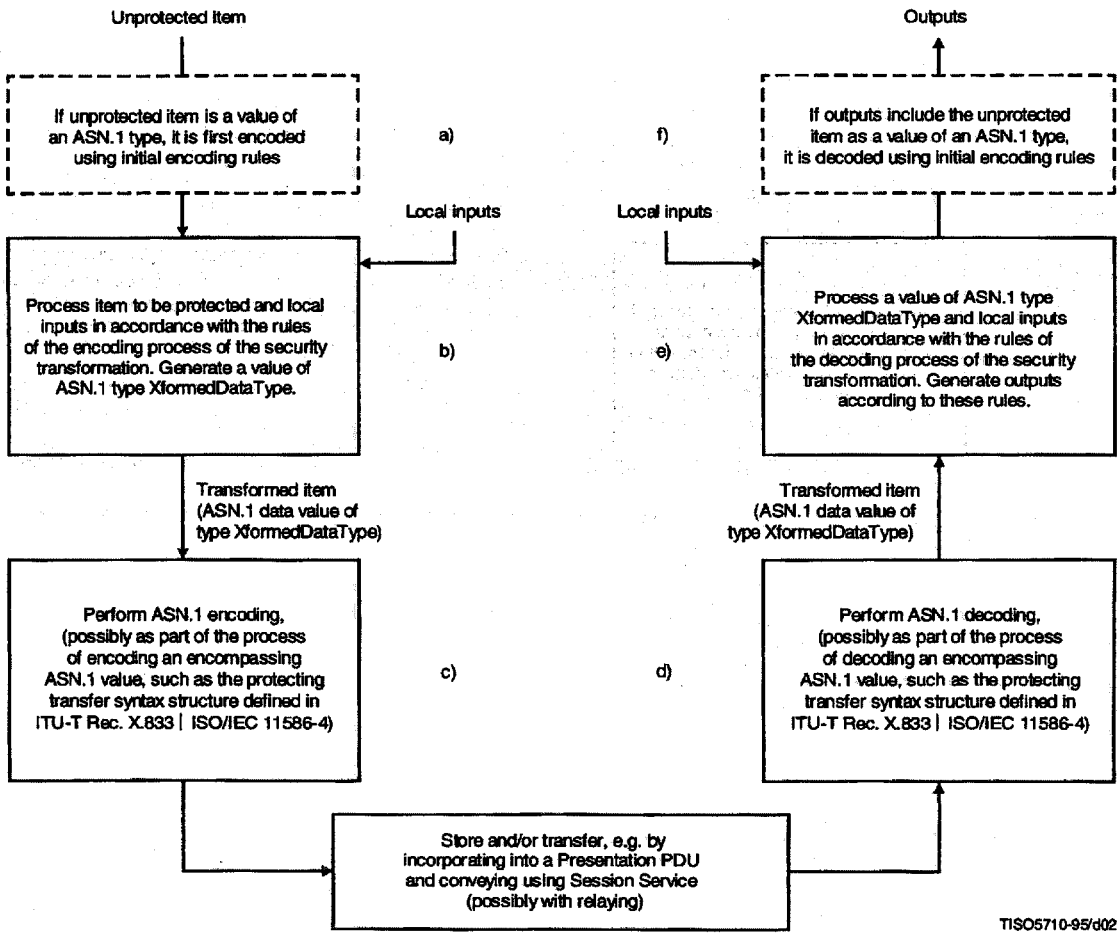


Figure 2 – Protected storage or transfer of a data item

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:23 1999

7.1.1 Architectural placement of security transformations in OSI upper layers

A security transformation operates in the context of a security association between two or more systems. There is a System Security Object (SSO) in each system, supporting such a security association. These SSOs perform the security transformation encoding/decoding processes (e.g. encipherment, digital signature generation/verification), and store necessary security state information (e.g. keys, algorithms, parameters, chaining state). The internal behaviour of such SSOs is governed by specific security transformation specifications, together with supporting specifications, e.g. for algorithms (which are outside the scope of this Recommendation I International Standard). In terms of Figure 2, the functions indicated in boxes b) and e) are modeled in SSOs.

There are also Security Communication Functions (SCFs) in the presentation entities of encoding and decoding systems. These SCFs support the communication requirements of the SSOs. In terms of Figure 2, the functions indicated in boxes a), c), d) and f) are modeled in SCFs. Definitions of SCF behaviour are contained in clause 8 of this Recommendation I International Standard, and in ITU-T Rec. X.833 I ISO/IEC 11586-4.

7.1.2 Security associations

A security transformation may be applied repeatedly to a sequence of logically-ordered data values, e.g. presentation data values transferred sequentially in one direction between two systems. The same protection is applied to each data value. Application of a security transformation to such a sequence is governed by a security association. More than one security association may exist at the same time between a pair of systems, typically providing different types of protection.

This Recommendation I International Standard addresses aspects of a security association which are relevant to upper layers communications or information storage. From the OSI upper layers perspective, a security association is a form of ASO-association.

Three kinds of security association are recognized by this Recommendation I International Standard:

a) *externally-established security association* – A security association which is established independently of instances of its use, and which has a globally-unique identifier enabling it to be referenced at the time of use. The means of establishing such a security association are not specified in this Recommendation I International Standard, and its lifetime is not restricted by provisions in this Recommendation I International Standard. The identifier of an externally established security association comprises an integer value, together with the identity of the system assigning that integer value. (The latter identity may be known implicitly, e.g. the sender or receiver of data, hence this identity need not always be carried in protocol.)

b) *single-item-bound security association* – A security association applying to a single independently-protected presentation data value which is not associated with a presentation context; attributes of the security association are indicated explicitly along with the presentation data value encoding. The lifetime of a single-item-bound security association is limited to the lifetime of the one presentation data value.

c) *presentation-context-bound security association* – A security association which is established in conjunction with the establishment of a protecting presentation context, and which applies to all presentation data values sent in one direction in that protecting presentation context; attributes of the security association are indicated explicitly along with the encoding of the first presentation data value in the protecting presentation context. This type of security association can apply only when protection is provided in conjunction with use of the OSI presentation service and protocol specifed in ITU-T Rec. X.216 I ISO/IEC 8822 and ITU-T Rec. X.226 I ISO/IEC 8823-1 respectively. The lifetime of the security association is the same as the lifetime of the corresponding protecting presentation context.

The operation of a security transformation may be governed by local security state information and/or by parameters which are transferred or stored with encoded data values. Local security state information may be maintained from one application of a security transformation to its next application, in the one security association. For example, for transformations providing integrity of the sequence of presentation data values within a security association, state information such as an integrity sequence number or a cryptographic chaining value will be retained from one application of the transformation to the next. Values of static parameters (see 7.1.3) are also retained throughout a security association.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:23 1999

7.1.3 Security transformation parameters

When using a security transformation, parameter values may need to be conveyed between the encoding and decoding functions, along with the transformed data values. Parameters are of two types:

a) *static parameters* – These parameters maintain constant values throughout a security association, and are specified by the encoder of data when or before the security transformation is first applied in a security association;

b) *dynamic parameters* – The values of these parameters may change dynamically while the transformation is in use in a security association; the encoder of the data indicates such changes within the data stream.

Examples of static parameters are:

- identifier(s) of the algorithm(s) used in a security transformation;
- if necessary, the mode of operation of an algorithm;
- the key(s), or identifier(s) of key(s), to be used with the above-mentioned algorithm(s);
- if necessary, the value(s) of initialization vector(s).

An example of a dynamic parameter is a key which is changed after a certain period of use.

Parameter values may be encoded unprotected or may themselves require protection. Unprotected parameters are conveyed in explicit fields of the protecting transfer syntax which supports the security transformation. Protected parameters are considered as inputs to the security transformation encoding process, along with the value to be protected. The security transformation rules must stipulate how these parameters are represented, how their representation is combined with the encoded abstract syntax value, and how the result is processed to generate an ASN.1 data value for transfer or storage.

NOTE – As an example of conveying protected parameters, see the definition of the GULS SIGNED Security Transformation in D.4.

Parameter data (e.g. keys) required by security transformations may also be obtained by other means, including:

- earlier Application Layer protocol exchanges (e.g. a key derivation security exchange conveyed by the SESE);
- local means (e.g. manual insertion of keys).

7.1.4 Determination of initial encoding rules

The rules for the initial encoding (and final decoding) processes (modelled in boxes a) and f) of Figure 2 are determined in one of the following ways:

a) a security transformation may provide for the conveying of an indication of initial encoding rules as a static (protected or unprotected) parameter of the security transformation;

b) as a default, every security transformation specification identifies default initial encoding rules.

NOTE – When using digital signatures for non-repudiation, the transformed item (i.e. the signed data) may need to be stored in a recipient system and/or relayed to another entity. In such circumstances, knowledge of the initial encoding rules used in computing the signature needs to be preserved. For digital signatures, it is recommended that the default encoding rules identified in the security transformation specification be used. Then, the required knowledge can be preserved by storing/relaying the security transformation identifier along with the signature.

7.1.5 Use of security transformations on an OSI presentation-connection

The OSI Presentation Layer associates a transfer syntax with each abstract syntax used. When a security transformation is employed, the transfer syntax is denoted a protecting transfer syntax.

In accordance with CCITT Rec. X.208 | ISO/IEC 8824, presentation data values may be transferred either:

a) within a negotiated presentation context; or

b) (as an option when using ASN.1 EXTERNAL or EMBEDDED PDV notation) outside a presentation context.

In both cases, a presentation data value which is to be protected is represented using a protecting transfer syntax. A protecting transfer syntax defined in accordance with ITU-T Rec. X.833 | ISO/IEC 11586-4 supports the communication of static and dynamic security transformation parameters.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:23 1999

Case a) above involves a protecting presentation context. All presentation data values transferred in one direction within a protecting presentation context are protected using the same security transformation, and are governed by the one security association. When a protecting presentation context is established (using the procedures for establishing a presentation context specified in ITU-T Rec. X.216 | ISO/IEC 8822 and ITU-T Rec. X.226 | ISO 8823-1), then the first presentation data value in each direction in that presentation context shall either:

a) reference an externally established security association; or

b) define a new presentation-context-bound security association.

When a presentation data value is encoded outside a presentation context, then the presentation data value shall either:

a) reference an externally established security association; or

b) define a new single-item-bound security association.

On an OSI presentation connection, different security associations apply to each direction of flow. These security associations may use the same security transformation but are not required to do so.

NOTE – The above restriction (i.e. that, when using the OSI presentation protocol, different security associations apply to each direction of flow) ensures that there can be no common cryptographic state variables shared between the two directions of flow. If such shared state could exist, there would be a need for complex state-maintenance protocol elements in the Presentation Layer to handle such events as Session Layer resynchronization. In practice, it is likely that the separate security associations for the two directions will have common attributes derived from one encompassing security association.

7.1.6 Use of security transformations independently of OSI presentation protocol

Security transformations may be used independently of the OSI presentation protocol, e.g. for protection in storage. Concepts and procedures described in 7.1.2 through 7.1.5 apply, with the following restrictions.

All protected presentation data values are represented outside presentation contexts.

A single-item-bound security association or externally established security association may be employed. Presentation-context-bound security associations cannot apply. Where the protected information is not being exchanged but is being protected only for use by the originator, then security transformations may also be employed without security associations.

If an externally established security association is employed, the lifetime of the externally established security association must span the storage lifetime of the protected data.

7.2 Notation for specifying security transformations

Security transformation specifications include specifications of data items needing to be recognized by the protecting transfer syntax structure. For this purpose, the following ASN.1 information object class definition (see ITU-T Rec. 681 | ISO/IEC 8824-2) is provided:

```
SECURITY-TRANSFORMATION ::= CLASS
-- This information object class definition is for use when
-- specifying a particular instance of a security transformation.
{
  &sT-Identifier     OBJECT IDENTIFIER UNIQUE,
  -- Identifier to be used in signalling the application
  -- of the particular security transformation
  &initialEncodingRules OBJECT IDENTIFIER
     DEFAULT {joint-iso-ccitt asn1 (1) ber-derived (2)
     canonical-encoding (0)},
  -- Default initial encoding rules to generate a bit
  -- string prior to applying the encoding process of a
  -- security transformation.
  &StaticUnprotectedParm OPTIONAL,
  -- ASN.1 type for conveying static unprotected parameters
  &DynamicUnprotectedParm OPTIONAL,
  -- ASN.1 type for conveying dynamic unprotected parameters
  &XformedDataType,
  -- ASN.1 type of the ASN.1 value produced by the security
  -- transformations encoding process
  &QualifierType OPTIONAL
  -- &QualifierType specifies the ASN.1 type of the qualifier
  -- parameter used with the PROTECTED-Q notation.
}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:24 1999

WITH SYNTAX
-- The following syntax is used to specify a particular security
-- transformation.

