INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.813

(06/98)

SERIES Q: SWITCHING AND SIGNALLING

Specifications of Signalling System No. 7 – Q3 interface

Security Transformations Application Service Element for Remote Operations Service Element (STASE-ROSE)

ITU-T Recommendation Q.813

(Previously CCITT Recommendation)

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.849
General	Q.700
Message transfer part (MTP)	Q.701–Q.709
Signalling connection control part (SCCP)	Q.711–Q.719
Telephone user part (TUP)	Q.720–Q.729
ISDN supplementary services	Q.730–Q.739
Data user part	Q.740–Q.749
Signalling System No. 7 management	Q.750–Q.759
ISDN user part	Q.760–Q.769
Transaction capabilities application part	Q.770–Q.779
Test specification	Q.780–Q.799
Q3 interface	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
General	Q.850–Q.919
Data link layer	Q.920–Q.929
Network layer	Q.930–Q.939
User-network management	Q.940–Q.949
Stage 3 description for supplementary services using DSS 1	Q.950–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION Q.813

SECURITY TRANSFORMATIONS APPLICATION SERVICE ELEMENT FOR REMOTE OPERATIONS SERVICE ELEMENT (STASE-ROSE)

Summary

This Recommendation provides specifications to support security transformations, such as encryption, hashing, sealing and signing, focusing on whole Remote Operations Service Element (ROSE) Protocol Data Units (PDUs). Security transformations are used to provide various security services such as authentication, confidentiality, integrity and non-repudiation. This Recommendation describes an approach to the provisioning of security transformations that is implemented in the application layer and requires no security-specific functionality in any of the underlying OSI stack layers.

Source

ITU-T Recommendation Q.813 was prepared by ITU-T Study Group 4 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 26th of June 1998.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Recommendation Q.813

SECURITY TRANSFORMATIONS APPLICATION SERVICE ELEMENT FOR REMOTE OPERATIONS SERVICE ELEMENT (STASE-ROSE)

(Geneva, 1998)

1 Scope, Purpose and Application

1.1 Scope

Security Transformations (ST) are used to provide various security services such as peer entity authentication, data origin authentication, confidentiality, integrity and non-repudiation. Security transformations include encryption, hashing, digital seals and digital signatures.

This Recommendation supports security services for ROSE PDUs within the application layer. It is independent of the underlying communications protocol stack. This Recommendation defines a new Application Service Element (ASE) called Security Transformations Application Service Element for ROSE (STASE-ROSE), which resides between the ROSE and the Presentation Layer in the OSI Protocol Stack. This Recommendation provides an approach for performing Security Transformations (ST) that imposes no requirements on any of the 6-lower layers of the communications stack. This is in contrast to methods [e.g. Generic Upper Layers Security (GULS)] that support security transformations through embedded functionality in the communications stack at the presentation layer.

This Recommendation further provides for peer entity authentication at association set-up time; for the negotiation of security parameters (such as security algorithms) that will be used in the course of the association; and for dynamic update, in the course of the association, of security parameters that are used for individual protocol data units.

The method presented in this Recommendation could be adapted for ASEs other than ROSE that interact directly with the presentation layer. However, this Recommendation focuses on ROSE and does not cover any possible extensions or generalizations.

How the actual security transformations are performed (e.g. producing and verifying digital signatures) is a local matter outside the scope of this Recommendation. In particular, the use of a generic security module, such as the Generic Security Service – Application Programming Interface (GSS-API) for performing security transformations is a local matter. Nevertheless, while this Recommendation does not mandate the use of GSS-API, it provides the necessary framework for using GSS-API together with STASE-ROSE (see Appendix III).

Key management is an important component of a security infrastructure. This Recommendation supports the exchange of information related to cryptographic keys. However, a general framework for key management is outside the scope of this Recommendation.

1.2 Purpose

The purpose of this Recommendation is to protect whole ROSE PDUs.

Recommendation Q.812 specifies File Transfer Administration and Management (FTAM), Common Information Management Application Service Element (CMISE) and X.500 Directory in the application layer for the Q3 and X interfaces of the Telecommunications Management Network (TMN). X.500 and CMISE use the services of the Remote Operation Service Element (ROSE). This Recommendation addresses the security of ROSE Protocol Data Units (PDUs). While this Recommendation is motivated by the need to secure TMN interactions or message exchanges, it can be used to provide security for any application that uses ROSE.

1.3 Application

This Recommendation applies to ROSE-based applications such as user applications that use CMISE or X.500 Directory. Providing protection for CMIP PDUs is a major goal of this Recommendation. Since CMIP is based on the 1988 version of ROSE (see Recommendations X.219 and X.229), this Recommendation also focuses on that version, rather than the 1994 version (see Recommendations X.880, X.881 and X.882). Therefore this Recommendation may not apply to the current version of Recommendation X.500 which is based on the 1994 version of ROSE.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

2.1 Normative references

– ITU-T Recommendation M.3010 (1996), *Principles for a telecommunications management network.*

– ITU-T Recommendation Q.811 (1997), *Lower layer protocol profiles for the Q3 and X interfaces.*

– ITU-T Recommendation Q.812 (1997), *Upper layer protocol profiles for the Q3 and X interfaces.*

– ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic reference model: The Basic Model.*

– CCITT Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN.1).*

– ITU-T Recommendation X.210 (1993) | ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic reference model: Conventions for the definition of OSI services.*

– ITU-T Recommendation X.217 (1995) | ISO/IEC 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the association control service element.*

– CCITT Recommendation X.219 (1988), *Remote Operations: Model, notation and service definition.*

– ITU-T Recommendation X.227 (1995) | ISO/IEC 8650-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the association control service element: Protocol specification.*

– CCITT Recommendation X.229 (1988), *Remote operations: Protocol specification.*

– ITU-T Recommendation X.500 (1997) | ISO/IEC 9594-1:1997, *Information technology – Open Systems Interconnection – The directory: Overview of concepts, models and services.*

– ITU-T Recommendation X.509 (1997) | ISO/IEC 9594-8:1997, *Information technology – Open Systems Interconnection – The directory: Authentication framework.*

– ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

– ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*

– ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*

– ITU-T Recommendation X.683 (1997) | ISO/IEC 8824-4:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Parametrization of ASN.1 specifications.*

– ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1:1998, *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.710 (1997) | ISO/IEC 9595:1998, *Information technology – Open Systems Interconnection – Common management information service.*

– ITU-T Recommendation X.711 (1997) | ISO/IEC 9596-1:1998, *Information technology – Open Systems Interconnection – Common management information protocol: Specification.*

– ISO/IEC 9979:1991, *Data cryptographic techniques – Procedures for the registration of cryptographic algorithms.*

2.2 Informative references

– ITU-T Recommendation X.803 (1994) | ISO/IEC 10745:1995, *Information technology – Open Systems Interconnection – Upper layers security model.*

– ITU-T Recommendation X.830 (1995) | ISO/IEC 11586-1:1996, *Information technology – Open Systems Interconnection – Generic Upper layers security: Overview, models and notation.*

– ITU-T Recommendation X.831 (1995) | ISO/IEC 11586-2:1996, *Information technology – Open Systems Interconnection – Generic Upper layers security: Security Exchange Service Element (SESE) service definition.*

– ITU-T Recommendation X.832 (1995) | ISO/IEC 11586-3:1996, *Information technology – Open Systems Interconnection – Generic Upper layers security: Security Exchange Service Element (SESE) protocol specification.*

– ITU-T Recommendation X.833 (1995) | ISO/IEC 11586-4:1996, *Information technology – Open Systems Interconnection – Generic Upper layers security: Protecting transfer syntax specification.*

– ISO/IEC 9798-3:1993, *Information technology – Security techniques – Entity authentication mechanisms – Part 3: Entity authentication using a public key algorithm.*

– ISO/IEC 11770-1:1996, *Information technology – Security techniques – Key management – Part 1: Framework.*

– ANSI X3.92-1981, Data Encryption Algorithm.

– ANSI X3.106-1983, Data Encryption Algorithm – Modes of Operation.

– NBS FIPS PUB 46-1, Data Encryption Standard, *National Bureau of Standards*, US Department of Commerce, Jan. 1988.

– NBS FIPS PUB 74, Guidelines for Implementing and Using the NBS Data Encryption Standard, *National Bureau of Standards*, US Department of Commerce, April 1981.

– NBS FIPS PUB 81, DES Modes of Operation, *National Bureau of Standards*, US Department of Commerce, Dec. 1980.

– NIST FIPS PUB 46-2, Data Encryption Standard, *National Institute of Standards and Technology*, US Department of Commerce, Dec. 1993.

– NIST FIPS PUB 180-1, Secure Hash Standard, *National Institute of Standards and Technology*, US Department of Commerce, May 1994.

– NIST FIPS PUB 186, Digital Signature Standard, *National Institute of Standards and Technology*, US Department of Commerce, May 1995.

– IETF RFC 2078, Generic Security Service Application Program Interface, Version 2, *Internet Engineering Task Force*, January 1997.

3 Definitions

This Recommendation defines the following terms.

3.1 association-initiating-application-entity; association-initiator: The application entity that initiates the application-association.

3.2 association-responding-application-entity; association-responder: The application-entity that responds to the initiation of an application-association by another application-entity.

3.3 sending-application-entity; sender: The application-entity that sends the APDU to the receiving application-entity.

3.4 receiving-application-entity; receiver: The application-entity that receives the APDU from the sending application-entity.

3.5 requester: The application-entity that issues a STASE-ROSE transfer primitive.

3.6 acceptor: The application-entity that receives the indication primitive.

3.7 STASE-ROSE: An application service element residing between the OSI Presentation Layer and ROSE which provides the transformations necessary for secure transfer of ROSE PDUs.

3.8 Secure Transfer: A mechanism to provide for the transfer of application-protocol-data-units (APDUs) between open systems in a secure manner.

3.9 STASE-ROSE-user; SR-user: The application service element that uses the services of STASE-ROSE. The Remote Operations Service Element (ROSE) is the only user of STASE-ROSE services.

3.10 STASE-ROSE-provider; SR-provider: The provider of the Security Transformation Application Service Element for ROSE.

3.11 ACSE-provider: The provider of the Association Control Service Element.

This Recommendation uses the definitions of security services and security mechanisms as specified in Recommendations X.800 and M.3016.

4 Abbreviations

This Recommendation uses the following abbriviations.

AARE	ACSE Association Response
AARQ	ACSE Association Request
ACSE	Association Control Service Element
AE	Application Entity
APDU	Application Protocol Data Unit
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CBC	Cipher Block Chaining
CMIP	Common Management Information Protocol
CMISE	Common Management Information Service Element
DER	Distinguished Encoding Rules
DES	Digital Encryption Standard
EBCDIC	Extended Binary Coded Decimal Interchange Code
FIPS PUB	Federal Information Processing Standards Publication
FTAM	File Transfer Administration and Management
GSS-API	Generic Security Service – Application Programming Interface
GULS	Generic Upper Layers Security
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IS	International Standard
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
MAC	Message Authentication Code
MD	Message Digest
NBS	National Bureau of Standards
NIST	National Institute for Standards and Technology
PDU	Protocol Data Unit
PKCS	Public Key Cryptography Standard

QoP	Quality of Protection
RFC	Request for Comments
ROSE	Remote Operations Service Element
RSA	Rivest Shamir Adelman
SR	STASE-ROSE
ST	Security Transformation
STASE-ROSE	Security Transformations Application Service Element – ROSE

5 Overview

This clause provides a high level overview of STASE-ROSE. It focuses on what STASE-ROSE does, deferring to subsequent subclauses further elaboration on how STASE-ROSE does it.

STASE-ROSE is an application service element residing between the OSI Presentation Layer and ROSE which provides the transformations necessary for secure transfer of ROSE PDUs. In addition, STASE-ROSE provides a means for exchanging information regarding the security being provided. STASE-ROSE is invoked by a request from ROSE on the transmitting side, and it provides an indication to ROSE on the receiving side. Both the request and the indication contain the ROSE PDU being protected, as well as (optionally) information regarding the type of the security being provided.

The ST and security information exchange aspects of STASE-ROSE are discussed below.

5.1 Security transformations

STASE-ROSE protects ROSE PDUs by applying selected Security Transformations (ST) to whole ROSE PDUs encoded with the Distinguished Encoding Rules (DER). In particular, STASE-ROSE supports the following STs:

- **confidential**: The DER-encoded ROSE PDU is encrypted for privacy protection with a symmetric key encryption algorithm.
- **public enciphered**: The DER-encoded ROSE PDU is encrypted for privacy protection with a public key encryption algorithm.
- **hashed**: STASE-ROSE computes a hash-based Message Authentication Code (MAC) of the DER-encoded ROSE PDU and a secret password and appends the result to the ROSE PDU for integrity protection.
- **sealed**: STASE-ROSE computes the digital seal of the DER-encoded ROSE PDU and appends the result to the ROSE PDU for integrity protection.
- **signed**: STASE-ROSE computes the digital signature of the DER-encoded ROSE PDU and appends the result to the ROSE PDU for non-repudiation protection.
- **confidential signed**: STASE-ROSE computes the digital signature of the DER-encoded ROSE PDU and appends the result to the encrypted (see "confidential" above) ROSE PDU for non-repudiation and privacy protection.
- **confidential hashed**: STASE-ROSE computes the MAC of the DER-encoded ROSE PDU and appends the result to the encrypted (see "confidential" above) ROSE PDU for integrity and privacy protection.