```
{
  IDENTIFIER                    &sT-Identifier
  [ INITIAL-ENCODING-RULES      &initialEncodingRules ]
  [ STATIC-UNPROT-PARM          &StaticUnprotectedParm ]
  [ DYNAMIC-UNPROT-PARM         &DynamicUnprotectedParm ]
  XFORMED-DATA-TYPE             &XformedDataType
  [ QUALIFIER-TYPE              &QualifierType ]
}
```

Examples of the use of this notation are given in Annex D.

The security transformation specification needs to also specify the following details (although no formal notation to support such specification is provided in this Recommendation | International Standard):

- *Encoding process* – A description of the transformation process applied, at the encoding end, to the unprotected item and the transferred protected parameters in order to generate the resultant transformed value (which is an ASN.1 value of type &XformedDataType).
- *Encoding process local inputs* – A list of locally-derived inputs to the encoding process.
- *Decoding process* – A description of the transformation process applied, at the decoding end, to the received or retrieved transformed value (which is of type &XformedDataType) in order to generate the resultant unprotected data bit string (if any) and the values of the transferred protected parameters.
- *Decoding process local inputs* – A list of locally-derived inputs to the decoding process.
- *Decoding process outputs* – A list of outputs of the decoding process (may or may not include a recovered value of the unprotected item).
- *Parameters* – A description of the semantic significance of all parameters, default values for parameters, and the circumstances under which dynamic parameter changes should occur.
- *Transformation qualifiers* – A description of rules applying to invoker-specified transformation qualifiers (if any) applying to this transformation.
- *Errors* – A description of error conditions which may be detected during the decoding process.

8 Abstract syntax notation for selective field protection

The following abstract syntax notation is for the specification of the abstract protection requirements for a selected ASN.1 data type. The protection required is mapped to one of a set of security transformations which provide (at an abstract level) the required form of protection. Some security transformations accept input qualifiers to control the operation of the required protection, for example, the identifier for the security association for which the protection is to be applied. For these cases, an extension of the basic notation is defined, to enable qualifiers to be specified by the user of the notation.

This clause specifies:

a) the basic protected abstract syntax notation, for specifying abstract protection requirements for a selected field in an abstract syntax specification;

b) the qualified protected abstract syntax notation, for specifying abstract protection requirements, along with an associated qualifier, for a selected field in an abstract syntax specification;

c) the protection mapping notation, for specifying the possible mappings to one or more security transformations which provide the required protection.

8.1 Basic notation

In order to assist the writer of an abstract syntax in indicating selective field protection requirements, the following ASN.1 parameterized type (see ITU-T Rec. X.683 | ISO/IEC 8824-4) is defined:

```
PROTECTED {BaseType, PROTECTION-MAPPING: protectionReqd} ::=
CHOICE
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:24 1999

```
{
  dirEncrypt BIT STRING (CONSTRAINED BY {BaseType
        -- dirEncrypt is for use only with the
        -- dirEncryptedTransformation,
        -- and generates the same encoding as the
        -- X.509/9594-8 ENCRYPTED type--}),
  dirSign   SEQUENCE
        {
          baseType BaseType  OPTIONAL,
            -- must be present for dirSignedTransformation
            -- and must be omitted for
            -- dirSignatureTransformation
          algorithmId AlgorithmIdentifier,
          encipheredHash BIT STRING (CONSTRAINED BY
              {BaseType -- contains enciphered hash
                        -- of a value of BaseType --})
        }
        -- dirSign is for use only with the
        -- dirSignedTransformation or
        -- dirSignatureTransformation, and generates
        -- the same encoding as the corresponding
        -- X.509/9594-8 SIGNED or SIGNATURE type--,
  noTransform [0] BaseType,
        -- noTransform invokes no security transformation.
        -- Subject to security policy, noTransform may be used
        -- if adequate protection is provided by lower layers
        -- and any application relays through which the data
        -- may pass are trusted to maintain the required
        -- protection. This alternative may only be used
        -- if protectionReqd.&bypassPermitted is TRUE,
  direct    [1] SyntaxStructure
          {{protectionReqd.&SecurityTransformation}},
        -- direct generates a protecting transfer syntax
        -- value, which is encoded using the same encoding
        -- rules as the surrounding ASN.1 (The type
        -- SyntaxStructure is imported from Rec. X.833 |
        -- ISO/IEC 11586-4),
  embedded    [2] EMBEDDED PDV (WITH COMPONENTS {
              identification (WITH COMPONENTS {
                    presentation-context-id,
                    context-negotiation (WITH COMPONENTS {
                          transfer-syntax (CONSTRAINED BY
                              {OBJECT IDENTIFIER :
                              protectionReqd.&protTransferSyntax})}),
                    transfer-syntax (CONSTRAINED BY
                              {OBJECT IDENTIFIER :
                              protectionReqd.&protTransferSyntax})}),
        data-value (WITH COMPONENTS {notation (BaseType)})
        -- The data value encoded is a value of type BaseType
        })
}
-- BaseType is the type to be protected, and protectionReqd is an ASN.1
-- object of class PROTECTION-MAPPING. The use of PROTECTED requires
-- the importation into the user's module of the PROTECTED parameterized
-- type, together with the necessary PROTECTION-MAPPING object
-- definition.
```

The PROTECTION-MAPPING object class and its significance are discussed in 8.3. The set of allowable objects for "protectionReqd" will vary in different abstract syntax specifications, dependent upon the range of distinct transformations required. The mappings from PROTECTION-MAPPING objects to transformations are contained in a set of PROTECTION-MAPPING object definitions. This set of definitions may be specified in a separate ASN.1 module from the (mechanism-independent) abstract syntax specification and from the (application-independent) transformation definition.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:25 1999

The various alternatives in the CHOICE are made available for use in different circumstances as follows:

- *dirEncrypt and dirSign* – These alternatives generate the &XformedDataType of the security transformation used. These alternatives are made available to provide a means whereby the PROTECTED notation can generate identical bit-encodings to the ENCRYPTED, SIGNED, and SIGNATURE parameterized types defined in ITU-T Rec. X.509 | ISO/IEC 9594-8.
- *noTransform* – This alternative uses no security transformation. It is permitted if the protection mapping in use (see 8.3 and 8.4) indicates &bypassPermitted = TRUE. The item is encoded in its unprotected form. According to security policy, noTransform may be used if adequate protection is provided by lower layers and any application relays through which the data may pass are trusted to maintain the required protection.
- *direct* – This alternative directly imports a protecting transfer syntax value, as defined in ITU-T Rec. X.833 | ISO/IEC 11586-4, into the encompassing ASN.1 specification. It supports use of an externally established security association or single-item-bound security association. It does not permit use of a negotiated presentation context. The encoding rules used for encoding the protecting transfer syntax structure [modelled in boxes c) and d) of Figure 2 in 7.1] are forced to be the same as those used for the ASN.1 type encompassing the PROTECTED notation.
- *embedded* – This alternative provides the greatest flexibility, including the ability to associate protection with a negotiated presentation context and the ability to use a different protecting transfer syntax to that defined in ITU-T Rec. X.833 | ISO/IEC 11586-4.

NOTE – It is recommended that use of these options be selected as follows:

a) Use the direct option if none of b), c) or d) applies.

b) When requiring bit-compatibility with the ENCRYPTED, SIGNED, or SIGNATURE parameterized types for backward-compatibility reasons, use the dirEncrypt or dirSign option as applicable.

c) When the protection mapping in use indicates &bypassPermitted = TRUE and when security policy permits, use the noTransform option.

d) When there is a requirement to associate protection with a negotiated presentation context, use the embedded option.

Error conditions may be detected in the decoding system in processing a value in a protected field. The ASN.1 exception handling notation defined in ITU-T Rec. X.682 | ISO/IEC 8824-3 can be used to deal with such error conditions.

Examples of the use of this notation appear in I.1.

8.2 Notation with transformation qualifier

As an alternative to the PROTECTED notation described in 8.1, the PROTECTED-Q notation allows its user to additionally provide a qualifier parameter. These qualifier parameters are used for one or both of the following purposes:

a) to identify a specific externally established security association;

b) to provide one or more parameters for use by the security transformation, e.g. algorithm, mode of operation, and/or key identifiers.

NOTE – Some algorithm identifiers may imply a particular mode of operation. In other cases, the mode of operation may be specified as an additional parameter.

Multiple qualifiers may be specified using an appropriate ASN.1 SEQUENCE or SET type. Within the encoding system, a qualifier is used by local system functions to determine the appropriate SSO and/or to convey a parameter to that SSO. A qualifier conveyed to a SSO needs to be compatible with the security transformation in use, as stated in the security transformation specification. When the specified protection mapping permits a choice of security transformations, the one selected for any instance of use must be one with a &QualifierType consistent with the value specified by the user of the PROTECTED-Q notation. The value of the qualifier may be (but is not necessarily) conveyed to the decoding system within the protecting transfer syntax (e.g. as the externally established security association identifier, or as a security transformation parameter).

The following ASN.1 parameterized type (see ITU-T Rec. X.683 | ISO/IEC 8824-4) is defined:

```
PROTECTED-Q {BaseType, PROTECTION-MAPPING: protectionReqd,
  PROTECTION-MAPPING.&SecurityTransformation.&QualifierType: qualifier} ::=
    PROTECTED {BaseType, protectionReqd} (CONSTRAINED BY
    {PROTECTION-MAPPING.&SecurityTransformation.&QualifierType: qualifier
        -- The value of qualifier must be made available to
        -- the security transformation used
      })
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:25 1999

-- BaseType is the type to be protected, and protectionReqd is an object of class
-- PROTECTION-MAPPING. The use of PROTECTED requires the importation into the user's
-- module of the PROTECTED parameterized type, together with the necessary
-- PROTECTION-MAPPING object definition.

Examples of the use of this notation are given in I.2 and I.3.

8.3 Mapping protection requirements to security transformations

A protection mapping relates a protection requirement, identified by name in an abstract syntax specification, to a specific transformation to be used to satisfy that requirement. This concept is introduced to allow such mappings to be specified separately from the main abstract syntax specification, which can then be mechanism-independent. For the protection named in an abstract syntax, the actual transformation used may be different in different application-contexts.

A protection mapping may constrain selection of a security transformation in the following ways:

- by giving a list of security transformations; a particular security transformation will be selected from this list at the time of use, based on local security policy and other local system considerations;
- by stating specialized selection rules.

Examples of protection mappings, which are defined fully in Annex E, are:

- *confidentiality* – Confidentiality-protect data through encipherment/decipherment, but allow bypassing of the encipherment/decipherment if security policy so dictates.
- *encrypted* – Perform encipherment/decipherment with an unspecified type of algorithm.
- *signed* – Generate/verify a digital signature attached to the signed data.
- *signature* – Generate/verify a digital signature for transfer separately from the signed data.

Other protection mappings are possible, e.g. for mapping specifically to transformations for public-key encipherment, symmetric encipherment, sealing, hashing, or one-way encipherment.

8.4 Notation for specifying protection mappings

For defining specific protection mappings, the following ASN.1 information object class definition (see ITU-T Rec. X.681 | ISO/IEC 8824-2) is provided:

```
PROTECTION-MAPPING ::= CLASS
{
   &SecurityTransformation SECURITY-TRANSFORMATION,
   -- &SecurityTransformation specifies an ASN.1 object set of the SECURITY-TRANSFORMATION class.
   -- Use of the particular protection mapping implies use of one of the specified transformations,
   -- with the choice being left to the encoding system. Rules for selecting between these security
   -- transformations may be specified in comments.
   &protTransferSyntax OBJECT IDENTIFIER
         DEFAULT {joint-iso-ccitt genericULS (20)
         generalTransferSyntax (2)},
   -- Identifies the particular protecting transfer syntax to be used in an EMDEDDED PDV
   -- encoding for the embedded option.
   &bypassPermitted BOOLEAN DEFAULT FALSE
   -- Indicates if bypassing of protection is permitted
}
WITH SYNTAX
{
   SECURITY-TRANSFORMATION          &SecurityTransformation
   [ PROTECTING-TRANSFER-SYNTAX     &protTransferSyntax ]
   [ BYPASS-PERMITTED               &bypassPermitted ]
}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:25 1999

9 Conformance

A system claiming conformance to this Recommendation I International Standard does so with respect to use of GULS security exchanges or security transformations specified in Annexes C and D:

a) When using any of the security exchanges specified in Annex C, as identified by the ASN.1 object identifier for the "GulsSecurityExchanges" module given in Annex C, the system shall support the applicable ASN.1 and any associated stipulations in Annex C.

b) When using any of the security transformations specified in Annex D, as identified by the ASN.1 object identifier for the "GulsSecurityTransformations" module given in Annex D, the system shall support the applicable ASN.1 and any associated stipulations in Annex D.

Particular static and dynamic conformance requirements are stated in the relevant subclauses of Annexes C and D.

Implementation of the constructs defined in Annexes C, D, and E is at the option of the user of this Recommendation I International Standard, and is not a mandatory conformance requirement.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:26 1999

Annex A

ASN.1 definitions

(This annex forms an integral part of this Recommendation | International Standard)

The following ASN.1 module provides the definitive ASN.1 specifications for the body of this Recommendation | International Standard.

```
Notation {joint-iso-ccitt genericULS (20)
        modules (1) notation (1)}
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

-- EXPORTS All --

IMPORTS
  -- From Directory Standards: --
  informationFramework, selectedAttributeTypes,
  authenticationFramework
     FROM UsefulDefinitions {joint-iso-ccitt ds (5) module (1)
           usefulDefinitions (0) 2}
  Name
     FROM InformationFramework          informationFramework
  UniqueIdentifier
     FROM SelectedAttributeTypes        selectedAttributeTypes
  AlgorithmIdentifier
     FROM AuthenticationFramework       authenticationFramework

  -- From Other GULS Modules: --
  genericProtectingTransferSyntax
     FROM ObjectIdentifiers {joint-iso-ccitt genericULS (20)
     modules (1) objectIdentifiers (0)}
  SyntaxStructure { }
     FROM GenericProtectingTransferSyntax
           genericProtectingTransferSyntax;

-- ************************************************ --
-- Notation for security identity and SA-identifiers --
-- ************************************************ --

-- Values of the SecurityIdentity type are used to identify entities
-- which assign externally-established security association identifiers,
-- and for other security-related purposes requiring globally-unique
-- identifiers.

SecurityIdentity ::= CHOICE
{
  directoryName        Name,
  objectIdentifier     OBJECT IDENTIFIER
}
ExternalSAID ::= SEQUENCE
{
  localSAID            INTEGER,
  assignerIdentity     SecurityIdentity   OPTIONAL
  -- Identity of the system which assigned the integer value
}

-- ****************************************** --
-- Notation for specifying security exchanges --
-- ****************************************** --

SECURITY-EXCHANGE ::= CLASS
-- This information object class definition is for use when
-- specifying a particular instance of a security exchange.
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:26 1999

```
{
  &SE-Items        SEC-EXCHG-ITEM,
  -- This is an ASN.1 information object set, comprising a set
  -- of security exchange items
  &sE-Identifier   Identifier        UNIQUE
  -- A local or global identifier for the particular security
  -- exchange
}
WITH SYNTAX
-- The following syntax is used to specify a particular security
-- exchange.
{
  SE-ITEMS         &SE-Items
  IDENTIFIER       &sE-Identifier
}
Identifier ::= CHOICE
{
  local            INTEGER,
  global           OBJECT IDENTIFIER
}

SEC-EXCHG-ITEM ::= CLASS
{
  &ItemType,
      -- ASN.1 type for this exchange item
  &itemId          INTEGER,
      -- Identifier for this item, e.g. 1, 2, 3, ..
  &Errors          SE-ERROR          OPTIONAL
      -- Optional list of errors which may result from
      -- transfer of this item
}
WITH SYNTAX
{
  ITEM-TYPE        &ItemType
  ITEM-ID          &itemId
  [ERRORS          &Errors]
}

SE-ERROR ::= CLASS
{
  &ParameterType     OPTIONAL,
  -- ASN.1 type of a parameter to accompany the signalling
  -- of the error condition back to the sender of the SEI
  &errorCode       Identifier        UNIQUE
  -- An identifier used in signalling the error condition
  -- back to the sender of the SEI
}
WITH SYNTAX
{
  [PARAMETER         &ParameterType]
  ERROR-CODE         &errorCode
}


-- ************************************************ --
-- Notation for specifying security transformations --
-- ************************************************ --

SECURITY-TRANSFORMATION ::= CLASS
-- This information object class definition is for use when
-- specifying a particular instance of a security transformation.
{
  &sT-Identifier     OBJECT IDENTIFIER UNIQUE,
  -- Identifier to be used in signalling the application
  -- of the particular security transformation
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:26 1999

```
  &initialEncodingRules OBJECT IDENTIFIER
     DEFAULT {joint-iso-ccitt asn1 (1) ber-derived (2)
     canonical-encoding (0)},
  -- Default initial encoding rules to generate a bit
  -- string prior to applying the encoding process of a
  -- security transformation.
  &StaticUnprotectedParm OPTIONAL,
  -- ASN.1 type for conveying static unprotected parameters
  &DynamicUnprotectedParm OPTIONAL,
  -- ASN.1 type for conveying dynamic unprotected parameters
  &XformedDataType,
  -- ASN.1 type of the ASN.1 value produced by the security
  -- transformations encoding process
  &QualifierType  OPTIONAL
  -- &QualifierType specifies the ASN.1 type of the qualifier
  -- parameter used with the PROTECTED-Q notation.

}
WITH SYNTAX
-- The following syntax is used to specify a particular security
-- transformation.
{
  IDENTIFIER                          &sT-Identifier
  [ INITIAL-ENCODING-RULES            &initialEncodingRules ]
  [ STATIC-UNPROT-PARM                &StaticUnprotectedParm ]
  [ DYNAMIC-UNPROT-PARM               &DynamicUnprotectedParm ]
  XFORMED-DATA-TYPE                   &XformedDataType
  [ QUALIFIER-TYPE                    &QualifierType ]
}


-- ************************************************** --
-- Notation for specifying selective field protection --
-- ************************************************** --

PROTECTED {BaseType, PROTECTION-MAPPING: protectionReqd} ::=
CHOICE
{
  dirEncrypt BIT STRING (CONSTRAINED BY {BaseType
         -- dirEncrypt is for use only with the
         -- dirEncryptedTransformation,
         -- and generates the same encoding as the
         -- X.509/9594-8 ENCRYPTED type--}),
  dirSign   SEQUENCE
        {
           baseType BaseType  OPTIONAL,
             -- must be present for dirSignedTransformation
             -- and must be omitted for
             -- dirSignatureTransformation
           algorithmId AlgorithmIdentifier,
           encipheredHash BIT STRING (CONSTRAINED BY
                {BaseType -- contains enciphered hash
                          -- of a value of BaseType --})
        }
         -- dirSign is for use only with the
         -- dirSignedTransformation or
         -- dirSignatureTransformation, and generates
         -- the same encoding as the corresponding
         -- X.509/9594-8 SIGNED or SIGNATURE type--,
  noTransform [0] BaseType,
         -- noTransform invokes no security transformation.
         -- Subject to security policy, noTransform may be used
         -- if adequate protection is provided by lower layers
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:27 1999

```
        -- and any application relays through which the data
        -- may pass are trusted to maintain the required
        -- protection. This alternative may only be used
        -- if protectionReqd.&bypassPermitted is TRUE,
  direct   [1] SyntaxStructure
            {{protectionReqd.&SecurityTransformation}},
        -- direct generates a protecting transfer syntax
        -- value, which is encoded using the same encoding
        -- rules as the surrounding ASN.1  (The type
        -- SyntaxStructure is imported from Rec. X.833 |
        -- ISO/IEC 11586-3)
  embedded  [2] EMBEDDED PDV (WITH COMPONENTS {
              identification (WITH COMPONENTS {
                    presentation-context-id,
                    context-negotiation (WITH COMPONENTS {
                          transfer-syntax (CONSTRAINED BY
                            {OBJECT IDENTIFIER :
                            protectionReqd.&protTransferSyntax})}),
                    transfer-syntax (CONSTRAINED BY
                          {OBJECT IDENTIFIER :
                          protectionReqd.&protTransferSyntax})}),
        data-value (WITH COMPONENTS {notation (BaseType)})
        -- The data value encoded is a value of type BaseType
        })
}
-- BaseType is the type to be protected, and protectionReqd is an ASN.1
-- object of class PROTECTION-MAPPING. The use of PROTECTED requires
-- the importation into the user's module of the PROTECTED parameterized
-- type, together with the necessary PROTECTION-MAPPING object
-- definition.

PROTECTED-Q {BaseType, PROTECTION-MAPPING: protectionReqd,
  PROTECTION-MAPPING.&SecurityTransformation.&QualifierType: qualifier} ::=
    PROTECTED {BaseType, protectionReqd} (CONSTRAINED BY
    {PROTECTION-MAPPING.&SecurityTransformation.&QualifierType: qualifier
        -- The value of qualifier must be made available to
        -- the security transformation used
        })
  -- BaseType is the type to be protected, and protectionReqd is an
  -- object of class PROTECTION-MAPPING. The use of PROTECTED requires
  -- the importation into the user's module of the PROTECTED parameterized
  -- type, together with the necessary PROTECTION-MAPPING object
  -- definition.


  -- ******************************************** --
  -- Notation for specifying protection mappings --
  -- ******************************************** --

  PROTECTION-MAPPING ::= CLASS
  {
      &SecurityTransformation SECURITY-TRANSFORMATION,
      -- &SecurityTransformation specifies an ASN.1 object set of the
      -- SECURITY-TRANSFORMATION class. Use of the particular
      -- protection mapping implies use of one of the specified
      -- transformations, with the choice being left to the
      -- encoding system. Rules for selecting between these security
      -- transformations may be specified in comments.
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:27 1999

```
    &protTransferSyntax OBJECT IDENTIFIER
      DEFAULT {joint-iso-ccitt genericULS (20)
      generalTransferSyntax (2)},
    -- Identifies the particular protecting transfer syntax to
    -- be used in an EMDEDDED PDV encoding for the embedded
    -- option.
    &bypassPermitted BOOLEAN DEFAULT FALSE
    -- Indicates if bypassing of protection is permitted
}
WITH SYNTAX
{
    SECURITY-TRANSFORMATION          &SecurityTransformation
    [ PROTECTING-TRANSFER-SYNTAX     &protTransferSyntax ]
    [ BYPASS-PERMITTED               &bypassPermitted ]
}

END
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:27 1999

Annex B

Registration of security exchanges and security transformations

(This annex forms an integral part of this Recommendation | International Standard)

B.1 Introduction

The identification of security exchanges and security transformations for use in accordance with various parts of the Generic Upper Layers Security Specifications requires unambiguous naming of such information objects. This annex specifies the procedures for allocating such names.

B.2 Registration procedures

This subclause specifies registration procedures for security exchanges and security transformations specified:

a) in ITU-T Recommendations | International Standards; or

b) by some organization which has need.

B.2.1 Registration in CCITT Recommendations | International Standards

In some cases the names of security exchanges or security transformations are specified in ITU-T Recommendations | International Standards referencing this ITU-T Recommendation | International Standard. The name shall be defined in accordance with ITU-T X.660 | ISO/IEC 9834-1. International Registration Authorities covering these types of information objects are not currently intended.

The referencing ITU-T Recommendation | International Standard will assign a name in accordance with CCITT Rec. X.660 | ISO/IEC 9834-1, but need not reference CCITT Rec. X.660 | ISO/IEC 9834-1.

B.2.2 Registration by some organization which has a need

The assignment of names for security exchange or security transformation specifications shall be in accordance with the general procedures and be of the form specified in CCITT Rec. X.660 | ISO/IEC 9834-1.

Organizations wishing to assign such names shall find an appropriate superior in the naming tree of CCITT Rec. X.660 | ISO/IEC 9834-1 and request that an arc be assigned to them.

NOTE – This includes ISO/IEC National Bodies, organizations with International Code Designators asigned in accordance with ISO 6523, telecommunications administrations and Registered Operating Agencies (ROAs).

B.3 Other relevant registers

Definitions of security transformations may, but are not required to, make use of register entries in the Register of Cryptographic Algorithms established in accordance with ISO/IEC 9979, for possible use as security transformation parameters.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:28 1999

Annex C

Security exchange specifications

(This annex forms an integral part of this Recommendation | International Standard)

Security exchanges may be defined in ITU-T Recommendations | International Standards or may be defined outside Recommendations | International Standards and registered by any organization able to assign object identifiers. Security exchange definitions should be made as broadly applicable as possible so they can be reused in multiple applications. This annex defines some security exchanges which are considered to be generally useful. There is no implied requirement for applications or implementations thereof to employ the specific security exchanges defined here, in preference to other security exchanges.

C.1 Directory Authentication Exchange (One-way)

The "Directory Authentication Exchange (One-way)" security exchange is based on the authentication exchange used in the Directory protocol (ITU-T Rec. X.511 | ISO/IEC 9594-3) for either simple or strong unilateral entity authentication. For details of the credentials data item see ITU-T Rec. X.511 | ISO/IEC 9594-3, and for a description of the associated semantics see ITU-T Rec. X.509 | ISO/IEC 9594-8.

```
dirAuthenticationOneWay  SECURITY-EXCHANGE ::=
{
  SE-ITEMS    {credentials}
  IDENTIFIER  global : {securityExchanges dir-authent-one-way (1)}
}
credentials    SEC-EXCHG-ITEM  ::=
{
  ITEM-TYPE   DirectoryAbstractService.Credentials
  ITEM-ID     1
}
```

This security exchange involves a single SEI transferred from claimant to verifier. No errors are defined; signalling of error conditions is left to other application protocol.

C.1.1 Conformance

An implementation claiming conformance to this security exchange definition shall satisfy the following conformance requirements:

- *Statement Requirements* – An implementor shall state whether the implementation acts as an initiator (initiates the security exchange), responder (responds to an initiation from another system), or both.
- *Static Requirements* – An implementation that acts as an initiator shall be able to generate the following security exchange item: credentials. An implementation that acts as a responder shall be able to process the following security exchange item: credentials.
- *Dynamic Requirements* – An implementation must implement the applicable procedures described in this annex, ITU-T Rec. X.511 | ISO/IEC 9594-3 and ITU-T Rec. X.509 | ISO/IEC 9594-8.

C.2 Directory Authentication Exchange (Two-way)

The "Directory Authentication Exchange (Two-way)" security exchange is based on the authentication exchange used in the Directory protocol (ITU-T Rec. X.511 | ISO/IEC 9594-3) for either simple or strong mutual entity authentication. For details of the credentials data item see ITU-T Rec. X.511 | ISO/IEC 9594-3, and for a description of the associated semantics see ITU-T Rec. X.509 | ISO/IEC 9594-8.

```
dirAuthenticationTwoWay  SECURITY-EXCHANGE ::=
{
  SE-ITEMS    {initiatorCredentials | responderCredentials}
  IDENTIFIER  global : {securityExchanges dir-authent-two-way (2)}
}
initiatorCredentials    SEC-EXCHG-ITEM  ::=
{
  ITEM-TYPE   DirectoryAbstractService.Credentials
  ITEM-ID     1
  ERRORS  {authenticationFailure}
}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:28 1999

```
responderCredentials    SEC-EXCHG-ITEM  ::=
{
  ITEM-TYPE     DirectoryAbstractService.Credentials
  ITEM-ID       2
}
authenticationFailure   SE-ERROR ::=
{
  PARAMETER     DirectoryAbstractService.SecurityProblem
  ERROR-CODE    local : 1
}
```

This security exchange involves two security exchange items, the first transferred from initiator to responder. If after the first transfer an error is detected, the responder should abort the security exchange. It may optionally use the authenticationFailure error code or may abort without specifying error reason. If no error is detected after the first transfer, the responderCredentials SEI is transferred from responder to initiator. No errors are defined for the second SEI transfer; signalling of error conditions is left to other application protocol.

C.2.1 Conformance

An implementation claiming conformance to this security exchange definition shall satisfy the following conformance requirements:

- *Statement Requirements* – An implementor shall state whether the implementation acts as an initiator (initiates the security exchange), responder (responds to an initiation from another system), or both.
- *Static Requirements* – An implementation that acts as an initiator shall be able to generate the following security exchange item: initiatorCredentials, and shall be able to process the following security exchange item: responderCredentials. An implementation that acts as a responder shall be able to generate the following security exchange item: responderCredentials, and shall be able to process the following security exchange item: initiatorCredentials.
- *Dynamic Requirements* – An implementation must implement the applicable procedures described in this annex, ITU-T Rec. X.511 | ISO/IEC 9594-3 and ITU-T Rec. X.509 | ISO/IEC 9594-8.