- **confidential sealed**: STASE-ROSE computes the digital seal of the DER-encoded ROSE PDU and appends the result to the encrypted (see "confidential" above) ROSE PDU for integrity and privacy protection.

STASE-ROSE can also pass through ROSE PDUs in the **clear**, without any encoding or any STs.

5.2 Security information exchange

The following messages, exchanged between STASE-ROSE entities, or between ROSE and STASE-ROSE specify which of the STs listed above is being used to protect the ROSE PDU:

- ROSE invokes STASE-ROSE on the originating side;
- the originating STASE-ROSE sends a STASE-ROSE PDU to a peer STASE-ROSE entity on the receiving side;
- STASE-ROSE on the receiving side provides an indication to ROSE.

Knowledge of which STs are performed is mandatory, but not sufficient for proper communications. Indeed, both sides need to know which algorithms are being used as well as the values of any parameters (e.g. encryption keys, initialization vectors) that are being used. This Recommendation provides several default values and mechanisms that require only the bare minimum of security-related information exchange. It also provides facilities for negotiating at association setup time which algorithms will be supported and for changing and specifying such information dynamically for every ROSE PDU.

5.2.1 Security information default values

The use of the negotiation capability of STASE-ROSE is optional. If two communicating entities do not use the negotiation capability, they must have an agreement on a set of security parameters, such as security algorithms, that will be used in the course of the association. The two parties can agree on any set of values for such parameters by means outside the scope of this Recommendation.

Unless otherwise agreed between the communicating entities, the following conventions, security algorithms and security mechanisms shall be used:

– The default encryption algorithm for symmetric key encryption shall be the Digital Encryption Standard (DES) in the Cipher Block Chaining (CBC) mode.

– If triple DES is needed, the default shall be Encryption Decryption Encryption (EDE) in the CBC outer feedback mode with three different DES keys.

– If no Initialization Vector (IV) is specified, then the first IV of the association shall consist of 64 zero valued bits, every subsequent IV shall consist of the last 8 bytes of the previous encrypted ROSE PDU.

– The default public key encryption algorithm shall be RSA[1, 2].

– The default hashing algorithm shall be MD5[3].

– The default MAC (for keyed hashing) shall be HMAC[4].

---

[1] RIVEST (R.), SHAMIR (A.) and ADELMAN (L. M.): A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM*, Version 21, No. 2, pp. 120-126, February 1978.

[2] RSA Data Security Inc., PKCS No. 1: RSA Encryption Standard, Version 1.5, November 1993.

[3] RIVEST (R.), IETF RFC 1319: The MD5 Message Digest Algorithm, April 1992.

[4] KRAWCZYK (H.), BELLARE (M.), CANETTI (R.), IETF RFC 2104, HMAC: Keyed-Hashing for Message Authentication, February 1997.

– The default seal shall be the MD5 hash of the DER encoded ROSE PDU encrypted with DES.

– The default digital signature shall be the MD5 hash of the DER encoded ROSE PDU encrypted with RSA and the user's private key.

– Peer entity authentication shall occur at association setup time. The optional authentication Functional Unit (FU) of the ACSE shall be used. Authentication information shall be carried in the calling-authentication-value and responding-authentication-value fields of the authentication FU of the ACSE AARQ and AARE PDUs respectively. The bit strings for the sender-acse-requirements and responder-acse-requirements fields of the authentication FU shall be set to include the authentication FU. The calling-authentication-value and responding-authentication-value fields are of type Authentication-value which is further defined in ISO 8650 as a CHOICE. The CHOICE for the Authentication-value shall be EXTERNAL. The presentation context shall include a reference to the abstract syntax which is used for the EXTERNAL. When the default values and conventions are used, it is not necessary to use the (optional) mechanism-name field of the authentication FU of the ACSE PDUs.

– If peer entity authentication with public key encryption is required it shall consist of:

  • the sender's unique identifier;

  • the receiver's unique identifier;

  • a time stamp, consisting of GeneralizedTime;

  • optionally, a symmetric encryption key that will be used by the sender during the association, encrypted with the receiver's public key;

  • a digital signature of the preceding fields, signed with the sender's private key;

  • optionally, the sender's public key certificate.

Unless otherwise agreed upon between the communicating entities by means outside the scope of this Recommendation, the digital signature will be computed using MD5 for hashing and RSA for public key encryption. The syntax for this authenticator is given in 5.2.1.1. The (optional) publicly encrypted symmetric encryption keys in the AARQ and AARE messages can be different, allowing the initiator and the responder of the association to use different keys in the course of the association.

The authenticators proposed in this Recommendation, as well as other parts of this Recommendation make use of time stamps. Systems clocks may be set back if they move too fast, or they might be set back due to some malfunction. This Recommendation requires that regardless of such events, consecutive time stamps produced by a system be monotonically increasing. Appendix I illustrates a possible construction of such monotonically increasing time.

This Recommendation does not specify which of the two authenticators to use. The communicating entities can agree by means outside the scope of this Recommendation on the authenticator they will use.

While this Recommendation defines two authenticators, the communicating parties may agree to use another authenticator. In such case, the ASN.1 syntax for the authenticator must be specified; it is also necessary to assign and register an abstract syntax name for the authenticator, to be used in presentation layer negotiation.

The communicating entities may agree by means outside the scope of this Recommendation on a different set of default values.

5.2.1.1 Abstract syntax for public key authenticator

The following module for public key authentication, to be carried in the Authentication-value field of the authentication FU of ACSE when peer entity authentication with public key encryption is required. This module supports two distinct cases:

– all the fields in the authenticator are specified explicitly;

– both communicating entities use GSS-API and the authentication information is transported as an octet string which is interpreted locally by GSS-API.

```
STASE-ROSE-Authentication-value {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0)
abstractSyntax(1) stase-authentication-value(0) }
DEFINITIONS IMPLICIT TAGS ::= BEGIN
-- EXPORTS everything

IMPORTS

SenderId, ReceiverId, Signature, SignatureCertificate
FROM Secure-Remote-Operations-APDUs {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0) stase-
data(2)};

Authentication-value ::= CHOICE {
                explicit                        [0] ExplicitAuthenticator,
                 gssAuthenticator               [1] GssAuthenticator,
        -- to be used only if the two communicating entities use GSS-API
                ...
                 }

ExplicitAuthenticator ::= SEQUENCE {
                                senderId        [0] SenderId,
                                receiverId      [1] ReceiverId,
                                time            [3] GeneralizedTime,
                                encryptedSymmetricKey [4] INTEGER    OPTIONAL,
        -- a symmetric encryption key encrypted with the receiver's public key
                                signature       [5] Signature,
        -- the sender's signature of the preceding fields encoded as ASCII characters
                                certificate     [6] SignatureCertificate        OPTIONAL
        -- the sender's public key certificate for the key used for the signature
        }

GssAuthenticator ::= SEQUENCE {
                        gssMechanism            [0] OBJECT IDENTIFIER OPTIONAL,
                        gssInitialContextToken  [1] OCTET STRING
                        }

END
```

5.2.1.2 Abstract syntax name

This Recommendation assigns the ASN.1 object identifier value:

{itu-t recommendation q(17) q813(813) stase(1) abstractSyntax(1) stase-authentication-value(0) }

as an abstract syntax name for the set of all presentation data values each of which is a value of ASN.1 type.

STASE-ROSE-Authentication-value.Authentication-value.

The corresponding object descriptor value shall be "STASE-ROSE-Authenticator".

5.2.2 Negotiation of security algorithms

The default values, whether as specified in this Recommendation or otherwise agreed upon between the communicating entities can be superseded dynamically as described below.

At association setup time the communicating entities can negotiate which algorithms they will support during the course of the association. In the user information field of the ACSE association request the association originator can, optionally, include:

- a set of acceptable symmetric encryption algorithms;
- a set of acceptable public encryption algorithms;
- a set of acceptable hashing algorithms;
- a set of acceptable keyed-hashing (MAC) algorithms;
- a set of acceptable digital seals algorithms;
- a set of acceptable digital signatures algorithms.

The AARE will contain sets of acceptable algorithms that are subsets of those in the AARQ. If this optional negotiation capability is used by the association originator in the AARQ, then it must be used by the association responder. An association response that does not use the optional negotiation capability is equivalent to a response with null values (i.e. empty sets). If any of the sets listed above is missing in the AARQ, then only the default value for the corresponding algorithm will be supported. If any of the sets listed above is present in the AARQ, then it should be present in the AARE, otherwise, the absence of such a set in the AARE is equivalent to empty set. If any of the sets listed above is present in the AARE but is empty, then no algorithm for the corresponding ST can be used, and therefore that ST will not be used in the course of the association. If any of the sets listed above contains exactly one element (in the AARE), then that element designates the default value for the corresponding ST for the duration of the association. If any of the sets listed above contains more than one element (in the AARE) and one of those elements corresponds to the default value for the corresponding ST (as specified in this Recommendation or as otherwise agreed upon between the two communicating entities) then that element designates the default value for the corresponding ST for the duration of the association. If an entity receiving a message does not concur with the choice of algorithms it can terminate the association. If any of the sets listed above is present in the AARE and contains elements that are not contained in the AARQ, then an error is declared. In case an error is detected the association is released. The process of negotiating security algorithm is summarized in Table 5-1 and is illustrated in Appendix II.

Table 5-1/Q.813 – Negotiated algorithms

<table>
<tr><td colspan="4" rowspan="4">Set of acceptable algorithms in AARE</td><td colspan="4">Set of acceptable algorithms in AARQ</td></tr>
<tr><td colspan="3">Present</td><td>Absent</td></tr>
<tr><td colspan="2">Non-empty</td><td rowspan="2">Empty (NULL)</td><td rowspan="2"></td></tr>
<tr><td>2 or more elements</td><td>1 element</td></tr>
<tr><td rowspan="4">Present</td><td rowspan="3">Non-empty (subset of AARQ set)</td><td rowspan="2">2 or more elements</td><td>Does not include predefined default</td><td>The AARE algorithms, no défaut</td><td>Error</td><td>Error</td><td>Error</td></tr>
<tr><td>Includes predefined default</td><td>The AARE algorithms, with predefined default</td><td>Error</td><td>Error</td><td>Error</td></tr>
<tr><td colspan="2">1 element</td><td>The AARE algorithms are the default</td><td>The selected algorithm is the default</td><td>Error</td><td>Error</td></tr>
<tr><td colspan="3">empty (NULL)</td><td>None</td><td>None</td><td>None</td><td>None</td></tr>
<tr><td colspan="4">Absent</td><td>None</td><td>None</td><td>None</td><td>Only the predefined default; if there is no predefined default, then none</td></tr>
</table>

In addition to negotiation of security algorithms STASE-ROSE also supports the negotiation of various encryption parameters:

- identification of symmetric encryption keys that can be used;
- identification of public keys that can be used;
- identification of seal keys that can be used;
- identification of password IDs that can be used;
- specification of sizes of public keys that can be used;
- specification of public keys that can be used;
- sender's secret key.

Unlike encryption algorithms, this Recommendation does not provide any default values for any of those parameters.

STASE-ROSE further supports the conveying of additional encryption parameters:

- specification of an initialization vector (for DES encryption) that shall be used;
- specification of the feedback bits that shall be used for k-bit output feedback mode or k-bit cipher feedback modes of DES;
- specification of a key digest for the verification of a public key;

- specification of a sequence number for the current message;
- specification of a timestamp for the current message;
- specification of a symmetric key encrypted with a symmetric key encryption key;
- specification of a symmetric key encrypted with a public key (of the receiver);
- specification of a key encryption key identifier;
- provisioning of X.509 certificates or certification paths of the sender's public keys that can be used without restrictions;
- provisioning of X.509 certificates or certification paths of the sender's public keys that can be used for encryption only;
- provisioning of X.509 certificates or certification paths of the sender's public keys that can be used only for digital signatures;
- specification of a symmetric session key encrypted with the receiver's public key and signed with the sender's secret key.

The values of those parameters can be provided by either party during association setup, but they are not subject to negotiation. For instance, each party can provide its own public key certificate(s) to the other party.

By the end of the negotiation phase (i.e. association setup) the two entities have an agreement as to which STs and which algorithms for those STs they will support. In some cases, but not necessarily all cases, they also have an agreement on the default algorithms for some or all the STs they have agreed to support. In some cases they may also have an agreement on the values of some or all of the security parameters.