C.3 Simple Negotiation Security Exchange

An application context may include support for more than one security exchange to provide the same security services through different protocols or security mechanisms. Support of alternate security exchanges or security mechanisms allows interoperation with peers that implement any of the alternatives.

To determine the security exchanges to be employed at the time of use, the Negotiation-SE is provided. The Negotiation-SE, an object of class SECURITY-EXCHANGE, is used to negotiate particular security exchanges; this information object consists of one or more security exchange identifiers. The Negotiation-SE is used by the initiating application to propose one or more security exchanges, and it is used by the responding application to indicate which of the proposed choices will be employed in subsequent operations. The Negotiation-SE may be used at any time to change the security exchanges in use.

Application-contexts that require negotiations must specify the use of the negotiation-SE.

The negotiation-SE consists of two SEIs, "offeredIds" and "acceptedIds", as shown below.

```
simpleNegotiationSE SECURITY-EXCHANGE  ::=
{
  SE-ITEMS    {offeredIds | acceptedIds}
  IDENTIFIER  global : {securityExchanges simple-negotiation-se (3)}
}
offeredIds      SEC-EXCHG-ITEM    ::=
{
  ITEM-TYPE   Negotiation-SEI
  ITEM-ID     1
}
acceptedIds     SEC-EXCHG-ITEM    ::=
{
  ITEM-TYPE   Negotiation-SEI
  ITEM-ID     2
}

Negotiation-SEI ::= SEQUENCE OF OBJECT IDENTIFIER
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:28 1999

C.3.1 Conformance

An implementation claiming conformance to this security exchange definition shall satisfy the following conformance requirements:

- *Statement Requirements* – An implementor shall state whether the implementation acts as an initiator (initiates the security exchange), responder (responds to an initiation from another system), or both.
- *Static Requirements* – An implementation that acts as an initiator shall be able to generate the following security exchange item: offeredIds, and shall be able to process the following security exchange item: acceptedIds. An implementation that acts as a responder shall be able to generate the following security exchange item: acceptedIds, and shall be able to process the following security exchange item: offeredIds.
- *Dynamic Requirements* – An implementation must implement the applicable procedures described in this annex.

C.4 Definitive ASN.1 specification

```
GulsSecurityExchanges {joint-iso-ccitt genericULS (20)
      modules (1) gulsSecurityExchanges (2)}
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

-- EXPORTS All --

IMPORTS
  securityExchanges, notation
    FROM ObjectIdentifiers {joint-iso-ccitt genericULS (20)
    modules (1) objectIdentifiers (0)}
  SECURITY-EXCHANGE, SEC-EXCHG-ITEM, SE-ERROR
    FROM Notation notation
  Credentials, SecurityProblem
    FROM DirectoryAbstractService {joint-iso-ccitt ds (5)
    module (1) directoryAbstractService (2) 2};


-- ******************************************* --
-- Directory Authentication Exchange (One-way) --
-- ******************************************* --

dirAuthenticationOneWay  SECURITY-EXCHANGE ::=
{
  SE-ITEMS     {credentials}
  IDENTIFIER  global : {securityExchanges dir-authent-one-way (1)}
}
credentials      SEC-EXCHG-ITEM  ::=
{
  ITEM-TYPE   DirectoryAbstractService.Credentials
  ITEM-ID      1
}


-- ******************************************* --
-- Directory Authentication Exchange (Two-way) --
-- ******************************************* --

dirAuthenticationTwoWay  SECURITY-EXCHANGE ::=
{
  SE-ITEMS     {initiatorCredentials | responderCredentials}
  IDENTIFIER  global : {securityExchanges dir-authent-two-way (2)}
}
initiatorCredentials          SEC-EXCHG-ITEM  ::=
{
  ITEM-TYPE   DirectoryAbstractService.Credentials
  ITEM-ID      1
  ERRORS  {authenticationFailure}
}
responderCredentials          SEC-EXCHG-ITEM  ::=
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:29 1999

```
{
  ITEM-TYPE   DirectoryAbstractService.Credentials
  ITEM-ID     2
}
authenticationFailure    SE-ERROR ::=
{
  PARAMETER   DirectoryAbstractService.SecurityProblem
  ERROR-CODE  local : 1
}

-- ************************** --
-- Simple Negotiation Exchange --
-- ************************** --

simpleNegotiationSE  SECURITY-EXCHANGE  ::=
{
  SE-ITEMS     {offeredIds | acceptedIds }
  IDENTIFIER   global : {securityExchanges simple-negotiation-se (3)}
}
offeredIds       SEC-EXCHG-ITEM     ::=
{
  ITEM-TYPE   Negotiation-SEI
  ITEM-ID     1
}
acceptedIds     SEC-EXCHG-ITEM     ::=
{
  ITEM-TYPE   Negotiation-SEI
  ITEM-ID     2
}

Negotiation-SEI ::= SEQUENCE OF OBJECT IDENTIFIER

END
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:29 1999

Annex D

Security transformation specifications

(This annex forms an integral part of this Recommendation | International Standard)

Security transformations may be defined in ITU-T Recommendations | International Standards or may be defined outside Recommendations | International Standards and registered by any organization in accordance with Annex B. Security transformation definitions should be made as broadly applicable as possible so they can be re-used in multiple applications. This annex defines some security transformations which are considered to be generally useful. There is no implied requirement for applications or implementations thereof to employ the specific security transformations defined here, in preference to other security transformations.

D.1 Directory ENCRYPTED security transformation

The Directory Encrypted security transformation is functionally equivalent to the ENCRYPTED parameterized type defined in ITU-T Rec. X.509 | ISO/IEC 9594-8. It provides for encipherment and decipherment.

```
dirEncryptedTransformation SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations dir-encrypted (1) }
  -- This transformation transforms a string of octets to a
  -- new bit string using an encipherment process.
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber (1)}
  XFORMED-DATA-TYPE BIT STRING
}
```

D.1.1 Other details

Encoding process:	An encipherment process, based on any chosen algorithm.
Encoding process local inputs:	Algorithm, algorithm parameters, encipherment key information.
Decoding process:	A decipherment process, based on the same algorithm.
Decoding process local inputs:	Algorithm, algorithm parameters, decipherment key information.
Decoding process outputs:	Recovered item to be protected, as a value of an ASN.1 type.
Parameters:	None.
Transformation qualifiers:	None.
Errors:	No error behaviour specified.
Security services:	Confidentiality.

D.1.2 Conformance

An implementation claiming conformance to this security exchange definition shall satisfy the following conformance requirements:

- *Statement Requirements* – An implementor shall state whether the implementation acts as an encoder, decoder, or both.
- *Static Requirements* – An implementation that acts as an encoder shall be able to generate the transformed item. An implementation that acts as a decoder shall be able to process the transformed item.
- *Dynamic Requirements* – An implementation must implement the applicable procedures described in this annex.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:29 1999

D.2 Directory SIGNED security transformation

The Directory SIGNED security transformation is functionally equivalent to the SIGNED parameterized type defined in ITU-T Rec. X.509 | ISO/IEC 9594-8. It provides for digital signature with appendix, with the transformed item including both the unprotected data to be signed and the signature appendix.

```
dirSignedTransformation  SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations dir-signed (2) }
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber-derived (2)
            distinguished-encoding (1)}
  XFORMED-DATA-TYPE SEQUENCE
  {
    toBeSigned        ABSTRACT-SYNTAX.&Type (CONSTRAINED BY {
     -- this type is constrained to being the to-be-signed
     -- type -- } ),
    algorithmId       AlgorithmIdentifier,
     -- of the algorithms used to compute the signature --
    encipheredHash    BIT STRING
  }
}
```

D.2.1 Other details

Encoding process:	The encoding process operates on a full (tag-length-value) ASN.1 DER encoding of a value of a single ASN.1 type (the "unprotected item"), and it produces a "transformed item" (a value of a SEQUENCE type as defined above). The encoded unprotected item is subjected to a function (e.g. hashing) which generates an intermediate octet string. The intermediate octet string is encoded using the ASN.1 Basic Encoding Rules, and the result is enciphered to obtain a bit string "encipheredHash". The transformed item is then constructed.
Encoding process local inputs:	Identifier of hashing and encipherment algorithm, algorithm parameters, encipherment key information.
Decoding process:	The value of the unprotected item is extracted from the transformed item and output. If the signature is to be verified, the following process is also followed. Signature verification requires the DER-encoding of the unprotected item. This can be obtained from the transformed item, but may require decoding and re-encoding with DER. The octets are subjected to a function (e.g. hashing) which generates an intermediate octet string. The encipheredHash value is deciphered and decoded using the ASN.1 Basic Encoding Rules, and the result is compared with the intermediate octet string. If they are the same, the signature is verified correctly. Otherwise an error is signalled.
Decoding process local inputs:	Identifier of hashing and encipherment algorithm, algorithm parameters, decipherment key information. Note that the algorithm and algorithm parameters can be obtained from the transformed item, but they have been stored/transferred unprotected. It is therefore recommended that these values be obtained as local inputs, possibly derived from fields conveyed within the unprotected item.
Decoding process outputs:	Recovered unprotected item, as a value of an ASN.1 type. In addition, either or both of the following outputs may optionally be produced: a) an indicator of whether or not the signature has been verified correctly; b) a copy of the transformed item or of the encipheredHash value, for storing locally for possible subsequent signature verification.
Parameters:	None.
Transformation qualifiers:	None.
Errors:	An error condition occurs if the signature verification fails.
Security Services:	Data origin authentication, data integrity, and (in certain situations) non-repudiation.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:30 1999

D.2.2 Conformance

An implementation claiming conformance to this security exchange definition shall satisfy the following conformance requirements:

- *Statement Requirements* – An implementor shall state whether the implementation acts as an encoder, decoder, or both.
- *Static Requirements* – An implementation that acts as an encoder shall be able to generate the transformed item. An implementation that acts as a decoder shall be able to process the transformed item.
- *Dynamic Requirements* – An implementation must implement the applicable procedures described in this annex.

D.3 Directory SIGNATURE security transformation

The Directory SIGNATURE security transformation is functionally equivalent to the SIGNATURE parameterized type defined in ITU-T Rec. X.509 | ISO/IEC 9594-8. It provides for digital signature with appendix, with the transformed item including the signature appendix but not the unprotected data being signed.

```
dirSignatureTransformation SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations dir-signature (3) }
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber-derived (2)
            distinguished-encoding (1)}
  XFORMED-DATA-TYPE SEQUENCE
  {
    algorithmId   AlgorithmIdentifier,
    -- of the algorithms used to compute the signature --
    encipheredHash BIT STRING
  }
}
```

D.3.1 Other details

Encoding process: The encoding process operates on a full (tag-length-value) ASN.1 DER encoding of a value of a single ASN.1 type (the "unprotected item"), and it produces a "transformed item" (a value of a SEQUENCE type as defined above). The encoded unprotected item is subjected to a function (e.g. hashing) which generates an intermediate octet string. The intermediate octet string is encoded using the ASN.1 Basic Encoding Rules, and the result is enciphered to obtain a bit string "encipheredHash". The transformed item is then constructed.

Encoding process local inputs: Identifier of hashing and encipherment algorithm, algorithm parameters, encipherment key information.

Decoding process: If the signature is to be verified, the following process is followed. Signature verification requires the DER-encoding of the unprotected item. This is obtained as a local input. The octets are subjected to a function (e.g. hashing) which generates an intermediate octet string. The encipheredHash value is deciphered and decoded using the ASN.1 Basic Encoding Rules, and the result is compared with the intermediate octet string. If they are the same, the signature is verified correctly. Otherwise, an error is signalled.

Decoding process local inputs: The unprotected item, identifier of hashing and encipherment algorithm, algorithm parameters, decipherment key information. Note that the algorithm and algorithm parameters can be obtained from the transformed item, but they have been stored/transferred unprotected. It is therefore recommended that these values be obtained as local inputs, possibly derived from fields conveyed within the unprotected item.

Decoding process outputs: Either or both of the following outputs may optionally be produced:

a) an indicator of whether or not the signature has been verified correctly;

b) a copy of the transformed item or of the encipheredHash value, for storing locally for possible subsequent signature verification.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:30 1999

Parameters:	None.
Transformation qualifiers:	None.
Errors:	An error condition occurs if the signature verification fails.
Security Services:	Data origin authentication, data integrity, and (in certain situations) non-repudiation.

D.3.2 Conformance

An implementation claiming conformance to this security exchange definition shall satisfy the following conformance requirements:

- *Statement Requirements* – An implementor shall state whether the implementation acts as an encoder, decoder, or both.
- *Static Requirements* – An implementation that acts as an encoder shall be able to generate the transformed item. An implementation that acts as a decoder shall be able to process the transformed item.
- *Dynamic Requirements* – An implementation must implement the applicable procedures described in this annex.

D.4 GULS SIGNED security transformation

The GULS SIGNED security transformation provides for digital signature or sealing with appendix, with the transformed item including both the unprotected data to be signed and the signature/seal appendix. It performs a comparable function as the Directory SIGNED security transformation, but has the following features:

- it can support any appendix-based signature or sealing technique, i.e. it is not restricted to an enciphered-hash technique like Directory SIGNED;
- it removes the restriction of using Distinguished Encoding Rules only; any single-valued encoding rules (including Canonical Encoding Rules) may be used;
- it supports protected parameters to indicate initial encoding rules, algorithm identifiers, algorithm parameters, and key information;
- it provides for identifying a digital signature algorithm and a hash function by distinct algorithm identifiers;
- it ensures that the signature is computed on the same encoding of the signed data item as is transferred, thereby averting a potential need for the decoder to decode and recode the data when verifying the signature.

Alternative protection mappings are defined in Annex E, enabling the notation PROTECTED {BaseType, signed} to map to either the Directory SIGNED or GULS SIGNED security transformation.