STASE-ROSE supports the specification and use of different algorithms for different PDUs. It further allows each communicating entity to use different algorithms from those used by the other entity for the same STs. This is relevant, of course, only if more than one algorithm have been agreed upon for a given ST during the negotiation phase. STASE-ROSE provides some straightforward rules regarding the dynamic specification of algorithms in the course of an association:

- When exactly one algorithm has been agreed upon for a given ST, the algorithm for that ST will not be specified again in the course of the association.
- If more than one algorithm have been agreed upon for a given ST, and a default algorithm for that ST has been agreed upon, then it is optional to specify the algorithm for that ST, subsequent to the association setup. The default algorithm will take effect for the ST if not specified, subsequent to the association setup.
- If more than one algorithm have been agreed upon for a given ST, and no default algorithm for that ST has been agreed upon, then each communicating party must specify the algorithm it uses for that ST the first time it sends a message that uses this ST. Each communicating entity may, but doesn't have to specify the algorithm it uses for that ST in subsequent messages.
- If more than one algorithm have been agreed upon for a given ST, and no default algorithm for that ST has been agreed upon, then every time the corresponding ST is used by a communicating entity, after the first time, the algorithm that was last used for this ST by that entity is the default algorithm for that entity.

The negotiation procedure uses the EncryptionParametersSelection parameter defined in 5.3. This parameter is passed to the ACSE as user data and it shall be carried in the User Information field in the ACSE AARQ and AARE PDUs.

5.3 Abstract syntax for negotiation of security parameters

The following module for negotiation of security parameters, to be used in the UserInformation field of ACSE is registered.

```
STASE-A-ASSOCIATE-Information {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0) stase-
userinfo(1)}
DEFINITIONS IMPLICIT TAGS ::= BEGIN
-- EXPORTS everything

IMPORTS

SenderId, ReceiverId, Signature, KeyId , PublicKeyCertificate, EncryptionCertificate, SignatureCertificate,
EncryptedAuthenticatedSymmetricKey
FROM Secure-Remote-Operations-APDUs {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0) stase-
data(2)};

EncryptionParametersSelection ::= SET
        {symmetricKeyIds        [0] SET OF KeyId                        OPTIONAL,
         publicKeyIds           [1] SET OF KeyId                        OPTIONAL,
         sealKeyIds             [2] SET OF KeyId                        OPTIONAL,
        signatureKeyIds         [3] SET OF KeyId                        OPTIONAL,
        passwordIds             [4] SET OF KeyId                        OPTIONAL,
        initializationVector    [5] OCTET STRING (SIZE(8))              OPTIONAL,
        feedBackBits            [6] INTEGER (1..63)                     OPTIONAL,
        -- for k-bit output feedback mode or k-bit cipher feedback mode of DES
        symmetricAlgorithms     [7] SET OF OBJECT IDENTIFIER            OPTIONAL,
        publicKeyAlgorithms     [8] SET OF OBJECT IDENTIFIER            OPTIONAL,
        signatureAlgorithms     [9] SET OF OBJECT IDENTIFIER            OPTIONAL,
        sealAlgorithms          [10] SET OF OBJECT IDENTIFIER           OPTIONAL,
        hashAlgorithms          [11] SET OF OBJECT IDENTIFIER           OPTIONAL,
        keyDigest               [12] OCTET STRING (SIZE(8..64))         OPTIONAL,
         -- for verification of public keys
        blockSize               [13] INTEGER                            OPTIONAL,
        -- for square mod-n hashing
        keySizes                [14] SET OF INTEGER                     OPTIONAL,
        -- for RSA
         publicKeys             [15] SET OF SEQUENCE
                        {modulus            INTEGER,
                         exponent           INTEGER
                        }                                               OPTIONAL,
        sequenceNumber          [16] INTEGER            OPTIONAL,
        timeStamp               [17] GeneralizedTime    OPTIONAL,
        encryptedKey            [18] OCTET STRING (SIZE(64..128)) OPTIONAL,
        -- symmetric session key, encrypted with Key-Encryption-Key
        encryptedSymmetricKey   [19] INTEGER            OPTIONAL,
        -- symmetric session key, encrypted with the receiver's public key
        keyEncryptionKey        [20] SEQUENCE (SIZE (1..3)) OF KeyId  OPTIONAL,
        -- one to three symmetric keys used for encrypting a session key
        keyListIds              [21] SET OF KeyListId   OPTIONAL,
        -- list of encryption keys that can be used during the association
        encryptionCertificate   [22] SET OF EncryptionCertificate       OPTIONAL,

-- X.509 certificates or certification paths of the sender's public keys used for encryption only

        signatureCertificate    [23] SET OF SignatureCertificate        OPTIONAL,

-- X.509 certificates or certification paths of the sender's public keys used for digital signatures only --
```

```
        encryptedAuthenticatedSymmetricKeys [24] SET OF EncryptedAuthenticatedSymmetricKey
                                                                        OPTIONAL,

-- symmetric session key, encrypted with the receiver's public key and signed with sender's key--

        macAlgorithms               [25] SET OF OBJECT IDENTIFIER       OPTIONAL,
        publicKeyCertificate        [26] SET OF PublicKeyCertificate    OPTIONAL,

-- X.509 certificates or certification paths of the sender's public keys with no usage restrictions --
        ...
        }

-- EncryptionParametersSelection is optionally used during association setup to negotiate which algorithms and other
-- encryption parameters will be supported during the association. It is not used in STASE-ROSE PDUs. --

KeyListId ::= CHOICE    {identifier     OBJECT IDENTIFIER,
                         name           GraphicString,
                         number         INTEGER
                        }

END
```

5.3.1 Abstract syntax name

This Recommendation assigns the ASN.1 object identifier value.

{itu-t recommendation q(17) q813(813) stase(1) abstractSyntax(1) stase-userinfo(1) }

as an abstract syntax name for the set of all presentation data values each of which is a value of ASN.1 type.

STASE-A-ASSOCIATE-Information.EncryptionParametersSelection

The corresponding object descriptor value shall be "STASE-ROSE-User-Information".

6 Model

In the OSI environment, communication between application-processes is represented in terms of communication between a pair of application entities (AEs) using the presentation service. Communication between some application entities may require secure transfer of application-protocol-data-units (APDUs).

APDUs sent by one AE (the sender) are received by the other AE (the receiver). Secure transfer ensures that APDUs transferred by the sender can be correctly verified for integrity, and/or checked for non-repudiation, and/or understood only by the intended receiver of a given APDU. Secure transfer involves security transformations (e.g. encryption) of APDUs from the sending-application-entity before transferring them and performing the reverse security transformations (e.g. decryption) before delivering the APDUs to the receiving-application-entity. STASE-ROSE only deals with the secure transfer of Remote Operations Service Element (ROSE) application-protocol-data-units.

Secure transfer is carried out within the context of the application-association. An application-association defines the relationship between a pair of AEs, and is formed by the exchange of application-protocol-control-information through the use of presentation layer services. The AE that initiates the association is called the association-initiating entity, or the association-initiator, while the AE that responds to the initiation of an application-association by another AE is called the application-responding AE, or the association-responder.

The functionality of an AE is factored into one application process and a set of application-service-elements (ASEs). Each ASE may itself be factored into a set of (more primitive) ASEs. The interaction between AEs is described in terms of their use of ASEs.

The specific combination of a application process and the set of ASEs which comprise an AE are defined by the application context.

Figure 1 illustrates an example of an application context involving STASE-ROSE (only the peer entity relationship between the top-most ASEs is shown).

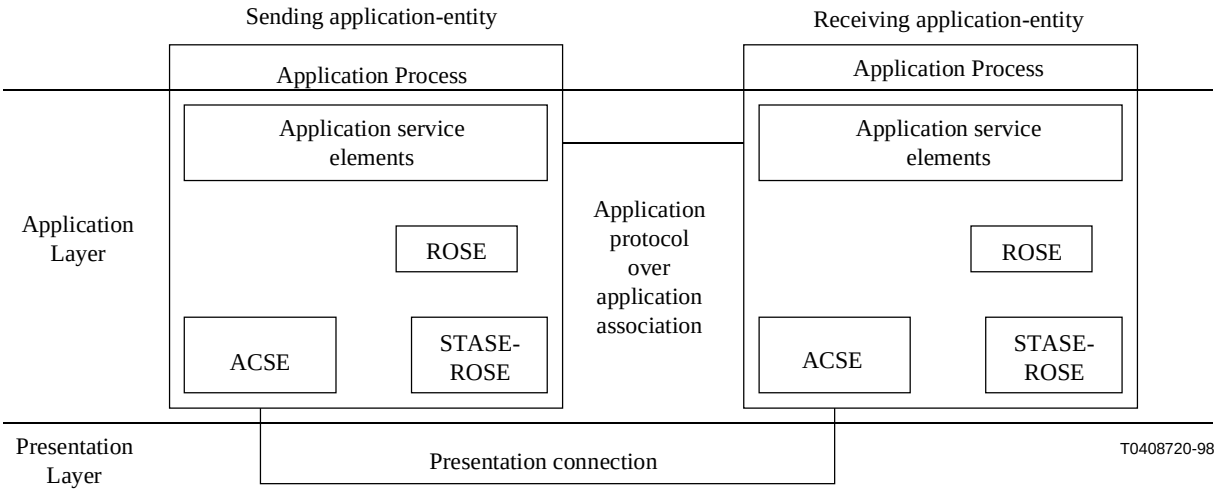

Figure 1/Q.813 – Mandatory ASEs when STASE-ROSE is used

The ASEs available to an application process require communication over an application-association. The control of application-association is performed by the application processing the services provided by the Association Control Service Element (ACSE).

Figure 2 depicts the ASEs that must be present when STASE-ROSE is used. It shows the ASEs for Telecommunications Management Network (see Recommendation M.3010) involving the Common Management Information Service Element (CMISE) in addition to ACSE, ROSE and STASE-ROSE.

This Recommendation can be used for any application that uses ROSE. However, for simplicity of presentation, the text in this clause refers only to CMISE as the ASE that uses ROSE.

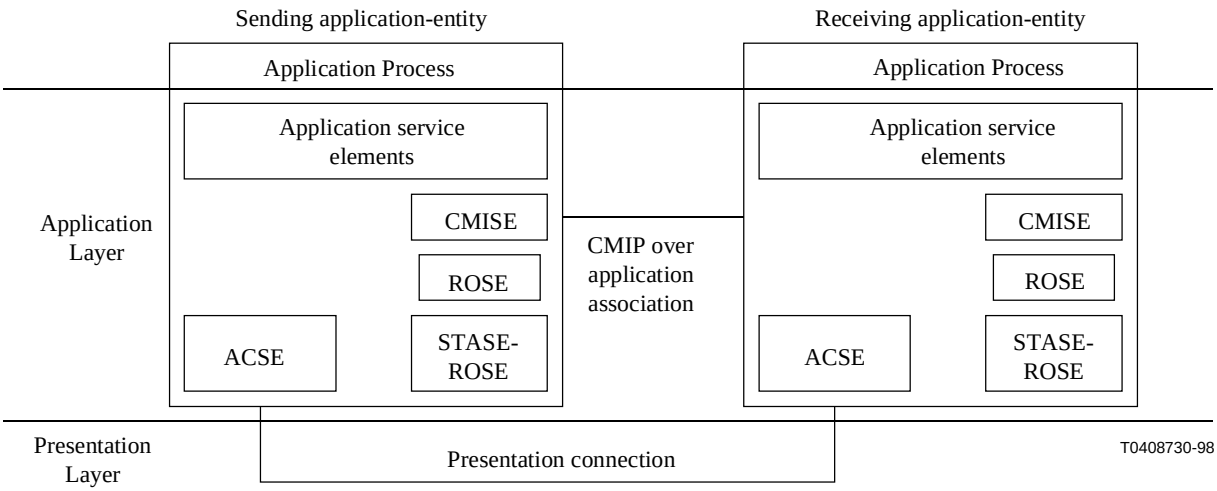

Figure 2/Q.813 – ASEs present when STASE-ROSE is used for securing CMISE

7 Service overview

7.1 Association services

This Recommendation does not provide separate services for the establishment and release of application associations. The application process of the application context involving STASE-ROSE relies on the services of Recommendation X.217 for the control of application associations.

During the association establishment phase, various ASEs in the application context may exchange initialization information to establish an association using ACSE. The application context, presentation and session requirements are conveyed using parameters of the A-ASSOCIATE service.

The A-RELEASE and A-ABORT services of Recommendation X.217 are used for the termination of an association. These may be invoked by either user-elements.

7.2 STASE-ROSE services

The STASE-ROSE service is listed in Table 7-1.

The following is a brief description of the STASE-ROSE service:

– SR-TRANSFER

  The SR-TRANSFER service enables ROSE to initiate secure transfer of ROSE PDUs to a peer ROSE.

Table 7-1/Q.813 – STASE-ROSE services

Service	Type
SR-TRANSFER	Non-confirmed

7.3 Relationship to presentation services

The STASE-ROSE service requires access to P-DATA service.

A named abstract syntax with a compatible transfer syntax (negotiated by the presentation layer) constitutes a presentation context. BER transfer syntax would typically be negotiated during the

establishment of an association with application context which included STASE-ROSE. Even though DER is used by STASE-ROSE to perform encoding before performing security transformations, DER is not part of the application context (unless DER is also used by the presentation layer).

7.4 Service definition

7.4.1 Conventions

This Recommendation defines services for the STASE-ROSE following the descriptive conventions defined in Recommendation X.210. The definition of the STASE-ROSE service includes a table that lists the parameters of the primitives. The presence of parameters in a table is described by one of the following values:

-- Not applicable

M Mandatory

U User option

C Conditional

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

In this Recommendation, the character "**.**" is used to address fields in an ASN.1 type. For example, **a.x** is used to address field **x** in field **a** in the Example1 and Example2 ASN.1 types below.

```
Example1 ::= SEQUENCE {
        a  SEQUENCE {
                x INTEGER,
                y BOOLEAN
        },
        b  INTEGER
}

A ::= SEQUENCE {
                x INTEGER,
                y BOOLEAN
}

Example2 := SEQUENCE {
                a A,
                b INTEGER
}
```

7.4.2 Association services

7.4.2.1 Association establishment

The A-ASSOCIATE service of Recommendation X.217 is invoked by the application process of an application context involving STASE-ROSE to establish an association with a peer application process. Association establishment is the first phase of any instance of secure transfer service activity.

Table 7-2 lists the parameters that are defined by this Recommendation to be the STASE-ROSE specific part of the user information parameter of the A-ASSOCIATE service. This information is specified by the association-initiator and exchanged when establishing an association. Exchange of this initialization information is optional prior to using STASE-ROSE services.

Table 7-2/Q.813 – A-ASSOCIATE user information

Parameter name	Request/Indication	Response/Confirmation
symmetricKeyIds	U	C
publicKeyIds	U	C
sealKeyIds	U	C
signatureKeyIds	U	C
passwordIds	U	C
initializationVector	U	U
feedBackBits	U	U
symmetricAlgorithms	U	C
publicKeyAlgorithms	U	C
signatureAlgorithms	U	C
sealAlgorithms	U	C
hashAlgorithms	U	C
keyDigest	U	U
blockSize	U	U
keySize	U	C
publicKeys	U	U
sequenceNumber	U	U
timeStamp	U	U
encryptedKey	U	U
encryptedSymmetricKey	U	U
keyEncryptionKey	U	U
keyListIds	U	C
publicKeyCertificates	U	U
encryptionCertificates	U	U
signatureCertificates	U	U
encryptedAuthenticatedSymmetricKeys	U	U
macAlgorithms	U	C

The condition C in Table 7-2 is that the parameter is present in the response/confirmation only if it is present in the request/indication. If it is present in the request/indication but not in the response/confirmation, then the response/confirmation is interpreted as if the parameter were present with null value.

The meanings of these parameters and the response expected from the responder is described in greater detail below:

- **symmetricKeyIds**: The set of key identifiers for the keys to be used on this association for symmetric encryption. The application-responder shall respond with the same set or a subset thereof if the symmetricKeyIds parameter is present in A-ASSOCIATE indication.
- **publicKeyIds**: The set of key identifiers for the keys to be used on this association for public key encryption. The application-responder shall respond with the same set or a subset thereof if the publicKeyIds parameter is present in A-ASSOCIATE indication.

- **sealKeyIds**: The set of key identifiers for the keys to be used on this association for sealing. The application-responder shall respond with the same set or a subset thereof if the sealKeyIds parameter is present in A-ASSOCIATE indication.
- **signatureKeyIds**: The set of key identifiers for the keys to be used on this association for digital signature. The application-responder shall respond with the same set or a subset thereof if the signatureKeyIds parameter is present in A-ASSOCIATE indication.
- **passwordIds**: The set of password identifiers for the passwords to be used on this association. The application-responder shall respond with the same set or a subset thereof if the parameter is present in A-ASSOCIATE indication.
- **initializationVector**: The Initialization Vector (IV) to be used for DES encryption in Ciphered Block Chaining (CBC) mode. Each party can use a different IV for the messages it sends.
- **feedBackBits**: The feedback bits to be used for DES encryption in k-bit cipher feedback or b-bit output feedback modes. Each party can use a different value for the feedback bits for the messages it sends.
- **symmetricAlgorithms**: The set of symmetric algorithms the association-initiator is capable of supporting. The association-responder shall respond with the same set or a subset thereof if the symmetricAlgorithms parameter is present in the A-ASSOCIATE indication.
- **publicKeyAlgorithms**: The set of public key algorithms the association-initiator is capable of supporting. The association-responder shall respond with the same set or a subset thereof if the publicKeyAlgorithms parameter is present in the A-ASSOCIATE indication.
- **signatureAlgorithms**: The set of digital signature algorithms that the association-initiator is capable of supporting. The association-responder shall respond with the same set or a subset thereof if the signatureAlgorithms parameter is present in the A-ASSOCIATE indication.
- **sealAlgorithms**: The set of sealing algorithms that the association-initiator is capable of supporting. The association-responder shall respond with the same set or a subset thereof if the sealAlgorithms parameter is present in the A-ASSOCIATE indication.
- **hashAlgorithms**: The set of hashing algorithms that the association-initiator is capable of supporting. The association-responder shall respond with the same set or a subset thereof if the hashAlgorithms parameter is present in the A-ASSOCIATE indication.
- **keyDigest**: This is a message digest (fingerprint) of a public key, it is used to verify the validity of a public key**.**
- **blockSize**: The block size to be used for square mod-n hashing. Each party can choose a different block size for the messages it sends.
- **keySize**: The key size for RSA encryption algorithm. The association-responder shall respond with the same or a different key size if the keySize parameter is present in the A-ASSOCIATE indication.
- **publicKeys**: The set publicKey to be used by the sender on this association. The association-responder may respond with the set of the responder's public keys.
- **sequenceNumber**: The starting sequence number for ROSE PDUs, if the association should be protected against replay and deletion attacks. Each party can choose to start with a different sequence number. If this parameter is present, the STASE-ROSE-provider of any party that supplies it shall assign a sequence number for each STASE-ROSE APDU sent on the application-association.
- **timeStamp**: The time at which the A-ASSOCIATE request was initiated by the association-initiator. The interpretation of this parameter is implementation dependent and is outside the

scope of this Recommendation. If the association-responder sends the timeStamp, its value shall correspond to the time at which the A-ASSOCIATE response primitive was issued.

- **encryptedKey**: A symmetric key used for (part of) the association and encrypted with a symmetric Key Encryption Key (KEK).
- **encryptedSymmetricKey**: A symmetric key used for (part of) the association and encrypted with the receiver's public key.
- **keyEncryptionKey**: Identifies one to three symmetric keys to be used as the symmetric KEK.
- **keyListIds**: The set of identifiers of lists of symmetric encryption keys that the association-initiator proposes to use. The association-responder shall respond with the same set or a subset thereof if the keyListIds parameter is present in the A-ASSOCIATE indication.
- **publicKeyCertificates**: X.509 Certification Path certifying the sender's public key.
- **encryptionCertificates**: X.509 Certification Path certifying the sender's public key which can be used only for encryption.
- **signatureCertificates**: X.509 Certification Path certifying the sender's public key which can be used only for digital signatures.
- **encryptedAuthenticatedSymmetricKeys**: A symmetric key used for (part of) the association and encrypted with the receiver's public key, followed by a timestamp (GeneralizedTime), the sender's ID, receiver's ID, and a signature computed over the ASCII representation of those four fields using the sender's private key.
- **macAlgorithms**: The set of MAC algorithms that the association-initiator is capable of supporting. The association-responder shall respond with the same set or a subset thereof if the macAlgorithms parameter is present in the A-ASSOCIATE indication.

7.4.2.2 Association release

The A-RELEASE service of Recommendation X.217 is invoked by the application process in the application context involving STASE-ROSE to request orderly termination of an association between peer application-entities. This Recommendation does not specify any use of the parameters of A-RELEASE service.

The A-ABORT service is invoked by the application process to request the abrupt termination of the application-association.

7.4.3 SR-TRANSFER service

The SR-TRANSFER service is used by one STASE-ROSE user (ROSE) to transfer a ROSE PDU in a secure manner to the peer STASE-ROSE user (ROSE).

The related service structure consists of two service primitives, as illustrated in Figure 3.

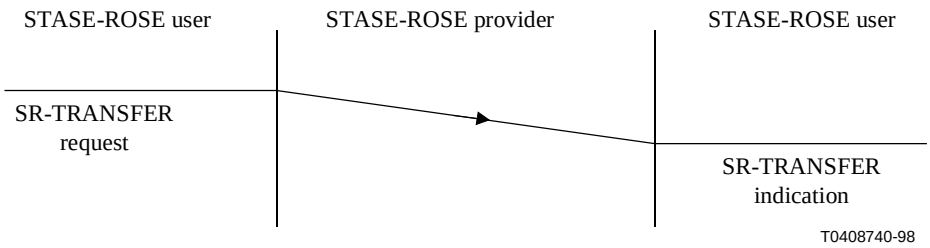

Figure 3/Q.813 – SR-TRANSFER service primitives

7.4.4 SR-TRANSFER parameters

Table 7-3 lists the SR-TRANSFER service parameters.

Table 7-3/Q.813 – SR-TRANSFER parameters

Parameter name	Request	Indication
ROSE-PDU	M	M(=)
Encryption-Type	M	M(=)
Encryption-Parameters	U	C(=)

7.4.4.1 ROSE-PDU

This parameter identifies the ROSE PDU to be transferred. This parameter has to be supplied by the requester of the service and the data values shall be in conformance with the ROSEapdus definition in Recommendation X.229.

7.4.4.2 Encryption-Type

This parameter identifies the type of STs desired by the service user for the current ROSE PDU. The following is a list of valid values for the type (note, usage of default values for encryption parameters is specified in 5.2.1):

- **clear**: No ST is desired.
- **simpleConfidential**: Whole PDU privacy protection using the defaults supported by the service provider.
- **confidential**: Whole PDU privacy protection using the parameters provided in the Encryption-Parameters.
- **simplePublicEnciphered**: Whole PDU privacy protection using default public key supported by the service provider.
- **publicEnciphered**: Whole PDU privacy protection using the public key provided in the Encryption-Parameters.
- **simpleHashed**: Hashing-based MAC of the PDU using the default values.
- **hashed**: Hashing-based MAC of PDU using the parameters provided in the Encryption-Parameters.
- **simpleSealed**: Sealing of the PDU using the default values.
- **sealed**: Sealing of the PDU using the parameters provided in the Encryption-Parameters.
- **simpleSigned**: Digital signature of the PDU using the default values.
- **signed**: Digital signature of the PDU using the parameters provided in the Encryption-Parameters.
- **simpleConfidentialSigned**: Whole PDU privacy protection and digital signature of the PDU using the default values.
- **confidentialSigned**: Whole PDU privacy protection and digital signature of the PDU using the parameters provided in the Encryption-Parameters.
- **simpleConfidentialMAC**: Whole PDU privacy protection and MAC of the PDU using the default values.
- **confidentialMAC**: Whole PDU privacy protection and MAC of the PDU using the parameters provided in the Encryption-Parameters.

- **simpleConfidentialSealed**: Whole PDU privacy protection and seal of the PDU using the default values.
- **confidentialSealed**: Whole PDU privacy protection and seal of the PDU using the parameters provided in the Encryption-Parameters.

7.4.4.3 Encryption-Parameters

This parameter identifies the parameters to be used for the security transformations. The presence of this parameter is dependent on the Encryption-Type selected by the user (as described in the previous subclause).

8 Interaction between application service elements

This clause describes the interactions between application process, ACSE, ROSE, STASE-ROSE, the ROSE user (e.g. CMISE) and presentation layer services at the various stages during communication between two application-entities. Other interactions, that result in the same set of messages exchanged between the communicating systems and the same functionality are possible. Selection of such interactions is a local matter. Annex A narrows this discussion to the case where CMISE is the ROSE user.

8.1 Association establishment

Figure 4 illustrates the sequence of interactions between the application process, various ASEs and the presentation provider during association establishment phase.

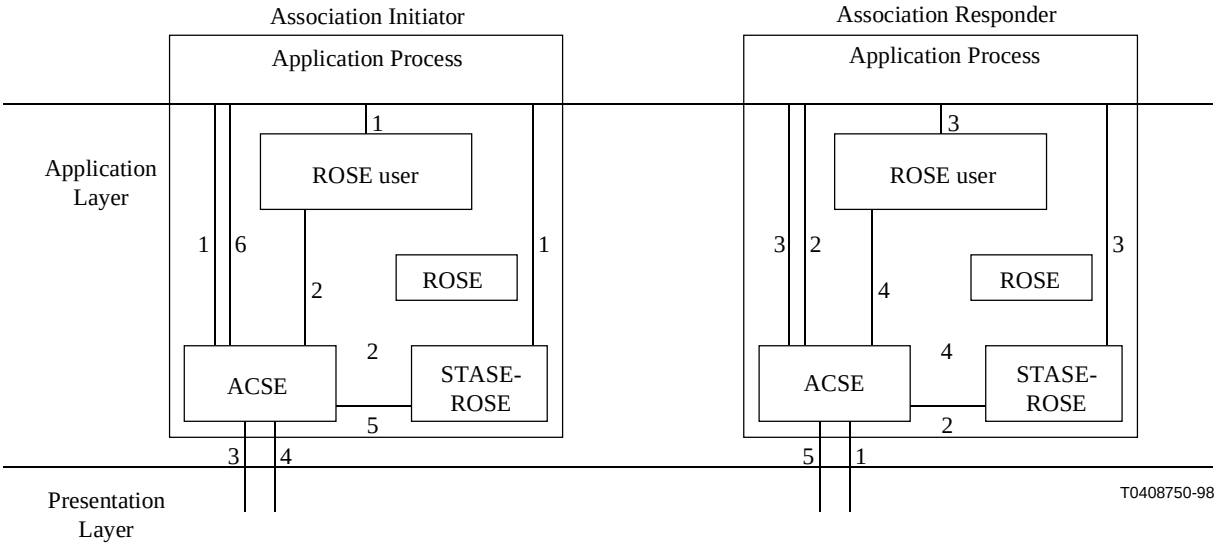

Figure 4/Q.813 – Interaction during association establishment

8.1.1 Association initiator

The following describes the interactions on the association initiating side in Figure 4:

1) The application process of the application entity involving STASE-ROSE issues an A-ASSOCIATE request to ACSE to establish an application association. If peer entity authentication is desired, the application process provides ACSE with the value of the authenticator to be carried in the Authentication-value field of the AARQ PDU (using the ACSE Authentication FU). During the same phase the application process may also inform

STASE-ROSE and the ROSE user ASE(s) (e.g. CMISE) of the requested association and provide STASE-ROSE with any proposed values for (some of) the encryption parameters.