```
gulsSignedTransformation {KEY-INFORMATION: SupportedKIClasses}
        SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations guls-signed (4) }
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber-derived (2)
            canonical-encoding (0) }
  -- This default for initial encoding rules may be overridden
  -- using a static protected parameter (initEncRules).
  XFORMED-DATA-TYPE SEQUENCE
  {
    intermediateValue  EMBEDDED PDV (WITH COMPONENTS {
       identification (WITH COMPONENTS
          {transfer-syntax (CONSTRAINED BY {
            -- The transfer syntax to be used is that
            -- indicated by the initEncRules value within
            -- the intermediate value -- }) PRESENT}),
       data-value (WITH COMPONENTS {notation (IntermediateType
          { {SupportedKIClasses} })})
            -- The data value encoded is a value of type IntermediateType
       }),
    appendix     BIT STRING (CONSTRAINED BY {
       -- the appendix value must be generated following
       -- the procedure specified in D.4 of DIS 11586-1 -- })
  }
}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:31 1999

```
IntermediateType {KEY-INFORMATION: SupportedKIClasses }::= SEQUENCE
{
  unprotectedItem  ABSTRACT-SYNTAX.&Type
            -- this type is constrained to being
            -- the type of the unprotected item, or
            -- BIT STRING if the unprotected item is
            -- not derived from an ASN.1 abstract
            -- syntax --,
  initEncRules     OBJECT IDENTIFIER DEFAULT
           {joint-iso-itu-t asn1 (1) ber-derived (2)
           canonical-encoding (0)},
  signOrSealAlgorithm  AlgorithmIdentifier OPTIONAL,
            -- Identifies the signing or
            -- sealing algorithm, and can convey
            -- algorithm parameters --
  hashAlgorithm    AlgorithmIdentifier OPTIONAL,
            -- Identifies a hash function,
            -- for use if a hash function is required
            -- and the signOrSealAlgorithm identifier
            -- does not imply a particular hash
            -- function. Can also convey algorithm
            -- parameters.--
  keyInformation   SEQUENCE
           {
             kiClass          KEY-INFORMATION.&kiClass
                              ({SupportedKIClasses}),
             keyInfo          KEY-INFORMATION.&KiType
                              ({SupportedKIClasses}
                              {@.kiClass})
           } OPTIONAL
            -- Key information may assume various
            -- formats, governed by supported members
            -- of the KEY-INFORMATION information
            -- object class (defined at start of the
            -- definitive ASN.1 module)
}
```

D.4.1 Other details

Encoding process: The encoding process operates on a value of a single ASN.1 type (the "unprotected item"), and it produces a "transformed item" (a value of a SEQUENCE type as defined above). (If the unprotected item is not derived from an ASN.1 abstract syntax specification, it may be considered a value of an ASN.1 BIT STRING type.) First an "intermediate value", of the ASN.1 type IntermediateType is generated. This is encoded using the initial encoding rules, determined as specified in 7.1.4. The resultant octets (the full tag-length-value encoding) are subjected to a signing or sealing process, which may or may not employ a hash function. This process generates an appendix value as a bit string. The transformed item is then constructed.

NOTE – Examples of the "signing or sealing process" for different algorithms are:

a) Compute a message authentication code in accordance with ISO 8730 (this is a type of seal).

b) Concatenate an encoding of a BIT STRING containing a secret key value to the encoding of the IntermediateType, then apply a hash function to the result (this is a type of seal).

c) Apply a hash function to the encoding of IntermediateType, then sign the resultant hash value using a digital signature or public-key encryption algorithm.

Encoding process local inputs: Identifier of signing or sealing algorithm, (optional) identifier of hashing algorithm, algorithm parameters, signing/sealing key information.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:31 1999

Decoding process:	The value of the unprotected item is extracted from the transformed item and output. If the signature is to be verified, the following process is also followed. Signature verification requires an encoding of the intermediate value, using the initial encoding rules. This can be obtained from the transformed item. The signature or seal verification process is performed.
Decoding process local inputs:	Signature/seal verification key information. Identifier of signing or sealing algorithm, identifier of hashing algorithm, and/or algorithm parameters may also be needed if they are not conveyed as protected parameters.
Decoding process outputs:	Recovered unprotected item, as a value of an ASN.1 type. In addition, either or both of the following outputs may optionally be produced: a) an indicator of whether or not the signature has been verified correctly; b) a copy of the transformed item or of the appendix value, for storing locally for possible subsequent signature verification.
Parameters:	Optional static protected parameters are: initial encoding rules, identifier of signature/sealing algorithm, parameters of signature/sealing algorithm, identifier of hashing algorithm, parameters of hashing algorithm, key information.
Transformation qualifiers:	None.
Errors:	An error condition occurs if the signature/seal verification fails.
Security Services:	Data origin authentication, data integrity, and (in certain situations) non-repudiation.

D.4.2 Conformance

An implementation claiming conformance to this security exchange definition shall satisfy the following conformance requirements:

- *Statement Requirements* – An implementor shall state whether the implementation acts as an encoder, decoder, or both.
- *Static Requirements* – An implementation that acts as an encoder shall be able to generate the transformed item. An implementation that acts as a decoder shall be able to process the transformed item.
- *Dynamic Requirements* – An implementation must implement the applicable procedures described in this annex.

D.5 GULS SIGNATURE security transformation

The GULS SIGNATURE security transformation provides for digital signature or sealing with appendix, with the transformed item including the signature/seal appendix but not the unprotected data to be signed. It performs a comparable function as the Directory SIGNATURE security transformation, but has the following features:

- it can support any appendix-based signature or sealing technique, i.e. it is not restricted to an enciphered-hash technique like Directory SIGNATURE;
- it removes the restriction of using Distinguished Encoding Rules only; any single-valued encoding rules (including Canonical Encoding Rules) may be used;
- it supports protected parameters to indicate initial encoding rules, algorithm identifiers, algorithm parameters, and key information;
- it provides for identifying a digital signature algorithm and a hash function by distinct algorithm identifiers;
- the encoding and decoding processes have been simplified.

Alternative protection mappings are defined in Annex E, enabling the notation PROTECTED {BaseType, signed} to map to either the Directory SIGNATURE or GULS SIGNATURE security transformation.

```
gulsSignatureTransformation {KEY-INFORMATION: SupportedKIClasses }
        SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations guls-signature (5) }
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber-derived (2)
            canonical-encoding (0)}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:31 1999

```
-- This default for initial encoding rules may be overridden
-- using a static protected parameter (initEncRules).
XFORMED-DATA-TYPE SEQUENCE
{
initEncRules     OBJECT IDENTIFIER DEFAULT
          {joint-iso-itu-t asn1 (1) ber-derived (2)
          canonical-encoding (0)},
signOrSealAlgorithm  AlgorithmIdentifier OPTIONAL,
          -- Identifies the signing or
          -- sealing algorithm, and can convey
          -- algorithm parameters --
hashAlgorithm    AlgorithmIdentifier OPTIONAL,
          -- Identifies a hash function,
          -- for use if a hash function is required
          -- and the signOrSealAlgorithm identifier
          -- does not imply a particular hash
          -- function. Can also convey algorithm
          -- parameters.--
keyInformation   SEQUENCE
          {
            kiClass          KEY-INFORMATION.&kiClass
                             ({SupportedKIClasses}),
            keyInfo          KEY-INFORMATION.&KiType
                             ({SupportedKIClasses}
                             {@.kiClass})
          } OPTIONAL,
          -- Key information may assume various
          -- formats, governed by supported members
          -- of the KEY-INFORMATION information
          -- object class (defined at start of the
          -- definitive ASN.1 module)
appendix       BIT STRING (CONSTRAINED BY {
      -- the appendix value must be generated following
      -- the procedure specified in D.5 of DIS 11586-1 -- })
}
}
```

D.5.1 Other details

Encoding process: The encoding process operates on a value of a single ASN.1 type (the "unprotected item"), and it produces a "transformed item" (a value of a SEQUENCE type as defined above). (If the unprotected item is not derived from an ASN.1 abstract syntax specification, it may be considered a value of an ASN.1 BIT STRING type.) First an "intermediate value", of the ASN.1 type IntermediateType defined in D.4 is generated. This is encoded using the initial encoding rules, determined as specified in 7.1.4. The resultant octets (the full tag-length-value encoding) are subjected to a signing or sealing process, which may or may not employ a hash function. This process generates an appendix value as a bit string. The transformed item is then constructed.

NOTE – Examples of the "signing or sealing process" for different algorithms are:

a) Compute a message authentication code in accordance with ISO 8730 (this is a type of seal).

b) Concatenate an encoding of a BIT STRING containing a secret key value to the encoding of the IntermediateType then apply a hash function to the result (this is a type of seal).

c) Apply a hash function to the encoding of IntermediateType then sign the resultant hash value using a digital signature or public-key encryption algorithm.

Encoding process local inputs: Identifier of signing or sealing algorithm, (optional) identifier of hashing algorithm, algorithm parameters, signing/sealing key information.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:32 1999

Decoding process:	If the signature is to be verified, the following process is followed. Signature verification requires an encoding of the intermediate value, using the initial encoding rules. This requires the value of the unprotected item, which is obtained as a local input. The protected parameter values can be obtained from the transformed item, but may require decoding and re-encoding with the required encoding rules. The signature or seal verification process is performed.
Decoding process local inputs:	Unprotected item, signature/seal verification key information. Identifier of signing or sealing algorithm, identifier of hashing algorithm, and/or algorithm parameters may also be needed if they are not conveyed as protected parameters.
Decoding process outputs:	Either or both of the following outputs may optionally be produced: a) an indicator of whether or not the signature has been verified correctly; b) a copy of the transformed item or of the appendix value, for storing locally for possible subsequent signature verification.
Parameters:	Optional static protected parameters are: initial encoding rules, identifier of signature/sealing algorithm, parameters of signature/sealing algorithm, identifier of hashing algorithm, parameters of hashing algorithm, key information.
Transformation qualifiers:	None.
Errors:	An error condition occurs if the signature/seal verification fails.
Security Services:	Data origin authentication, data integrity, and (in certain situations) non-repudiation.

D.5.2 Conformance

An implementation claiming conformance to this security exchange definition shall satisfy the following conformance requirements:

- *Statement Requirements* – An implementor shall state whether the implementation acts as an encoder, decoder, or both.
- *Static Requirements* – An implementation that acts as an encoder shall be able to generate the transformed item. An implementation that acts as a decoder shall be able to process the transformed item.
- *Dynamic Requirements* – An implementation must implement the applicable procedures described in this annex.

D.6 Definitive ASN.1 specification

```
GulsSecurityTransformations {joint-iso-itu-t genericULS (20)
      modules (1) gulsSecurityTransformations (3) }
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

-- EXPORTS All --

IMPORTS
  securityTransformations, notation
    FROM ObjectIdentifiers {joint-iso-itu-t genericULS (20)
    modules (1) objectIdentifiers (0) }
  SECURITY-TRANSFORMATION, SecurityIdentity
    FROM Notation notation
  AlgorithmIdentifier
    FROM AuthenticationFramework {joint-iso-itu-t ds (5)
    module (1) authenticationFramework(7) 2 };

-- ************************************** --
-- Notation for specifying key information --
-- ************************************** --

KEY-INFORMATION ::= CLASS
-- This information object class definition is for use when
-- specifying key information relating to particular classes
-- of protection mechanisms (e.g. symmetric, asymmetric).
-- It may be useful in defining various security transformations.
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:32 1999

```
{
  &kiClass
    CHOICE
    {  local       INTEGER,
         -- local objects can only be defined within this
         -- ASN.1 module.
       global   OBJECT IDENTIFIER
         -- global objects are defined elsewhere
    }UNIQUE,
  &KiType
}
WITH SYNTAX
{
  KEY-INFO-CLASS     &kiClass
  KEY-INFO-TYPE      &KiType
}

symmetricKeyInformation KEY-INFORMATION ::= {
  KEY-INFO-CLASS       local: 0
  KEY-INFO-TYPE SEQUENCE
  {
    entityId             SecurityIdentity,
    keyIdentifier        INTEGER
  }
}

asymmetricKeyInformation KEY-INFORMATION ::= {
  KEY-INFO-CLASS       local: 1
  KEY-INFO-TYPE SEQUENCE
  {
    issuerCAName       SecurityIdentity OPTIONAL,
    certSerialNumber     INTEGER OPTIONAL,
    signerName         SecurityIdentity OPTIONAL,
    keyIdentifier      BIT STRING OPTIONAL
  }
}

-- ******************************************** --
-- Directory ENCRYPTED Security Transformation --
-- ******************************************** --

dirEncryptedTransformation SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations dir-encrypted (1) }
   -- This transformation transforms a string of octets to a
   -- new bit string using an encipherment process.
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber (1) }
  XFORMED-DATA-TYPE BIT STRING
}

-- ************************************** --
-- Directory SIGNED Security Transformation --
-- ************************************** --

dirSignedTransformation   SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations dir-signed (2) }
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber-derived (2)
            distinguished-encoding (1)}
  XFORMED-DATA-TYPE SEQUENCE
  {
    toBeSigned           ABSTRACT-SYNTAX.&Type (CONSTRAINED BY {
     -- this type is constrained to being the to-be-signed type -- } ),
    algorithmId          AlgorithmIdentifier,
     -- of the algorithms used to compute the signature --
    encipheredHash BIT STRING
  }
}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:32 1999

```
-- ****************************************** --
-- Directory SIGNATURE Security Transformation --
-- ****************************************** --

dirSignatureTransformation SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations dir-signature (3) }
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber-derived (2)
            distinguished-encoding (1)}
  XFORMED-DATA-TYPE SEQUENCE
  {
    algorithmId      AlgorithmIdentifier,
    -- of the algorithms used to compute the signature --
    encipheredHash BIT STRING
  }
}