2) STASE-ROSE provides ACSE with any proposed values for (some of) the encryption parameters. The mechanism by which STASE-ROSE informs ACSE is an implementation matter and is not addressed by this Recommendation. This information will be carried in the ACSE user information field using the EncryptionParametersSelection defined in 5.3. During the same phase the ROSE user ASE(s) may also provide ACSE with information relevant to that ASE(s). All such information is carried in the ACSE user information field. The ACSE user information field defined in ITU-T Recommendation X.227 | ISO/IEC 8650-1 consists of a SEQUENCE OF EXTERNAL. The information for STASE-ROSE, if any, shall be carried in the first EXTERNAL. The application context shall specify which EXTERNAL(s) shall carry information for each of its other ASE(s).

3) ACSE issues a P-CONNECT request to the presentation-provider to establish an application-association.

   The presentation service provider then transfers the P-CONNECT request and receives a response (not shown above).

4) The presentation provider issues a P-CONNECT confirmation primitive to ACSE, confirming the establishment of a presentation connection.

5) ACSE informs STASE-ROSE about the establishment of a new application-association and provides STASE-ROSE with the values (if any) of the encryption parameters. The mechanism by which ACSE informs STASE-ROSE is an implementation matter and is not addressed by this Recommendation.

6) ACSE issues an A-ASSOCIATE confirm primitive to the application process confirming establishment of association. ACSE provides the application process with the values (if any) of the encryption parameters. The mechanism by which ACSE informs the application process is an implementation matter and is not addressed by this Recommendation.

8.1.2 Association responder

The following describes the interactions on the association responding side in Figure 4.

The presentation service provider receives a connect request from the remote presentation provider:

1) The presentation-provider issues a P-CONNECT indication primitive to ACSE, informing a remote service-users interest in establishing an association.

2) ACSE issues an A-ASSOCIATE indication primitive to the application process. During the same phase ACSE informs STASE-ROSE about the request for the establishment of a new application-association and provides STASE-ROSE with the proposed values (if any) of the encryption parameters. ACSE further provides ASE-specific information (carried in the user information field), if any, to the ROSE users. The mechanism by which ACSE informs STASE-ROSE is an implementation matter and is not addressed by this Recommendation.

3) The application process issues an A-ASSOCIATE response primitive to ACSE accepting or rejecting the application-association. If the A-ASSOCIATE indication contains proposed values for (some of) the encryption parameters, then the application process may indicate to STASE-ROSE which values of those the encryption parameters should be the accepted. The mechanism by which the application process informs STASE-ROSE is an implementation matter and is not addressed by this Recommendation. During this phase the application process may also inform the ROSE user ASE(s) of the association.

4) STASE-ROSE provides ACSE with the accepted values, if any, for the encryption parameters in the AARQ. The mechanism by which STASE-ROSE informs ACSE is an implementation matter and is not addressed by this Recommendation. This information will be carried in the ACSE user information field using the EncryptionParametersSelection defined in 5.3. During the same phase the ROSE user ASE(s) may also provide ACSE with information relevant to that ASE(s). The ACSE user information field defined in ITU-T Rec. X.227 | ISO/IEC 8650-1 consists of a SEQUENCE OF EXTERNAL. The information for STASE-ROSE, if any, shall be carried in the first EXTERNAL. The application context shall specify which EXTERNAL(s) shall carry information for each of its other ASE(s).

5) ACSE issues a P-CONNECT response primitive to the presentation-provider accepting or rejecting establishment of association.

8.2 Association release

Figure 5 illustrates the sequence of interactions between the application process, various ASEs and the presentation provider during association release phase.

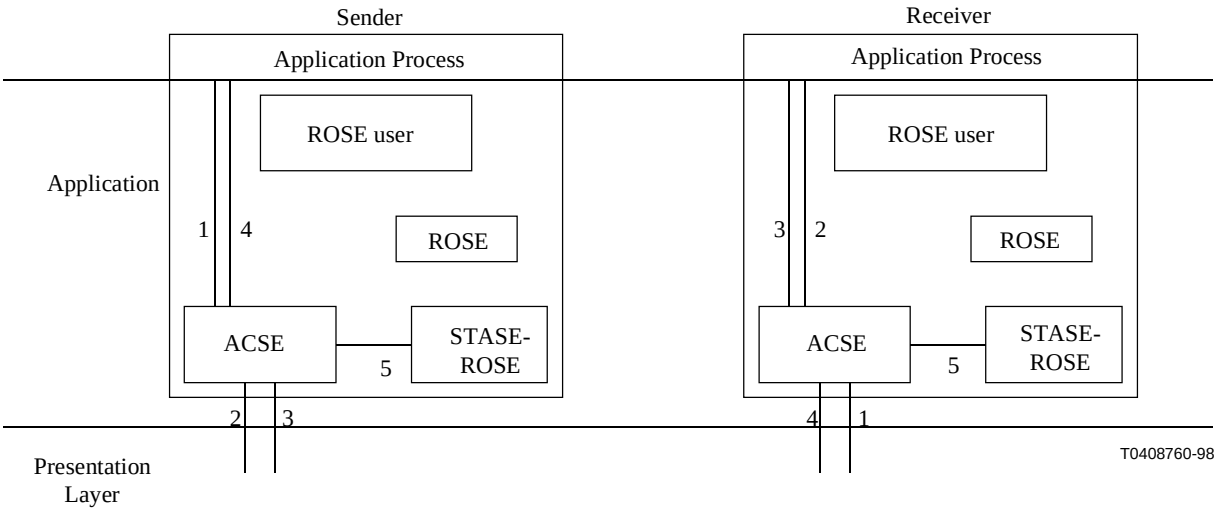

Figure 5/Q.813 – Interaction during association release

8.2.1 Sender

The following describes the interactions on the sending side in Figure 5:

1) The application process of the application context involving STASE-ROSE issues an A-RELEASE request to ACSE to release an application association.

2) ACSE issues a P-RELEASE request to the presentation-provider to release an application-association.

   The presentation service provider then transfers the P-RELEASE request to the peer application-entity and receives a response (not shown above).

3) The presentation provider issues a P-RELEASE confirmation primitive to ACSE, confirming the release of a presentation connection.

4) ACSE issues an A-RELEASE confirm primitive to the application process confirming release of the application-association.

5) ACSE informs STASE-ROSE and other ASEs (not shown in the figure) about the release of the application-association. The mechanism by which ACSE informs STASE-ROSE and other ASEs is an implementation matter and is not addressed by this Recommendation.

8.2.2 Receiver

The following describes the interactions on the receiving side in Figure 5.

The presentation service provider receives a release request from the remote presentation provider:

1) The presentation-provider issues a P-RELEASE indication primitive to ACSE, informing a remote service-users interest in releasing an association.
2) ACSE issues an A-RELEASE indication primitive to the application process.
3) The application process issues an A-RELEASE response primitive to ACSE accepting release of the application-association.
4) ACSE issues a P-RELEASE response primitive to the presentation-provider accepting release of association.
5) ACSE informs STASE-ROSE and other ASEs about the release of the application-association. The mechanism by which ACSE informs STASE-ROSE and other ASEs is an implementation matter and is not addressed by this Recommendation.

8.3 Association abort

Figure 6 illustrates the sequence of interactions between the application process, various ASEs and the presentation provider during association aborting.

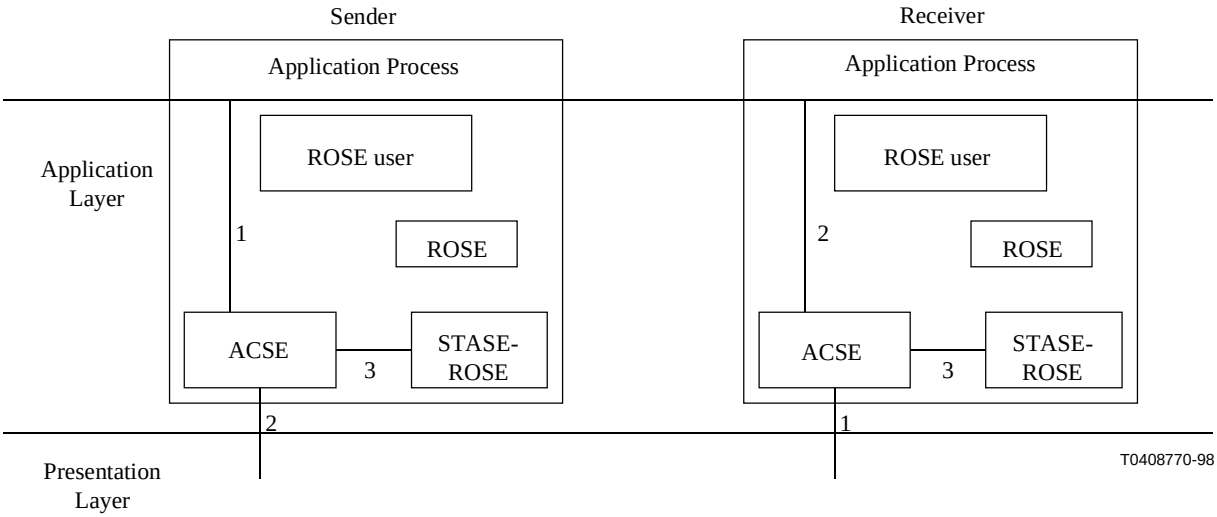

Figure 6/Q.813 – Interaction during association abort

8.3.1 Sender

The following describes the interactions on the sending side in Figure 6:

1) The application process using STASE-ROSE issues an A-ABORT request to ACSE to abort an application association.
2) ACSE issues a P-ABORT request to the presentation-provider to abort presentation connection.

3) ACSE informs STASE-ROSE and other ASEs about the aborting of the application-association. The mechanism by which ACSE informs STASE-ROSE and other ASEs is an implementation matter and is not addressed by this Recommendation.

8.3.2 Receiver

The following describes the interactions on the receiving side in Figure 6.

The presentation service provider detects the aborting of a presentation connection:

1) The presentation-provider issues a P-ABORT indication primitive to ACSE, informing that an application connection has been aborted.
2) ACSE issues an A-ABORT indication primitive to the application process.
3) ACSE informs STASE-ROSE and other ASEs about the aborting of the application-association. The mechanism by which ACSE informs STASE-ROSE and other ASEs is an implementation matter and is not addressed by this Recommendation.

8.4 Data transfer

Figure 7 shows the interaction between the application process, various ASEs and the presentation provider during data transfer phase.

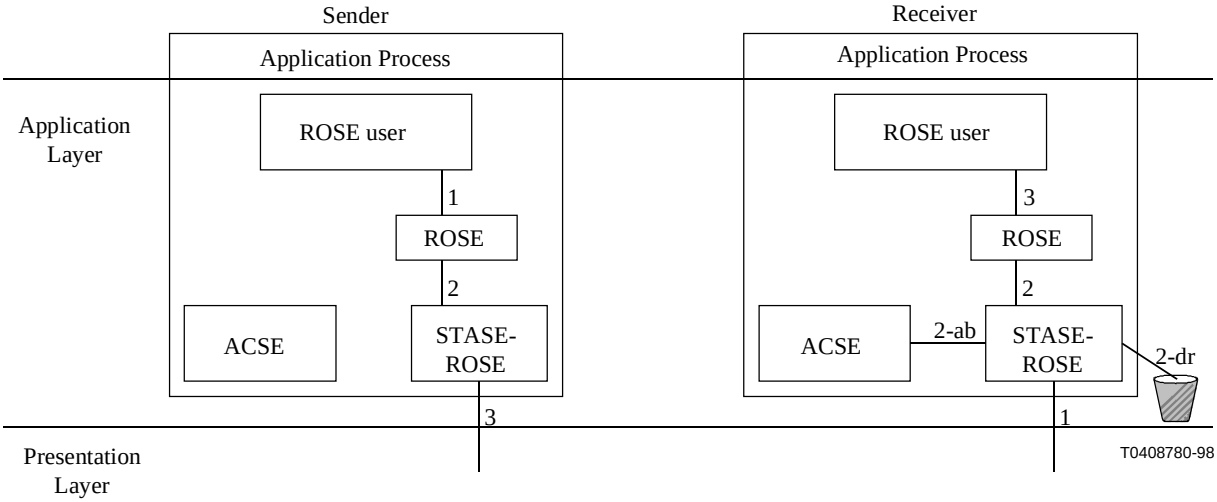

Figure 7/Q.813 – Interaction during data transfer

8.4.1 Sender

The following describes the interactions on the sending side in Figure 7:

1) The ROSE user, prompted (not shown) by the application process, issues a ROSE request or response primitive to ROSE, requesting transfer of data.
2) ROSE issues the SR-TRANSFER request primitive to STASE-ROSE, requesting secure transfer of data.
3) STASE-ROSE performs the required encoding and security transformation on the ROSE PDU (present in the request parameters) and issues a P-DATA request primitive to the presentation provider. STASE-ROSE, as currently defined, can be used by services corresponding to all Functional Units of CMIS except the extended services.

8.4.2 Receiver

The following describes the interactions on the sending side in Figure 7:

1) The presentation-provider issues a P-DATA indication primitive to STASE-ROSE, informing the arrival of data from the peer application entity on an application connection.

2) STASE-ROSE performs the reverse security transformations on the incoming data, checks for the validity of the PDU (e.g. validity of the seal or signature, currency of the timestamp, value of the sequence number), and if valid it issues a SR-TRANSFER indication primitive to ROSE.

   ROSE issues a ROSE indication or confirmation primitive to the ROSE user which then informs (not shown) the application process of the arrival of data from a peer application-entity.

   The receiving STASE-ROSE may find the incoming APDU unacceptable (e.g. decryption fails). In such case the action to be taken by STASE-ROSE is a local matter. However, this Recommendation recommends the following two alternatives:

   - The receiving STASE-ROSE implementation may drop the incoming APDU as shown by **2 dr** in Figure 7.
   - The receiving STASE-ROSE may issue an A-ABORT primitive to ACSE as shown in **2-ab** in Figure 7.

In either case it is recommended (not shown) that the event be reported to the user element, that the event be logged in a security audit trail and that a security alarm issued to the local security administrator.

9 STASE-ROSE protocol

The protocol specified in this clause supports the STASE-ROSE services described previously. As mentioned previously, STASE-ROSE uses the presentation layer P-DATA service for the transfer of secure ROSE PDUs.

The STASE-ROSE-protocol-machine (SRPM) communicates with ROSE by means of the SR-TRANSFER service primitives described previously. The SRPM is driven by service requests from ROSE, and by indication primitives from the presentation-service. The SRPM in turn, issues indication primitives to ROSE, and request primitives to the presentation-service. P-DATA request and P-DATA indication presentation service primitives are used.

The reception of a STASE-ROSE service primitive or reception of a presentation-service primitive, and the generation of the dependent actions are local matters outside the scope of this Recommendation.

During the exchange of APDUs, the existence of an application-association between the peer AEs is presumed. The association shall be established using the parameters specified in 7.4.2.

NOTE – Each application association may be identified in an end system by an internal, implementation dependent mechanism so that the STASE-ROSE service-user (ROSE) and SRPM can refer to it.

9.1 Abstract syntax definition of APDUs

In addition to the ASN.1 types defined in Recommendation X.229, the following types are defined for STASE-ROSE.

```
Secure-Remote-Operations-APDUs {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0) stase-data(2)}
DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS everything

IMPORTS

ROSEapdus
FROM Remote-Operations-ADPUs {joint-iso-ccitt remote-operations(4) apdus(1)}

AE-title
FROM ACSE-1 {joint-iso-ccitt association-control (2) abstract-syntax(1) apdus(0) version(1)}

DistinguishedName
FROM InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework (1)}

-- the referenced module and corresponding syntax are found in Annex D/Rec. X.711 – 1998.

Certificate, CertificationPath
FROM AuthenticationFramework {joint-iso-ccitt ds(5) modules(1) authenticationFramework(7)};

SR-APDU ::= CHOICE
        { clear                     [0] ROSEapdus,
          simpleConfidential        [1] OCTET STRING,
          confidential              [2] Enciphered ,
          simplePublicEnciphered    [3] SimplePublicEnciphered,
          publicEnciphered          [4] PublicEnciphered ,
          hashed                    [5] HashedROSEpdu ,
          sealed                    [6] SealedROSEpdu ,
          signed                    [7] SignedROSEpdu ,
          confidentialSigned        [8] ConfidentialSigned,
         confidentialMAC            [9] ConfidentialMAC,
          confidentialSealed        [10] ConfidentialSealed,
          gssToken                  [11] GssToken,
          ...
         }

Enciphered ::= SEQUENCE
        {encrypted                  OCTET STRING,
         encryptionParameters       EncryptionParameters OPTIONAL
        }

-- encrypted represents the DER encoded and encrypted ROSE PDU.
-- encryptionParameters represents the parameters used for encryption.

SimplePublicEnciphered ::= CHOICE
                        { integers      SEQUENCE OF INTEGER,
                          string        OCTET STRING
                        }

-- SimplePublicEnciphered represents the DER encoded and public key encrypted ROSE PDU.
-- A large PDU may be broken into smaller blocks, each of which may be encrypted
-- as an INTEGER. The size of such blocks depends on the public key encryption algorithm
-- used and on the size of the public key; specification of such block sizes is outside the
-- scope of this Recommendation.
-- In some cases the result of public key encryption may be represented as an OCTET STRING.

PublicEnciphered ::= SEQUENCE
                        {publicEncrypted    SimplePublicEnciphered,
                         encryptionParameters     EncryptionParameters OPTIONAL
                        }
```

```
-- publicEncrypted represents the DER encoded and public key encrypted ROSE PDU.
-- encryptionParameters represents the parameters used for encryption.

Hash ::= SEQUENCE{
            hashValue               OCTET STRING (SIZE(8..64)),
            encryptionParameters    EncryptionParameters OPTIONAL
            }

-- hashValue represents the message digest resulting from hashing the DER encoded
-- ROSE PDU.
-- encryptionParameters represents the parameters used for the hashing algorithm.

HashedROSEpdu ::= SEQUENCE
             {data      OCTET STRING,
              hash      CHOICE { hash           Hash,
                                 simpleHash     OCTET STRING (SIZE (8..64))
                                 }
      }

-- data represents the DER encoded ROSE PDU
-- hash represents the hash value either as a simple OCTET STRING or the Hash
-- structure defined above.

Seal ::= SEQUENCE
            {sealValue                  OCTET STRING (SIZE(8..128)),
             encryptionParameters       EncryptionParameters OPTIONAL
            }

-- sealValue represents the seal value for the DER encoded ROSE PDU.
-- encryptionParameters represents the parameters used by the seal generation algorithm.

SealedROSEpdu ::= SEQUENCE
            {data       OCTET STRING,
             seal       CHOICE {seal            Seal,
                                simpleSeal      OCTET STRING (SIZE(8..64))
                                }
      }

-- data represents the DER encoded ROSE PDU
-- seal represents the seal value either as a simple OCTET STRING or the Seal structure
-- defined above.

Signature ::= SEQUENCE
                {signatureValue             SEQUENCE (SIZE(1..4)) OF INTEGER,
                 encryptionParameters       EncryptionParameters OPTIONAL
                }

-- signatureValue represents the signature for the DER encoded ROSE PDU.
-- encryptionParameters represents the parameters for the signature algorithm.

SignedROSEpdu ::= SEQUENCE
      {data         OCTET STRING,
       signature    CHOICE {signature           [1] Signature,
                            simpleSignature     [2] SEQUENCE (SIZE(1..4)) OF INTEGER
                            }
      }

-- data contains the DER encoding of the ROSE PDU.
-- signature represents the signature of the DER encoded ROSE PDU, either as a simple
-- INTEGER or the Signature structure defined above.
```

```
ConfidentialSigned ::= SEQUENCE
        { encrypted   OCTET STRING,
          signature   CHOICE {signature              [1] Signature,
                              simpleSignature        [2] SEQUENCE (SIZE(1..4)) OF INTEGER
                              }
        }

-- encrypted represents the encryption of the DER encoded ROSE PDU.
-- signature represents the signature of the DER encoded ROSE PDU in either a simple form
-- or as Signature type defined above.

ConfidentialMAC ::= SEQUENCE
        { encrypted   OCTET STRING,
          mac         CHOICE {mac              [1] Hash,
                              simpleMAC        [2] OCTET STRING (SIZE (8..64))
                              }
        }

-- encrypted represents the encryption of the DER encoded ROSE PDU.
-- mac represents the MAC of the DER encoded ROSE PDU in either a simple form
-- or as Hash type defined above.

ConfidentialSealed ::= SEQUENCE
        { encrypted   OCTET STRING,
          seal        CHOICE {sealed     [1] Seal,
                              simpleSealed   [2] OCTET STRING (SIZE (8..64))
                              }
        }

-- encrypted represents the encryption of the DER encoded ROSE PDU.
-- seal represents the seal of the DER encoded ROSE PDU in either a simple form
-- or as Seal type defined above.

EncryptionParameters ::= SET
        {symmetricKeyId         [0] KeyId              OPTIONAL,
        publicKeyId             [1] KeyId              OPTIONAL,
        sealKeyId               [2] KeyId              OPTIONAL,
        signatureKeyId          [3] KeyId              OPTIONAL,
        passwordId              [4] KeyId              OPTIONAL,
        initializationVector    [5] OCTET STRING (SIZE(8))   OPTIONAL,
        feedBackBits            [6] INTEGER (1..63)          OPTIONAL,
        -- for k-bit output feedback mode or k-bit cipher feedback mode of DES
        symmetricAlgorithm      [7] OBJECT IDENTIFIER        OPTIONAL,
        publicKeyAlgorithm      [8] OBJECT IDENTIFIER        OPTIONAL,
        signatureAlgorithm      [9] OBJECT IDENTIFIER        OPTIONAL,
        sealAlgorithm           [10] OBJECT IDENTIFIER       OPTIONAL,
        hashAlgorithm           [11] OBJECT IDENTIFIER       OPTIONAL,
        keyDigest               [12] OCTET STRING (SIZE(8..64)) OPTIONAL,
        -- for verification of public keys
        blockSize               [13] INTEGER                 OPTIONAL,
        -- for square mod-n hashing
        keySize                 [14] INTEGER                 OPTIONAL,
        -- for RSA
        publicKey               [15] SEQUENCE
                                     {modulus    INTEGER,
                                      exponent   INTEGER
                                     }                       OPTIONAL,
        sequenceNumber          [16] INTEGER                 OPTIONAL,
        timeStamp               [17] GeneralizedTime         OPTIONAL,
```

```
        encryptedKey            [18] OCTET STRING (SIZE(64..128)) OPTIONAL,
        -- symmetric session key, encrypted with Key-Encryption-Key
        encryptedSymmetricKey   [19] INTEGER      OPTIONAL,
        -- symmetric session key, encrypted with the receiver's public key
        keyEncryptionKey    [20] SEQUENCE (SIZE (1..3)) OF KeyId          OPTIONAL,
        -- one to three symmetric keys used for encrypting a session key
        publicKeyCertificate    [21]    PublicKeyCertificate              OPTIONAL,
-- X.509 certificate or certification path of the sender's public key with no usage restrictions
        encryptionCertificate   [22] EncryptionCertificate          OPTIONAL,
-- X.509 certificate or certification path of the sender's public key used for encryption only
        signatureCertificate    [23] SignatureCertificate           OPTIONAL,
-- X.509 certificate or certification path of the sender's public key used for digital signatures only --
encryptedAuthenticatedSymmetricKey [24] EncryptedAuthenticatedSymmetricKey OPTIONAL,
-- symmetric session key, encrypted with the receiver's public key and signed with sender's key--
        macAlgorithm            [25] OBJECT IDENTIFIER                    OPTIONAL,
                ...
                }

-- EncryptionParameters is an extensible type that is used as a catch-all for any
-- parameters that may be used by any of the STs. In most applications only a small
-- number, if any, of the components of EncryptionParameters will be used.

KeyId ::= CHOICE        {
                         name           GraphicString,
                         number         INTEGER
                        }

PublicKeyCertificate ::= CHOICE {certificate            [0] Certificate,
                                 certificationPath      [1] CertificationPath
                                 }

EncryptionCertificate ::= CHOICE {certificate           [0] Certificate,
                                  certificationPath     [1] CertificationPath
                                  }

SignatureCertificate ::= CHOICE {certificate            [0] Certificate,
                                 certificationPath      [1] CertificationPath
                                }

EncryptedAuthenticatedSymmetricKey ::= SEQUENCE {
                                 encryptedSymmetricKey  INTEGER,
        -- symmetric session key, encrypted with the receiver's public key
                                 time           GeneralizedTime,
                                 sender         SenderId,
                                 receiver       ReceiverId,
                                 signature      Signature
-- the signature is computed over ASCII representation of the preceding four fields with the sender's private key
                                 }
SenderId ::= CHOICE {
                        identifier      [1] DistinguishedName,
                        name            [2] GraphicString,
                        application     [3] AE-title
                        }

ReceiverId ::= SenderId

GssToken::=CHOICE {
        micToken    [1]     MicToken ,
        wrapToken   [2]     OCTET STRING
    }
```

```
MicToken::=SEQUENCE {
            rosePDU     [1]     OCTET STRING ,
            token       [2]     OCTET STRING
      }

END
```

9.2 Abstract syntax name

This Recommendation assigns the following object identifier:

{itu-t recommendation q(17) q813(813) stase(1) abstractSyntax(1) stase-data(2)}

as an abstract syntax name for the set of presentation data values each of which is a value of the ASN.1 type

Secure-Remote-Operations-APDUs.SR-APDU

where the ROSE PDUs argument components are filled by the user of ROSE.