-- ********************************** --
-- GULS SIGNED Security Transformation --
-- ********************************** --

gulsSignedTransformation {KEY-INFORMATION: SupportedKIClasses }
        SECURITY-TRANSFORMATION ::=
{
  IDENTIFIER {securityTransformations guls-signed (4) }
  INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber-derived (2)
            canonical-encoding (0)}
  -- This default for initial encoding rules may be overridden
  -- using a static protected parameter (initEncRules).
  XFORMED-DATA-TYPE SEQUENCE
  {
    intermediateValue  EMBEDDED PDV (WITH COMPONENTS {
      identification (WITH COMPONENTS
        {transfer-syntax (CONSTRAINED BY {
          -- The transfer syntax to be used is that
          -- indicated by the initEncRules value within
          -- the intermediate value -- }) PRESENT}),
      data-value (WITH COMPONENTS {notation (IntermediateType
        { {SupportedKIClasses} })})
          -- The data value encoded is a value of type
          -- IntermediateType
      }),
    appendix     BIT STRING (CONSTRAINED BY {
        -- the appendix value must be generated following
        -- the procedure specified in D.4 of DIS 11586-1 -- })
  }
}
IntermediateType {KEY-INFORMATION: SupportedKIClasses }::= SEQUENCE
{
  unprotectedItem  ABSTRACT-SYNTAX.&Type
            -- this type is constrained to being
            -- the type of the unprotected item, or
            -- BIT STRING if the unprotected item is
            -- not derived from an ASN.1 abstract
            -- syntax --,
  initEncRules     OBJECT IDENTIFIER DEFAULT
            {joint-iso-itu-t asn1 (1) ber-derived (2)
            canonical-encoding (0)},
  signOrSealAlgorithm  AlgorithmIdentifier OPTIONAL,
            -- Identifies the signing or
            -- sealing algorithm, and can convey
            -- algorithm parameters --
  hashAlgorithm    AlgorithmIdentifier OPTIONAL,
            -- Identifies a hash function,
            -- for use if a hash function is required
            -- and the signOrSealAlgorithm identifier
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:33 1999

```
            -- does not imply a particular hash
            -- function. Can also convey algorithm
            -- parameters.--
 keyInformation   SEQUENCE
          {
            kiClass   KEY-INFORMATION.&kiClass
                      ({SupportedKIClasses}),
            keyInfo   KEY-INFORMATION.&KiType
                      ({SupportedKIClasses}
                      {@.kiClass})
          } OPTIONAL
            -- Key information may assume various
            -- formats, governed by supported members
            -- of the KEY-INFORMATION information
            -- object class (defined at start of the
            -- definitive ASN.1 module)
}


-- ************************************** --
-- GULS SIGNATURE Security Transformation --
-- ************************************** --

gulsSignatureTransformation {KEY-INFORMATION: SupportedKIClasses }
        SECURITY-TRANSFORMATION ::=
{
 IDENTIFIER {securityTransformations guls-signature (5) }
 INITIAL-ENCODING-RULES {joint-iso-itu-t asn1 (1) ber-derived (2)
          canonical-encoding (0)}
 -- This default for initial encoding rules may be overridden
 -- using a static protected parameter (initEncRules).
 XFORMED-DATA-TYPE SEQUENCE
 {
 initEncRules     OBJECT IDENTIFIER DEFAULT
          {joint-iso-itu-t asn1 (1) ber-derived (2)
           canonical-encoding (0)},
 signOrSealAlgorithm  AlgorithmIdentifier OPTIONAL,
            -- Identifies the signing or
            -- sealing algorithm, and can convey
            -- algorithm parameters --
 hashAlgorithm    AlgorithmIdentifier OPTIONAL,
            -- Identifies a hash function,
            -- for use if a hash function is required
            -- and the signOrSealAlgorithm identifier
            -- does not imply a particular hash
            -- function. Can also convey algorithm parameters.--
 keyInformation   SEQUENCE
          {
            kiClass   KEY-INFORMATION.&kiClass
                      ({SupportedKIClasses}),
            keyInfo   KEY-INFORMATION.&KiType
                      ({SupportedKIClasses}
                      {@.kiClass})
          } OPTIONAL,
            -- Key information may assume various
            -- formats, governed by supported members
            -- of the KEY-INFORMATION information
            -- object class (defined at start of the
            -- definitive ASN.1 module)
 appendix       BIT STRING (CONSTRAINED BY {
        -- the appendix value must be generated following
        -- the procedure specified in D.5 of DIS 11586-1 -- })
 }
}

END
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:33 1999

Annex E

Protection mapping specifications

(This annex forms an integral part of this Recommendation | International Standard)

Protection mappings are defined in ASN.1 modules. Any such module may be defined in ITU-T Recommendations | International Standards or may be defined outside Recommendations | International Standards and registered by any organization able to assign object identifiers. Protection mapping definitions should be made as broadly applicable as possible so they can be reused in multiple applications. This annex defines some protection mappings which are considered to be generally useful. There is no implied requirement for applications or implementations thereof to employ the specific protection mappings defined here, in preference to other protection mappings.

```
DirectoryProtectionMappings {joint-iso-itu-t genericULS (20)
      modules (1) dirProtectionMappings (4) }
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
-- These protection mappings generate bit-compatible encodings
-- to the parameterized types in the Directory Authentication
-- Framework

-- EXPORTS All --

IMPORTS
  notation, gulsSecurityTransformations
    FROM ObjectIdentifiers {joint-iso-itu-t genericULS (20)
    modules (1) objectIdentifiers (0) }
  PROTECTION-MAPPING
    FROM Notation notation
  dirEncryptedTransformation, dirSignedTransformation,
  dirSignatureTransformation
    FROM GulsSecurityTransformations
          gulsSecurityTransformations;

-- ************************************** --
-- Directory encrypted Protection Mapping --
-- ************************************** --

-- This protection mapping enables the notation
-- PROTECTED {BaseType, encrypted}
-- to replace the notation
-- ENCRYPTED {BaseType}
-- as provided by ITU-T Rec. X.509 | ISO/IEC 9594-8:1994, and to
-- generate an identical bit-encoding.
-- Security Service:  confidentiality

encrypted PROTECTION-MAPPING ::=
{
  SECURITY-TRANSFORMATION {dirEncryptedTransformation }
}


-- ********************************** --
-- Directory signed Protection Mapping --
-- ********************************** --

-- This protection mapping enables the notation
-- PROTECTED {BaseType, signed}
-- to replace the notation
-- SIGNED {BaseType}
-- as provided by ITU-T Rec. X.509 | ISO/IEC 9594-8:1994, and to
-- generate an identical bit-encoding.
-- Security Service:  data origin authentication, data integrity and
-- (in certain situations) non-repudiation.
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:33 1999

```
signed PROTECTION-MAPPING ::=
{
  SECURITY-TRANSFORMATION {dirSignedTransformation }
}

-- ************************************** --
-- Directory signature Protection Mapping --
-- ************************************** --

-- This protection mapping enables the notation
-- PROTECTED {BaseType, signature}
-- to provide a functionally-equivalent replacement of the notation
-- SIGNATURE BaseType
-- as provided by ITU-T Rec. X.509 | ISO/IEC 9594-8.
-- Security Service:  data origin authentication, data integrity and
-- (in certain situations) non-repudiation.

signature PROTECTION-MAPPING ::=
{
  SECURITY-TRANSFORMATION  {dirSignatureTransformation }
}
END


GULSProtectionMappings {joint-iso-itu-t genericULS (20)
        modules (1) gulsProtectionMappings (5) }
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
-- These protection mappings are more versatile that the
-- preceding protection mappings which were specifically designed
-- to generate identical bit-encodings as the Directory
-- Authentication Framework parameterized types.

-- EXPORTS All --

IMPORTS
  notation, gulsSecurityTransformations
    FROM ObjectIdentifiers {joint-iso-itu-t genericULS (20)
    modules (1) objectIdentifiers (0) }
  PROTECTION-MAPPING
    FROM Notation notation
  dirEncryptedTransformation, gulsSignedTransformation,
  gulsSignatureTransformation, symmetricKeyInformation,
  asymmetricKeyInformation
    FROM GulsSecurityTransformations
          gulsSecurityTransformations;


-- ********************************** --
-- confidentiality Protection Mapping --
-- ********************************** --

-- This protection mapping enables the notation
-- PROTECTED {BaseType, confidentiality}
-- to map to either dirEncryptedTransformation or to no transformation
-- at the choice of the encoding system, dependent upon local security
-- policy and other local environment considerations.
-- Security Service:  confidentiality

confidentiality PROTECTION-MAPPING ::=
{
  SECURITY-TRANSFORMATION {dirEncryptedTransformation }
  BYPASS-PERMITTED TRUE
}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:34 1999

```
-- ****************************** --
-- GULS signed Protection Mapping --
-- ****************************** --

-- This protection mapping causes the notation
-- PROTECTED {BaseType, signed}
-- to map to the gulsSignedTransformation.
-- Security Service:  data origin authentication, data integrity and
-- (in certain situations) non-repudiation.

signed PROTECTION-MAPPING ::=
{
  SECURITY-TRANSFORMATION {gulsSignedTransformation
    {{symmetricKeyInformation | asymmetricKeyInformation }}}
}

-- ********************************** --
-- GULS signature Protection Mapping --
-- ********************************** --

-- This protection mapping causes the notation
-- PROTECTED {BaseType, signature}
-- to map to the gulsSignatureTransformation.
-- Security Service:  data origin authentication, data integrity and
-- (in certain situations) non-repudiation.

signature PROTECTION-MAPPING ::=
{
  SECURITY-TRANSFORMATION {gulsSignatureTransformation
    {{symmetricKeyInformation | asymmetricKeyInformation }}}
}

END
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:34 1999

Annex F

Object identifier usage

(This annex forms an integral part of this Recommendation | International Standard)