The corresponding Object descriptor value shall be "STASE-ROSE-Data"".

9.3 Algorithms identifiers

Unless otherwise agreed by the communicating entities (by means outside the scope of this Recommendation), the ST algorithms will be restricted to those identified in ISO/IEC 9979, and they will be identified by the OBJECT IDENTIFIERs provided in that International standard.

9.4 Application contexts names

9.4.1 Secure TMN context

The application context name where application entity is composed of SMASE, CMISE, ROSE, STASE-ROSE and ACSE shall have the following object identifier value assignment:

{itu-t recommendation q(17) q813(813) stase(1) stase-application-context (2) secureTMNContext(0)}

and the following object-descriptor value "Secure-TMN-Interactive-Application-Context".

The ACSE user information field defined in ITU-T Rec. X.227 | ISO/IEC 8650-1 consists of a SEQUENCE OF EXTERNAL. This Recommendation specifies that for this application context the order of the EXTERNALs in the ACSE user information field is:

– data supplied for STASE-ROSE, if any;

– data supplied for CMISE, as defined in ITU-T Rec. X.711 | ISO/IEC 9596-1, if any;

– data supplied for SMASE as defined in ITU-T Rec. X.701 | ISO/IEC 10040, if any.

9.4.2 Secure Directory Application Context

The application context name where application entity is composed of X.500, ROSE, STASE-ROSE and ACSE shall have the following object identifier value assignment:

{itu-t recommendation q(17) q813(813) stase(1) stase-application-context (2) secureDirectoryContext(1)}

and the following object-descriptor value "Secure–Directory-Application-Context".

9.5 STASE-ROSE procedures

The STASE-ROSE protocol consists of a single element of procedure:

Transfer.

In the following subclauses, a summary of this element of procedure is presented. This consists of a summary of the relevant APDUs, and a high level overview of the STASE-ROSE service primitives, the APDUs involved, and the transfer service that is used.

9.5.1 Transfer

9.5.1.1 Purpose

The transfer procedure is used by a STASE-ROSE service-user (ROSE) to transfer a ROSE APDU in a secure manner. The STASE-ROSE shall apply the necessary security transformations to the ROSE PDU and transfer to the peer STASE-ROSE.

9.5.1.2 APDUs used

The transfer procedure uses the STASE-ROSE APDU.

The fields of the STASE-ROSE APDU (SR-APDU) are listed in Table 9-1. Only one of the fields shall be used in a single SR-APDU. Subclause 9.5.1.4 describes the usage of fields in the SR-APDU. Refer to 9.1 for the ASN.1 definition of the SR-APDU.

Table 9-1/Q.813 – STASE-ROSE APDU fields

Field Name	Source	Sink
clear	req	ind
simpleConfidential	req	ind
confidential	req	ind
simplePublicEnciphered	req	ind
publicEnciphered	req	ind
hashed	req	ind
sealed	req	ind
signed	req	ind
confidentialSigned	req	ind
confidentialMAC	req	ind
confidentialSealed	req	ind

9.5.1.3 Transfer procedure

This procedure is driven by the following events:

a) an SR-TRANSFER request primitive from the requester;

b) a P-DATA indication primitive from the presentation service.

9.5.1.3.1 Overview

In an open system environment each system can have its own internal representation of information elements such as characters, integers and reals. In order to support communications between heterogeneous open systems such information items can be specified using the ASN.1 notation, and they can be exchanged in an agreed upon transfer syntax such as BER or DER. If an ST (such as

encryption) is performed on an information item in the application layer, using the system's internal representation (e.g. ASCII representation of characters), then the result of this transformation would be meaningless to another open system that uses a different internal representation in the application layer (e.g. EBCDIC representation of characters). For this reason STs should be carried on a transfer syntax of information items.

This Recommendation specifies that STs are performed on DER encoded ROSE PDUs.

9.5.1.3.2 SR-TRANSFER request primitive

The requesting SRPM forms a SR-APDU from the parameter values of the SR-TRANSFER request primitive.

The SR-APDU fields are constructed as follows:

1) If the value of Encryption-Type parameter (see 7.4.4) is equal to **clear**, the ROSE-PDU parameter shall be directly assigned to the **clear** field in the **SR-APDU**.

2) The following procedure is performed for any other value of Encryption-Type parameter:
   - STASE-ROSE shall first encode the ROSE-PDU parameter using DER.
   - If privacy protection of ROSE PDU is required (i.e. the value of Encryption-Type parameter is one of simpleConfidential, confidential, simplePublicEnciphered, publicEnciphered), STASE-ROSE shall encrypt the DER encoded octet stream and assign it to one of the fields in **SR-APDU** as follows:
     - If the value of the Encryption-Type parameter is equal to **simpleConfidential**, the DER encoded octet stream shall be encrypted using the default values as described in 5.2. The encrypted data is assigned to the **simpleConfidential** field.
     - If the value of the Encryption-Type parameter is **confidential**, the DER encoded octet stream shall be encrypted using the information provided in the Encryption-Parameters parameter. The encrypted data and the Encryption-Parameters shall be assigned to the **confidential.encrypted** and **confidential.encryptionParameters** fields in the SR-APDU respectively.
     - If the value of the Encryption-Type parameter is **simplePublicEnciphered,** the DER encoded octet stream shall be encrypted using the default public key information as described in 5.2. The encrypted data is assigned to the **simplePublicEnciphered** field.
     - If the value of the Encryption-Type parameter is **publicEnciphered,** the DER encoded octet stream shall be encrypted using the Encryption-Parameters parameter. The encrypted data and the Encryption-Parameters are assigned to the **enciphered.publicEncrypted** and **enciphered.encryptionParameters** fields respectively.
   - If integrity check is required (i.e. the value of Encryption-Type parameters is one of simpleHashed, hashed, simpleSealed, sealed), STASE-ROSE shall compute the digital seal or hash of the DER encoded octet stream and assign it to the different fields in the **SR-APDU** structure as follows:
     - If the value of the Encryption-Type parameter is equal to **simpleHashed,** the DER encoded octet stream is hashed using the default values as described in 5.2. The DER encoded octet stream and its hash value are assigned to the **hashed.data** and **hashed.hash.simpleHash** fields in the **SR-APDU** structure respectively.

– If the value of the Encryption-Type parameter is equal to **hashed,** the DER encoded octet stream is hashed using the information provided in the Encryption-Parameters parameter. The DER encoded byte stream, its hash value and the Encryption-Parameters are assigned to the **hashed.data, hashed.hash.hash.hashValue** and **hashed.hash.hash.encryptionParameters** fields respectively, in the **SR-APDU** structure.

– If the value of the Encryption-Type parameter is equal to **simpleSealed,** the DER encoded octet stream is sealed using the default values as described in 5.2. The DER encoded octet stream and its seal value are assigned to the **sealed.data** and **sealed.seal.simpleSeal** fields respectively in the **SR-APDU** structure.

– If the value of the Encryption-Type parameter is equal to **sealed,** the DER encoded octet stream is sealed using the information provided in the Encryption-Parameters parameter. The DER encoded byte stream, its seal value and the Encryption-Parameters are assigned to the **sealed.data, sealed.seal.seal.sealValue** and **sealed.seal.seal.encryptionParameters** fields respectively, in the **SR-APDU** structure.

- If non-repudiation is required (i.e. the value of the Encryption-Type parameter is simpleSignature or signature), STASE-ROSE shall compute the digital signature of DER encoded octet stream and assign to the SR-APDU structure as follows:

  – If the value of the Encryption-Type parameter is equal to **simpleSigned,** the digital signature of the DER encoded octet stream is computed using the default values as described in 5.2. The DER encoded octet stream and its signature are assigned to the **signed.data** and **signed.signature.simpleSignature** fields respectively in the **SR-APDU** structure.

  – If the value of the Encryption-Type parameter is equal to **signed,** the digital signature of the DER encoded octet stream is computed using the information provided in the Encryption-Parameters parameter. The DER encoded byte stream, its signature and the Encryption-Parameters are assigned to the **signed.data signed.signature.signature.signatureValue,** and **signed.signature.signature.encryptionParameters** fields in the **SR-APDU** structure respectively.

- If both privacy and non-repudiation are desired (i.e. the value of Encryption-Type parameter is simpleConfidentialSigned or confidentialSigned), STASE-ROSE shall compute the encryption of the DER encoded PDU and the signature of the DER encoded PDU. Unless otherwise agreed upon by the communicating entities by means outside the scope of this Recommendation, the signature shall be computed on the clear, i.e. not encrypted, DER encoded ROSEpdu. It shall assign the results of encryption and signing to the **SR-APDU** structure as shown below:

  – If the value of Encryption-Type parameter is **simpleConfidentialSigned,** the digital signature and encrypted value of the DER encoded octet stream are computed using the default values as described in 5.2. The encrypted data and digital signature shall be assigned to the **confidentialSigned.encrypted** and **confidentialSigned.signature.simpleSignature** fields in the **SR-APDU** structure respectively.

  – If the value of Encryption-Type parameter is **confidentialSigned,** the digital signature and encrypted value of the DER encoded octet stream are computed using the information provided in the Encryption-Parameters parameter. The encrypted data, the digital signature and the Encryption-Parameters shall be assigned to the

confidentialSigned.encrypted,
confidentialSigned.signature.signature.signatureValue and
confidentialSigned.signature.signature.encryptionParameters fields in the **SR-APDU** structure respectively.

- If both privacy and integrity are desired (i.e. the value of Encryption-Type parameter is simpleConfidentialMAC or confidentialMAC), STASE-ROSE shall compute the encryption of the DER encoded PDU and the MAC of the DER encoded PDU. Unless otherwise agreed upon by the communicating entities by means outside the scope of this Recommendation, the MAC shall be computed on the clear, i.e. not encrypted, DER encoded ROSEpdu. It shall assign the results of encryption and signing to the **SR-APDU** structure as shown below:
  - If the value of Encryption-Type parameter is **simpleConfidentialMAC,** the MAC and encrypted value of the DER encoded octet stream are computed using the default values as described in 5.2. The encrypted data and MAC shall be assigned to the **confidentialMAC.encrypted** and **confidentialMAC.mac.simpleMAC** fields in the **SR-APDU** structure respectively.
  - If the value of Encryption-Type parameter is **confidentialMAC,** the MAC and encrypted value of the DER encoded octet stream are computed using the information provided in the Encryption-Parameters parameter. The encrypted data, the MAC and the Encryption-Parameters shall be assigned to the **confidentialMAC.encrypted, confidentialMAC.mac.mac.hashValue** and **confidentialMAC.mac.mac.encryptionParameters** fields in the **SR-APDU** structure respectively.
- If both privacy and digital seal-based integrity are desired (i.e. the value of Encryption-Type parameter is simpleConfidentialSealed or confidentialSealed), STASE-ROSE shall compute the encryption of the DER encoded PDU and the seal of the DER encoded PDU. Unless otherwise agreed upon by the communicating entities by means outside the scope of this Recommendation, the seal shall be computed on the clear, i.e. not encrypted, DER encoded ROSEpdu. It shall assign the results of encryption and signing to the **SR-APDU** structure as shown below:
  - If the value of Encryption-Type parameter is **simpleConfidentialSealed,** the seal and encrypted value of the DER encoded octet stream are computed using the default values as described in 5.2. The encrypted data and seal shall be assigned to the **confidentialSealed.encrypted** and **confidentialSealed.seal.simpleSealed** fields in the **SR-APDU** structure respectively.
  - If the value of Encryption-Type parameter is **confidentialSealed,** the seal and encrypted value of the DER encoded octet stream are computed using the information provided in the Encryption-Parameters parameter. The encrypted data, the seal and the Encryption-Parameters shall be assigned to the **confidentialSealed.encrypted, confidentialSealed.seal.seal.sealValue** and **confidentialSealed.seal.seal.encryptionParameters** fields in the **SR-APDU** structure respectively.

The SR-APDU thus formed is transferred to the peer STASE-ROSE as the user-data parameter of the P-DATA transfer request primitive of the presentation service.

The requesting SRPM waits either for a P-DATA indication primitive from the presentation layer or a SR-TRANSFER request primitive from the requester.

9.5.1.3.3 P-DATA indication primitive

The accepting SRPM receives an SR-APDU from its peer as user-data on a P-DATA transfer indication primitive. The following procedure will be performed by the STASE-ROSE to retrieve the SR-TRANSFER indication primitives:

1) If the **clear** field in **SR-APDU** has been selected in the incoming APDU, the value of Encryption-Type parameter shall be set to **clear**, the ROSE-PDU parameter shall be set to the **clear** field in the **SR-APDU**.