This annex documents the upper reaches of the object identifier subtree in which all of the object identifiers assigned in this series of Specifications reside. It does so by providing an ASN.1 module called ObjectIdentifiers in which all non-leaf nodes in the subtree are assigned names. The full set of ASN.1 modules defined in this Series of Standards is also identified.

```
ObjectIdentifiers {joint-iso-itu-t genericULS (20)
      modules (1) objectIdentifiers (0) }
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

-- EXPORTS All --

genericULS              OBJECT IDENTIFIER ::=
                        {joint-iso-itu-t genericULS (20) }

-- Categories of information object --

modules                 OBJECT IDENTIFIER ::= {genericULS 1}
generalTransferSyntax   OBJECT IDENTIFIER ::= {genericULS 2}
specificTransferSyntax  OBJECT IDENTIFIER ::= {genericULS 3}
securityExchanges       OBJECT IDENTIFIER ::= {genericULS 4}
securityTransformations OBJECT IDENTIFIER ::= {genericULS 5}

-- ASN.1 modules --

objectIdentifiers       OBJECT IDENTIFIER ::= {modules 0}
notation                OBJECT IDENTIFIER ::= {modules 1}
gulsSecurityExchanges   OBJECT IDENTIFIER ::= {modules 2}
gulsSecurityTransformations
                        OBJECT IDENTIFIER ::= {modules 3}
dirProtectionMappings
                        OBJECT IDENTIFIER ::= {modules 4}
gulsProtectionMappings
                        OBJECT IDENTIFIER ::= {modules 5}
seseAPDUs               OBJECT IDENTIFIER ::= {modules 6}
genericProtectingTransferSyntax
                        OBJECT IDENTIFIER ::= {modules 7}
END
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:34 1999

Annex G

Guidelines for the use of generic upper layers security facilities

(This annex does not form an integral part of this Recommendation | International Standard)

G.1 Introduction

This annex explains how the GULS Standards can be used to provide security for a particular application, assuming that GULS tools are suitable to provide security for that application.

It is desirable that the designer of any OSI application protocol employ the same security solutions as are employed in other OSI application protocols. This cannot, in general, be fully achieved, because different applications have different security requirements, and security solutions will require some tailoring to meet the needs of different applications. However, it is usually possible for different applications to adopt common solutions based on the identification of common security requirements.

The purpose of this series of Recommendations | International Standards is to provide a collection of security protocol facilities which can contribute to the incorporation of security solutions into any application protocol, and which encourage the adoption of common security solutions in different applications. However, these specifications do not themselves provide all the specification for common security solutions.

G.2 Generic facilities provided

The facilities provided in the GULS Standards include:

- a general means of building Application Layer protocol components to support the exchange of security-related information between a pair of communicating application-entity-invocations (the *security exchange concept*, which is supported by the *SESE*);
- a general approach to using Presentation Layer facilities to perform security-related transformations on information items in order to protect these items (the *generic protecting transfer syntax*);
- abstract syntax notation tools to assist an application protocol designer in specifying that security protection is to apply to selected fields of his protocol (a *PROTECTED* parameterized type, and the *PROTECTED-Q* variant of this type).

Another generic security facility of this nature is the authentication functional unit of ACSE. While that facility is defined in ITU-T Rec. X.217 | ISO/IEC 8649 and CCITT Rec. X.227 | ISO/IEC 8650-1, rather than in this Recommendation | International Standard, this annex will address use of that facility in developing security solutions for applications.

It is intended that the facilities described be used by the designers of new applications, when addressing their security needs. However, these facilities may also be used in adding security features to existing OSI application protocols. To some extent, this may be achieved by building a new application context which incorporates these facilities, without necessarily having to modify existing ASE specifications. However, to provide some security services (e.g. selective field confidentiality or integrity), changes to other ASE specifications will be necessary.

G.3 Aspects of security solutions not provided in this Recommendation | International Standard

The scope of this Recommendation | International Standard is limited to the communication of information associated with the provision of security services, i.e. it does not extend to the full details of providing any security service or of implementing any security mechanism. General aspects of security mechanisms are described in the security frameworks Recommendations | International Standards (ITU-T Recs. X.811, X.812, X.813. X.814, X.815 | ISO/IEC 10181). Recommendations | International Standards for certain specific security mechanisms and supporting security techniques are developed by ISO/IEC JTC1/SC27.

In particular, implementation of the generic facilities described in this Recommendation | International Standard depends upon one or both of:

- specifications of particular *security exchanges*, designed to support specific security mechanisms (e.g. a specific authentication exchange);
- specifications of particular *security transformations*, which transform user data for protection purposes in some particular way (e.g. an encipherment process).

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:35 1999

Such specifications are not provided in this Recommendation I International Standard (except for some generally-useful instances defined in Annexes C and D). However, this Recommendation I International Standard does include tools and guidelines to assist in the production of such specifications. It should be noted that when such specifications are produced, they should be able to be used to support many different applications, when used in conjunction with the facilities described in this Recommendation I International Standard.

In addition, this Recommendation I International Standard does not specify procedures for the establishment of externally established security associations.

This Recommendation I International Standard does not include the definition of a service interface to the security exchanges which is independent of the mechanisms used.

G.4 Use of the GULS facilities in providing security services

Following is an indication of how the generic facilities described in these Recommendations I International Standards may be used in supporting the provision of the security services identified in CCITT Rec. X.800 I ISO 7498-2 for the Application Layer. These services will counter vulnerabilities identified by specific application groups.

Details of the provision of the security services given below using the GULS tools may be defined separately in other ITU-T Recommendations I International Standards.

G.4.1 Entity authentication

Entity authentication (as described in ITU-T Rec. X.811 I ISO/IEC 10181-2) generally involves an *authentication exchange*, which is an n-way exchange of authentication information between two parties (n is typically 1, 2, or 3, but may be larger). Hence, an authentication exchange can be considered to be a special case of a security exchange.

There are two potential ways of supporting an authentication exchange using the generic upper layers security facilities:

- in the particular case where the authentication exchange is restricted to being one-way or two-way, and where it is restricted to occurring only in conjunction with OSI application-association establishment, then the authentication exchange can be conveyed using the ACSE authentication functional unit;
- in all cases (the above restrictions do not apply), the authentication exchange can be conveyed using the SESE.

Note that the type of identity being authenticated is immaterial, and is not restricted to being that of any OSI entity. Also, entity authentication is not restricted to occurring at the start of an association (e.g. it might occur at the start of a TP dialogue, or at any time within an association).

Entity authentication may also involve communication with a third party. The protocol for this purpose could be an application protocol, in which case security exchanges may also be employed in this protocol.

G.4.2 Data origin authentication

A common method of data origin authentication is to attach a signature or seal to the item whose source is being authenticated. This can be achieved by conveying the item in a security association which employs a signing or sealing type of security transformation.

To protect a category of complete PDUs in this way, the application context specification would contain rules indicating that such PDUs need to conveyed in a protecting presentation context. To protect an individual information item within an abstract syntax, the PROTECTED parameterized type could be used.

G.4.3 Access control

Many aspects of access control are application-specific and cannot be dealt with in a generic way. However, the communication of access control information (relating to the granting, enforcing and revoking of access control rights) may be achieved using a security exchange. For example, the transfer of an access control certificate may be viewed as a simple (one-way) security exchange. Such a certificate can then be attached to any other PDU, by conveying it using the security exchange services of the SESE. An example of using the SESE for this purpose is given in I.4.

Integrity and/or data origin authentication of exchanged access control information are usually also very important. Provision of the necessary protection may be achieved by conveying the access control information in a security association which employs a signing or sealing type of security transformation.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:35 1999

G.4.4 Connection and connectionless confidentiality

Confidentiality of a complete PDU can be achieved by conveying it in a security association which employs an encipherment type of security transformation. To protect a category of complete PDUs in this way, the application context specification would contain rules indicating that such PDUs need to conveyed in a protecting presentation context.

G.4.5 Selective field confidentiality

Confidentiality of any protocol field can be achieved by conveying it in a security association which employs an encipherment type of security transformation. To identify which individual information items within an abstract syntax require such protection, the PROTECTED parameterized type can be used.

G.4.6 Traffic flow confidentiality

A transformation encoding process might provide for the attachment of padding data to a protected item, but not for the generation of PDUs containing only padding data.

G.4.7 Connection and connectionless integrity

Integrity of a complete PDU can be achieved by conveying it in a security association which employs a signing or sealing type of security transformation. To protect a category of complete PDUs in this way, the application context specification would contain rules indicating that such PDUs need to conveyed in a protecting presentation context.

G.4.8 Selective field integrity

Integrity of any protocol field can be achieved by conveying it in a security association which employs a signing or sealing type of security transformation. To identify which individual information items within an abstract syntax require such protection, the PROTECTED parameterized type could be used.

G.4.9 Non-repudiation

The provision of a non-repudiation service (wih proof of origin or proof of delivery) typically requires integrity and/or data origin authentication to be applied to communicated data. Provision of the necessary protection may be achieved by conveying the data in a security association which employs a signing or sealing type of security transformation.

Some non-repudiation mechanisms are based on the use of non-repudiable signatures applied to communicated data. This can be achieved by conveying the data in a security association which employs a signing type of security transformation, using an asymmetric encipherment technique.

G.4.10 Audit

The provision of a security audit service typically requires other security services beyond those described in G.4.1 through G.4.9. The SESE can be used to exchange information such as security alarm and audit messages between entities. (However, there also exist other Recommendations | International Standards addressing such information exchange, e.g. CCITT Rec. X.736 | ISO/IEC 10164-7 and CCITT Rec. X.740 | ISO/IEC 10164-8.)

G.5 Key management

Key management is a complex area, many aspects of which are outside the scope of OSI. However, use of many types of protecting transformation functions in the Presentation Layer will depend upon keys having been established.

There are various ways in which keys may be established, such as:

a) manual distribution, or other means entirely outside the scope of OSI;

b) establishment of keys in a separate (earlier or overlapping) association, e.g. using OSI Systems Management services;

c) establishment of keys within the same association, but before the key is required by the transformation. This might involve, for example, a Diffie-Hellman key derivation exchange or the sending of a key protected for confidentiality purposes under some other transformation and/or some other key.

In case c), the key derivation or distribution exchange might be realized as a security exchange, and might use the security exchange services of the SESE. This might be done as part of the protocol supporting establishment of an externally established security association.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:36 1999

A key derivation or distribution may be provided as an integral part of a security exchange supporting another service, e.g. entity authentication.

Security transformation dynamic parameters conveyed in the security transfer syntax may also pertain to key management, e.g. by indicating that a particular key is to be used from a given point onwards.

G.6 Guidelines for specifying application-contexts

In general, use of the SESE will require special rules, which are not part of an ASE specification, to be written into an application-context specification. Such rules need to specify:

a) *ASEs* – Inclusion of the SESE as one of the ASEs in the application-context;

b) *Security exchanges* – The particular set of security exchanges to be supported, which implies a specific SESE abstract syntax;

c) *SESE PDU mappings* – Mappings of SESE PDUs to other services, i.e. the P-DATA service, or as an embedded presentation data value in a PDU of another ASE;

d) *PDV concatenation constraints* – Requirements for concatentation of particular SESE PDUs with presentation data values of other ASEs;

e) *PDV embedding constraints* – Requirements for embedding other presentation data values in SESE PDUs;

f) *Procedural constraints* – Rules regarding interactions of the SESE state machine with the state machines of other ASEs, e.g. to ensure that the state of other ASE protocol machines, at the successful termination or aborting of each security exchange, is well-defined and not deadlocked;

g) *Presentation context constraints* – Requirements for establishing particular transfer syntaxes for particular abstract syntaxes.

G.7 Example

Suppose it is desired to build a new application context for OSI File Transfer, Access and Management (FTAM), defined in ISO 8571, which adds three security features to the basic FTAM protocol:

a) a strong mutual authentication exchange to be employed in conjunction with association establishment;

b) an access control certificate, the format of which is defined in some other standard, to be bound to every F-SELECT or F-CREATE request;

c) confidentiality and integrity protection are to be applied to all file contents data transmitted.

It is desired to achieve this without modifying the FTAM ASE abstract syntax.

The first step is to identify, or specify if necessary, the required security exchanges. A two-way security exchange is required for feature a), and a one-way security exchange is required for feature b). If only authentication is required, the security exchange dirAuthenticationTwoWay defined in Annex C could be used for a). Alternatively, a security exchange which combines authentication and key establishment might be used, in which case the key(s) derived from the exchange could be used in providing feature c). For the purposes of this example, the dirAuthenticationTwoWay security exchange will be assumed. The security exchange for feature b) could be the boundAccessControlCert security exchange defined in example I.4.

The next step is to specify a SESE abstract syntax to support these security exchanges. A later example in Part 3 of these specifications will show how this is done.

The final step is to specify the required application-context. Following the guidelines in G.6, this specification will include the following rules:

a) *ASEs* – The set of ASEs includes the SESE, as well as the (unmodified) FTAM and ACSE ASEs;

b) *Security exchanges* – The dirAuthenticationTwoWay and boundAccessControlCert security exchanges are supported;

c) *SESE PDU mappings* – The SE-TRANSFER PDUs conveying the dirAuthenticationTwoWay security exchange maps to A-ASSOCIATE request and response PDUs respectively (e.g. as additional components to the user information field); the SE-TRANSFER PDU conveying the boundAccessControlCert security exchange maps to P-DATA;

d) *PDV concatenation constraints* – None;

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:36 1999

e) *PDV embedding constraints* – Each FTAM PDU (or PDU group) containing F-SELECT request or F-CREATE request is embedded in an SE-TRANSFER PDU conveying a boundAccessControlCert security exchange;

f) *Procedural constraints* – Any error condition encountered in the dirAuthenticationTwoWay security exchange results in aborting of the application-association;

g) *Presentation context constraints* – The presentation context used for transferring file contents data must employ a protecting transfer syntax with a protection mapping implying confidentiality and integrity protection.

A new object identifier is assigned to this application context.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:36 1999

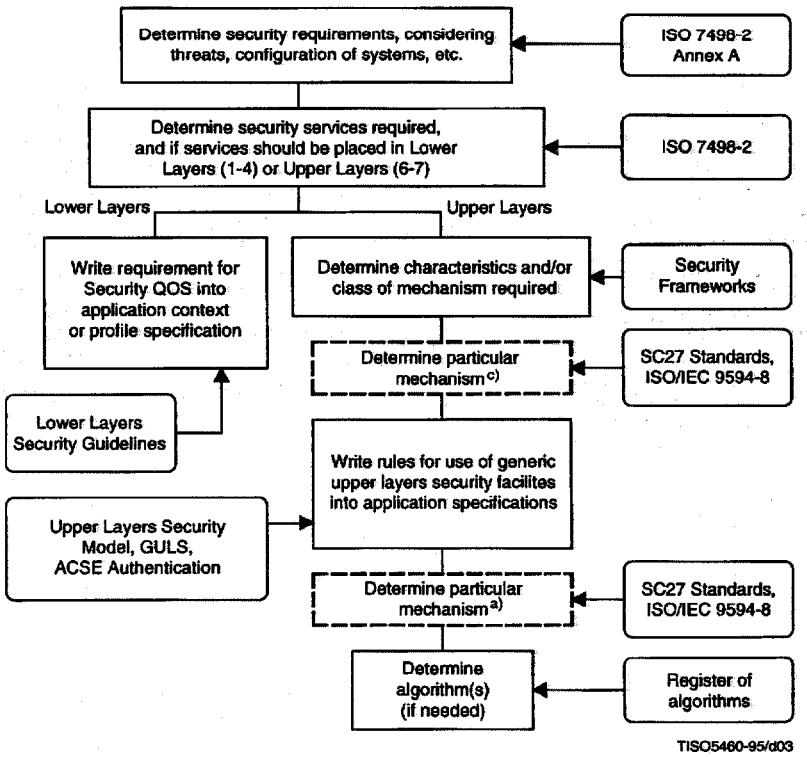
Annex H

Relationship to other standards

(This annex does not form an integral part of this Recommendation | International Standard)

This annex explains the relationship between the GULS Specifications and other standards, assuming that GULS tools are suitable to provide security for a particular application.

Figure H.1 illustrates the overall process of incorporating security into an application protocol standard.



a) Some aspects of mechanism determination can be deferred to a profiling stage, after protocol construction.

Figure H.1 – Guidelines for incorporating security into an application layer protocol

Figure H.2 illustrates where the GULS facilities fit into this overall process. The following comments refer to specific boxes in Figure H.2.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:37 1999

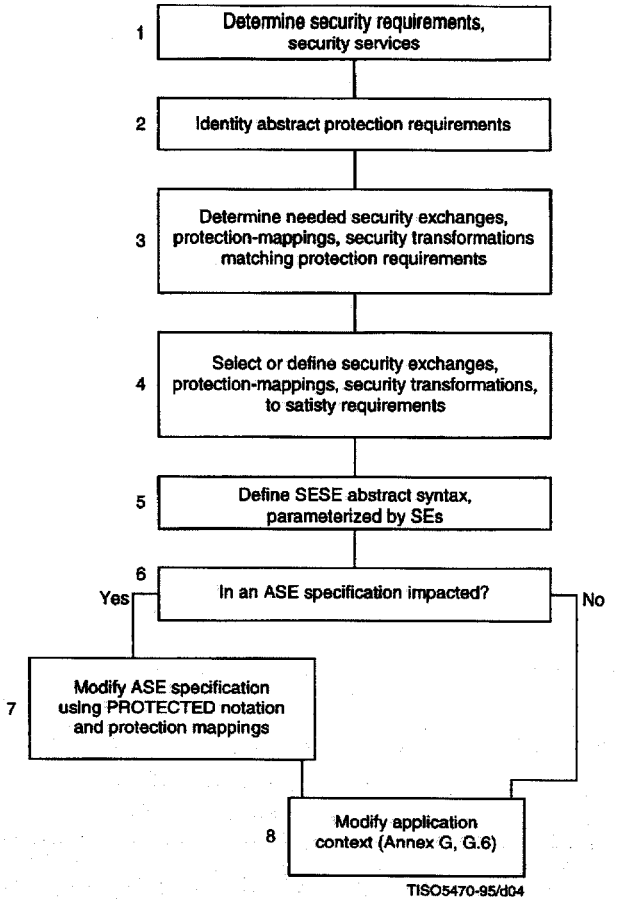


Figure H.2 – Incorporating GULS into an OSI Application Protocol

Box 4:

Security exchanges, security transformations and protection mappings may be specified by many different types of organization. The general intention is that such specifications should be reused in different applications in preference to producing new specifications which perform the same basic function. A developer of an application protocol should seek existing specifications from the following sources:

- annexes to this part of the Generic Upper Layers Security specification;
- specifications in other ITU-T Recommendations or International Standards, either as a specification usable by several applications or a particular OSI application;
- existing registered specifications, e.g. specifications developed and registered by profiling forums.

If no suitable specification can be identified, a specification should be developed by the organization requiring it, and standardized or registered with a view to future use in other applications.

Box 6:

In general, an ASE specification will only require modification if a change to an abstract syntax specification is necessary. This is only required if selective field security functions (confidentiality, integrity or data origin authentication) at a granularity smaller than that of a presentation data value are introduced. In other cases, new security services can be accommodated by changes in application-context specifications, without impacting ASE specifications.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:37 1999

To specify support for the security of an OSI application it will be necessary to produce:

a) Specifications for protocols to support security through use of particular classes of mechanism. These can include:

- security exchanges which can be specified using the SECURITY-EXCHANGE notation defined in 6.2;
- security transformations which can be specified using the SECURITY-TRANSFORMATION notation defined in 7.2.

Additional specifications may also need to be produced to define:

- use of services provided by other OSI applications processes, e.g. directory process, key management process;
- interactions and interdependencies between security transformations, security exchanges and use of other OSI application processes.

As far as possible these specifications should be applicable to a range of OSI applications.

b) Specifications incorporated with OSI application protocol specifications, to relate security provisions to application protocol specific objects. These can include:

- *Selective field protections required on application data objects* – This can be specified using the PROTECTED or PROTECTED-Q notation defined in 8.1 and 8.2.
- *Requirements for the establishment of Security Associations* – Tools for the specification of such requirements may be a subject of separate standardization.

As far as possible this should be in terms independent of particular classes of mechanism.

c) Specifications for the application of specific classes of mechanism to secure specific OSI applications. These can include:

- ASO contexts specifying use of ASEs / ASOs (e.g. SESE) for security along with other ASEs / ASOs.
- Mappings from the required protection types to security transformations which can be specified using the PROTECTION-MAPPING notation defined in 8.4.
- Requirements to apply specific security transformations to all PDVs of particular abstract syntaxes.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:37 1999

Annex I

Examples of use of the generic upper layers security facilities

(This annex does not form an integral part of this Recommendation | International Standard)

I.1 Example of use of nested PROTECTED notation

As an illustrative example of the use of the PROTECTED parameterized type, suppose an application protocol designer needs to specify a PDU with the following characteristics:

a) the entire PDU is to be sealed under an integrity mechanism;

b) the PDU contains separate fields with the following characteristics:

1) a field (of type TypeOne) requires no further security protection;

2) another field (of type TypeTwo) is to be confidentiality-protected using a symmetric algorithm; the encipherment key is also to be conveyed in the PDU, enciphered using an asymmetric algorithm under the public key of the recipient;

3) another field (of type TypeThree) is to be signed using the private key of the sender.

This PDU can be specified as an ASN.1 type as follows:

```
SecurePDU ::= PROTECTED
{
    SEQUENCE
    {
        encipheredConfKey      EncipheredConfKey,
        confidentialInfo       ConfidentialInfo,
        signedInfo             SignedInfo,
        clearInfo              TypeOne
    },
    sealed
}
EncipheredConfKey ::=   PROTECTED { ConfKey, encipheredKey}
ConfidentialInfo ::=    PROTECTED { TypeTwo, enciphered }
SignedInfo ::=          PROTECTED { TypeThree, signed }

ConfKey ::= BIT STRING
    -- Value sent is the randomly-generated value supplied
    -- and used by the security transformation used for
    -- the sym-enciphered protection mapping.
```

This ASN.1 will generate, for each instance of the PROTECTED type, an encoding as specified in clause 8. The entire PDU is one such encoding. The other three such encoding are nested within the first one. Each encoding uses a different type of transformation. The protection provided by the outer encoding applies to the entire inner contents.

This specification depends upon protection mappings "encipheredKey", "enciphered", "signed", and "sealed", which map them to transformations. The latter definitions could be in the same ASN.1 module as SecurePDU or could be parameters of that module, provided in a later stage of developing the full application-context.

An example of a set of PROTECTION-MAPPING definitions is as follows:

```
encipheredKey PROTECTION-MAPPING ::=
{   -- enciphered using an asymmetric algorithm using the public key of the recipient
    SECURITY-TRANSFORMATION { dirEncryptedTransformation }
}
enciphered PROTECTION-MAPPING ::=
{   -- enciphered using a symmetric algorithm; the key used is
    -- the last value delivered under the protection-mapping
    -- "pk-enciphered"
    SECURITY-TRANSFORMATION { dirEncryptedTransformation }
}
signed PROTECTION-MAPPING ::=
{   -- signed using the private key of the sender
    SECURITY-TRANSFORMATION { dirSignedTransformation }
}
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:38 1999

```
sealed PROTECTION-MAPPING ::=
{   -- sealed under an integrity mechanism
    SECURITY-TRANSFORMATION { sealedTransformation }
    -- sealedTransformation is not currectly defined in this
    -- Specification.
}
```

I.2 Use of PROTECTED notation with transformation qualifier – Example 1

Following is an illustration of the use of the PROTECTED-Q parameterized type, based on the example in I.1, but with qualifiers specified for use by the security transformations. The qualifiers indicate either a particular algorithm or an algorithm source for each security transformation, and the type of key to be used for each security transformation.

The PDU can be specified as an ASN.1 type as follows:

```
SecurePDU ::= PROTECTED-Q
{
    SEQUENCE
    {
        encipheredConfKey       EncipheredConfKey,
        confidentialInfo        ConfidentialInfo,
        signedInfo              SignedInfo,
        clearInfo               TypeOne
    },
    sealed, { sealAlgorithm, preEstablishedKey }
}
EncipheredConfKey ::=   PROTECTED-Q { ConfKey, encipheredKey,
                        { rsaAlgorithm, receiverAsymKeyPair }}
ConfidentialInfo ::=    PROTECTED-Q { TypeTwo, enciphered,
                        { deaAlgorithm, accompanyingEncipheredKey }}
SignedInfo ::=          PROTECTED-Q { TypeThree, signed,
                        { signAlgorithm, senderAsymKeyPair }}

ConfKey ::= BIT STRING

rsaAlgorithm    AlgorithmSelector       ::= specificAlgorithm: { iso ... }
deaAlgorithm    AlgorithmSelector       ::= specificAlgorithm: { iso ... }
signAlgorithm   AlgorithmSelector       ::= algorithmSource: userDependent
sealAlgorithm   AlgorithmSelector       ::= algorithmSource: systemDefault
```

In this example, the ASN.1 type for all qualifiers is:

```
QualifierType ::= SEQUENCE
{
    algorithmSelector   AlgorithmSelector,
    keySelector         KeySelector
}
AlgorithmSelector ::= CHOICE
{
    specificAlgorithm   OBJECT IDENTIFIER,
    algorithmSource     BIT STRING
    {
        systemDefault (0),
        -- Standard system default algorithm to be used.
        userDependent (1)
        -- Algorithm selection based on local user information.
    }
}
KeySelector ::= BIT STRING
{
    preEstablishedKey               (0),
    -- Key has been previously established between the parties.
    userSuppliedKey                 (1),
    -- Key is supplied by the sending user.
    accompanyingEncipheredKey       (2),
    -- Key accompanies the protected field, conveyed in another
    -- PROTECTED field using the encipheredKey protection mapping, as
    -- another component of the same enclosing ASN.1 construct.
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:38 1999

```
        senderAsymKeyPair                (3),
        -- Encoding key is the private key of the sender; decoding key is
        -- corresponding public key
        receiverAsymKeyPair              (4)
        -- Encoding key is the public key of the receiver; decoding key is
        -- corresponding private key
    }
```

The protection mapping definitions need to reflect the possible use of the transformation qualifiers, for example:

```
encipheredKey PROTECTION-MAPPING ::=
{   -- enciphers a key for use in protecting another field
    SECURITY-TRANSFORMATION { qualEncryptedTransformation }
    -- a variant of dirEncryptedTransformation which accepts
    -- algorithm and/or key source qualifier(s) of type
    -- QualifierType
}
enciphered PROTECTION-MAPPING ::=
{   -- general encipherment
    SECURITY-TRANSFORMATION { qualEncryptedTransformation }
    -- a variant of dirEncryptedTransformation which accepts
    -- algorithm and/or key source qualifier(s) of type
    -- QualifierType
}
signed PROTECTION-MAPPING ::=
{   -- general digital signature
    SECURITY-TRANSFORMATION { qualSignedTransformation }
    -- a variant of gulsSignedTransformation which accepts
    -- algorithm and/or key source qualifier(s) of type
    -- QualifierType
}
sealed PROTECTION-MAPPING ::=
{   -- sealed under an integrity mechanism
    SECURITY-TRANSFORMATION { qualSealedTransformation }
    -- a variant of gulsSignedTransformation which accepts
    -- algorithm and/or key source qualifier(s) of type
    -- QualifierType
}
```

I.3 Use of PROTECTED notation with transformation qualifier – Example 2

Following is an illustration of the use of the PROTECTED-Q parameterized type, using security association identifier as a qualifier to a "confidentiality" protection requirement (see E.4).

Prior to the use of the "confidentiality" protection on data, an externally established security association is established between the two communicating systems. This establishes the security transformation and the static parameters needed to control its operation, to provide the required confidentiality protection (i.e. the algorithm, mode of operation and keys need). This may be achieved, for example, through use of an OSI application layer protocol which uses the SESE to support the necessary security exchange. In addition to establishing the static parameters, this security association establishment protocol establishes a security association identifier, sa-id, which can be used in both the encoding and decoding systems to refer to the set of static parameters.

The specification that a data item of type ClearInfo is to be "confidentiality" protected using the static parameters identified by pc-id would be of the form:

```
PROTECTED-Q { ClearInfo, confidentiality, sa-id }
```

In this example, the qualifier is of type:

```
SecurityAssociationId ::= ExternalSAID
```

as defined in the Notation ASN.1 module Annex A.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:38 1999

I.4 Example of use of security exchange and PROTECTED notation in combination

In this example, system A sends an access request to system B using an access control certificate. The access control certificate is protected against unauthorized use by the method described in Annex B of ITU-T Rec. X.812 | ISO/IEC 10181-3 (Access Control Framework). The access control certificate contains a protection value (PV) which relates to a control value (CV) through the following relationship:

$$PV = OWF\ (CV),$$

where OWF represents a one-way function. The knowledge of the CV proves the ownership of the access control certificate. This means that the CV needs to be sent enciphered to B. Assume that both A and B have public key pairs. It is then necessary to send CV enciphered under the public key of B. In addition, the access control certificate and the request are also sent, sealed under the private key of A.

This requirement can be satisfied by defining a security exchange which conveys the necessary security information, with the access request (typically a presentation data value from an application-specific ASE) embedded in the security exchange. The security exchange definition could be as follows:

```
boundAccessControlCert  SECURITY-EXCHANGE ::=
{
    SE-ITEMS            { boundACC }
    IDENTIFIER                  { ... object identifier ... }
}
boundACC    SEC-EXCHG-ITEM  ::=
{
    ITEM-TYPE                   PROTECTED { SealedSequence, sealed }
    ITEM-ID 1
}
SealedSequence ::= SEQUENCE
{
    accessControlCert   AccessControlCert,
    encipheredCV        EncipheredCV,
    accessRequest       EMBEDDED PDV
            -- The access request PDU is embedded here
}
AccessControlCert       ::= PROTECTED {...certificate contents..., signed }

EncipheredCV            ::= PROTECTED { BIT STRING, encrypted }
```


Copyright by the International Organization For Standardization
Thu Jun 03 07:51:39 1999

Annex J

Bibliography

(This annex does not form an integral part of this Recommendation | International Standard)

- ISO 8730:1990, Banking – *Requirements for message authentication (wholesale).*
- CCITT Recommendation X.227 (1992), *Connection-oriented protocol specification for the Association Control Service Element.*

  ISO 8650-1:1988, *Information processing systems – Open Systems Interconnection – Protocol specification for the Association Control Service Element.*
- CCITT Recommendation X.736 (1992) | ISO/IEC 10164-7:1992, *Information technology – Open Systems Interconnection – Systems Management: Security alarm reporting function.*
- CCITT Recommendation X.740 (1992) | ISO/IEC 10164-8:1993, *Information technology – Open Systems Interconnection – Systems Management: Security audit trail function.*
- ITU-T Recommendation X.813[2)] | ISO/IEC 10181-4 ...[2)], *Information technology – Open Systems Interconnection – Security frameworks for open systems: Non-repudiation framework.*
- ITU-T Recommendation X.814[2)] | ISO/IEC 10181-5 ...[2)], *Information technology – Open Systems Interconnection – Security frameworks for open systems: Confidentiality framework.*
- ITU-T Recommendation X.815[2)] | ISO/IEC 10181-6 ...[2)], *Information technology – Open Systems Interconnection – Security frameworks for open systems: Integrity framework.*

2) Presently at the stage of draft.

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:39 1999

4851903 0660931 059

© ISO/IEC

ICS 35.100

Descriptors: data processing, information interchange, network interconnection, open systems interconnection, application layer, communication procedure, security techniques, surveys, models, notations.

Price based on 54 pages

Copyright by the International Organization For Standardization
Thu Jun 03 07:51:39 1999