2) The following procedure is performed if any other field is selected in the incoming APDU:

   - If the **simpleConfidential** field has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **simpleConfidential** field using the default values as described in 5.2. The value of the Encryption-Type parameter shall be **simpleConfidential.** The procedure to be followed when decryption fails is described later in this subclause.
   - If the **confidential** field has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **confidential.encrypted** field using the **confidential.encryptionParameters** field. The value of the Encryption-Type parameter shall be **confidential.** The value of the Encryption-Parameters parameters shall be the value of **confidential.encryptionParameters** field. The procedure to be followed when decryption fails is described later in this subclause.
   - If the **simplePublicEnciphered** field has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **simplePublicEnciphered** field using the default values as described in 5.2. The value of the Encryption-Type parameter shall be **simplePublicEnciphered.** The procedure to be followed when decryption fails is described later in this subclause.
   - If the **publicEnciphered** field has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **publicEnciphered.publicEncrypted** field using the **publicEnciphered.encryptionParameters** field. The value of the Encryption-Type parameter shall be **confidential.** The value of the Encryption-Parameters parameters shall be the value of **publicEnciphered.encryptionParameters** field. The procedure to be followed when decryption fails is described later in this subclause.
   - If **hashed** field has been selected in the incoming APDU and:
     – If the **hashed.hash.simpleHash** field has been selected, the value of the **hashed.data** field shall be used as the DER encoding of the ROSE PDU. The hash value of DER encoded octet stream shall be computed using the default values as described in 5.2 and compared with the **hashed.hash.simpleHash** field. The value of the Encryption-Type parameter shall be **simpleHashed**. The procedure to be performed when hash value comparison fails is described later in this subclause.
     – If the **hashed.hash.hash** field has been selected, the value of the **hashed.data** field shall be used as the DER encoding of the ROSE PDU. The hash value of DER encoded octet stream shall be computed using the **hashed.hash.hash.encryptionParameters** field and compared with the **hashed.hash.hash.hashValue** field. The value of the Encryption-Type parameter shall be **hashed**. The value of Encryption-Parameters parameters shall be assigned the value of **hashed.hash.hash.encryptionParameters** field. The procedure to be performed when hash value comparison fails is described later in this subclause.

- If **sealed** field has been selected in the incoming APDU and:
  - If the **sealed.seal.simpleSeal** field has been selected, the value of the **sealed.data** field shall be used as the DER encoding of the ROSE PDU. The digital seal of DER encoded byte stream shall be computed using the default values as described in 5.2 and compared with the **sealed.seal.simpleSeal** field. The value of the Encryption-Type parameter shall be **simpleSealed**. The procedure to be performed when digital seal comparison fails is described later in this subclause.
  - If the **sealed.seal.seal** field has been selected, the value of the **sealed.data** field shall be used as the DER encoding of the ROSE PDU. The digital seal of DER encoded octet stream shall be computed using the **sealed.seal.seal.encryptionParameters** field and compared with the **sealed.seal.seal.sealValue** field. The value of the Encryption-Type parameter shall be **sealed**. The value of Encryption-Parameters parameters shall be assigned the value of **sealed.seal.seal.encryptionParameters** Field. The procedure to be performed when digital seal comparison fails is described later in this subclause.
- If **signed** field has been selected in the incoming APDU and:
  - If the **signed.signature.simpleSignature** field has been selected, the value of the **signed.data** field shall be used as the DER encoding of the ROSE PDU. The digital signature of DER encoded octet stream shall be computed using the default values as described in 5.2 and compared with the **signed.signature.simpleSignature** field. The value of the Encryption-Type parameter shall be **simpleSigned**. The procedure to be performed when digital signature comparison fails is described later in this subclause.
  - If the **signed.signature.signature** field has been selected, the value of the **signed.data** field shall be used as the DER encoding of the ROSE PDU. The digital signature of DER encoded octet stream shall be computed using the **signed.signature.signature.encryptionParameters** field and compared with the **signed.signature.signature.signatureValue** field. The value of the Encryption-Type parameter shall be **signed**. The value of Encryption-Parameters parameters shall be assigned the value of **signed.signature.signature.encryptionParameters** Field. The procedure to be performed when digital signature comparison fails is described later in this subclause.
- If the **confidentialSigned** field has been selected in the incoming APDU and:
  - If the **confidentialSigned.signature.simpleSignature** has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **confidentialSigned.encrypted** field using the default values as described in 5.2. The digital signature of the DER encoded octet stream shall be computed using the default values and compared with the **confidentialSigned.signature.simpleSignature** field. The value of the Encryption-Type parameter shall be set to simpleConfidentialSigned. If either decryption or digital signature comparison fails, the procedures discussed at the end of this subclause shall be followed.
  - If the **confidentialSigned.signature.signature** has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **confidentialSigned.encrypted** field using the **confidentialSigned.signature.signature.encryptionParameters** field. The digital signature of the DER encoded octet stream shall be computed using the **confidentialSigned.signature.signature.encryptionParameters** field and

compared with the **confidentialSigned.signature.signature.signatureValue** field. The value Encryption-Type parameter shall be set to **confidentialSigned** and the value of Encryption-Parameters parameter shall be set to **confidentialSigned.signature.signature.encryptionParameters**. If either decryption or digital signature comparison fails, the procedures discussed at the end of this subclause shall be followed.

- If the **confidentialMAC** field has been selected in the incoming APDU and:
  - If the **confidentialMAC.mac.simpleMAC** has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **confidentialMAC.encrypted** field using the default values as described in 5.2. The MAC of the DER encoded octet stream shall be computed using the default values and compared with the **confidentialMAC.mac.simpleMAC** field. The value of the Encryption-Type parameter shall be set to simpleConfidentialMAC. If either decryption or MAC comparison fails, the procedures discussed at the end of this subclause shall be followed.
  - If the **confidentialMAC.mac.mac** has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **confidentialMAC.encrypted** field using the **confidentialMAC.mac.mac.encryptionParameters** field. MAC of the DER encoded octet stream shall be computed using the **confidentialMAC.mac.mac.encryptionParameters** field and compared with the **confidentialMAC.mac.mac.hashValue** field. The value Encryption-Type parameter shall be set to **confidentialMAC** and the value of Encryption-Parameters parameter shall be set to **confidentialMAC.mac.mac.encryptionParameters**. If either decryption or MAC comparison fails, the procedures discussed at the end of this subclause shall be followed.
- If the **confidentialSealed** field has been selected in the incoming APDU and:
  - If the **confidentialSealed.seal.simpleSealed** has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **confidentialSealed.encrypted** field using the default values as described in 5.2. The seal of the DER encoded octet stream shall be computed using the default values and compared with the **confidentialSealed.seal.simpleSealed** field. The value of the Encryption-Type parameter shall be set to **simpleConfidentialSealed**. If either decryption or seal comparison fails, the procedures discussed at the end of this subclause shall be followed.
  - If the **confidentialSealed.seal.seal** has been selected, the DER encoded octet stream corresponding to the ROSE PDU shall be retrieved by decrypting the **confidentialSealed.encrypted** field using the **confidentialSealed.seal.seal.encryptionParameters** field. The seal of the DER encoded octet stream shall be computed using the **confidentialSealed.seal.seal.encryptionParameters** field and compared with the **confidentialSealed.seal.seal.sealValue** field. The value Encryption-Type parameter shall be set to **confidentialSealed** and the value of Encryption-Parameters parameter shall be set to **confidentialSealed.seal.seal.encryptionParameters**. If either decryption or seal comparison fails, the procedures discussed at the end of this subclause shall be followed.
- STASE-ROSE shall decode the DER encoded octet stream retrieved from the incoming APDU and assign it to the ROSE-PDU parameter.

If the SRPM is able to perform the above procedures successfully, it shall issue an SR-TRANSFER indication primitive to the acceptor with the parameters retrieved.

The procedures to be performed when any of the values in the SR-APDU are unacceptable to the SRPM is a local matter. However, this Recommendation recommends one of the two actions to be taken by the SRPM implementation when an unacceptable SR-APDU is received.

1) The SRPM shall drop the SR-APDU. The associated local application-entity may be informed, in an implementation dependent manner, that an unacceptable APDU has been received and dropped.

2) The SRPM shall invoke the A-ABORT service provided by the Association Control Service Element to abort the application-association. The associated local application-entity may be informed, in an implementation dependent manner, that an unacceptable APDU has been received and the application-association has been aborted.

In either case it is recommended that the event be logged in a security audit trail and a security alarm issued to the local security administrator.

The accepting SRPM waits either for a p-data indication primitive from presentation layer or SR TRANSFER request primitive from the service-user.

9.5.1.4 Use of the SR APDU fields

The SR APDU fields are used as follows:

1) **clear**: This field is used if no security transformations are requested by the SR-user.

2) **simpleConfidential**: This field is used if privacy protection is requested by SR-user and default encryption parameters are used.

3) **confidential**: This field is used if privacy protection is requested by the SR-user and the encryption parameters are provided by SR-user.

4) **simplePublicEnciphered**: This field is used when PKCS is requested by the SR-user and the encryption parameters are not provided by the SR-user.

5) **publicEnciphered**: This field is used when PKCS is used and the encryption parameters are provided by the SR-user.

6) **hashed**: This field is used when hashing based protection is requested by the SR-user.

7) **sealed**: This field is used when digital seal based protection is requested by the SR-user.

8) **signed**: This field is used when non-repudiation is requested by the SR-user.

9) **confidentialSigned**: This field is used when both non-repudiation and complete privacy protection are requested by the SR-user.

10) **confidentialMAC**: This field is used when both hashing-based integrity and complete privacy protection are requested by the SR-user.

11) **confidentialSealed**: This field is used when both seal-based integrity and complete privacy protection are requested by the SR-user.

9.6 Mapping of STASE-ROSE services to presentation service

This subclause defines how the presentation-service primitives described in Recommendation X.216 are used by SRPM. The mapping of STASE-ROSE service primitives and APDUs to the presentation-service primitives are defined in Table 9-2.

The P-DATA service is a non-confirmed service. The use of the P-DATA request and P-DATA indication primitive parameters is as follows:

- **User Data**: The APDU to be transferred. Its maximum size is not restricted by this mapping.

Table 9-2/Q.813 – Presentation service mapping overview

STASE-ROSE Service	APDU	Presentation service
SR-TRANSFER request/indication	SR-APDU	P-DATA request/indication

10 Mapping of ROSE services to STASE-ROSE services

This clause defines how the STASE-ROSE service primitives described in this Recommendation are used by ROSE services defined in Recommendation X.219. The following table defines the mapping:

Table 10-1/Q.813 – Mapping of ROSE services to STASE-ROSE services

ROSE Service	APDU	STASE-ROSE service
RO-INVOKE request/indication	ROIV	SR-TRANSFER request/indication
RO-RESULT request/indication	RORS	SR-TRANSFER request/indication
RO-ERROR request/indication	ROER	SR-TRANSFER request/indication
RO-REJECT-U request/indication	RORJ	SR-TRANSFER request/indication
RO-REJECT-P request/indication	RORJ	SR-TRANSFER request/indication

The SR-TRANSFER service is a non-confirmed service.

11 Conformance

An implementation claiming conformance to this Recommendation shall comply with the following requirements:

- **Statement requirements**: An implementor shall state the following:
  a) the application context for which conformance is claimed;
  b) if the negotiation of security parameters at association setup time is supported;
  c) which ST algorithms, if any, are supplied with the implementation, and whether the implementation can use additional ST algorithms provided by the user.
- **Static requirements**: The system shall conform to:
  d) Conform to the abstract syntax definition of the APDUs defined in clause 9.

     Support distinguished encoding rules specified in Recommendation X.690 with object identifier {joint-iso-ccitt asn1(1) ber-derived(2) distinguished-encoding(1)} and object descriptor "Distinguished encoding of a single ASN.1 type" for the purpose of generating and interpreting application protocol information (e.g. CMISE protocol information); this is in addition to the use of the BER in the presentation layer for encoding STASE-ROSE PDUs.
  e) Support the ACSE protocol defined in Recommendation X.227, to establish and release association.

- **Dynamic requirements**: The system shall:
  - f) conform to the element of procedure defined in clause 9;
  - g) conform to the mappings to the used services for which conformance is claimed, as defined in clauses 9 and clause 10.

12 SRPM state tables

This clause defines a single STASE-ROSE-protocol-machine (SRPM) in terms of a state table. The state table shows the interrelationship between the state of an application association, the incoming events that occur in the protocol, the actions taken, and, finally the resultant state of the application-association.

The SRPM state table does not constitute a formal definition of the SRPM. It is included to provide a more precise specification of procedure defined in clause 9.

This clause contains the following tables:

a) Table 12-1 specifies the abbreviated name, source and name/description of each incoming event. The sources are:
   - STASE-ROSE user (SR-user);
   - ACSE (ACSE);
   - presentation service-provider (PS-provider).

b) Table 12-2 specifies the abbreviated name of each state of the SRPM.

c) Table 12-3 specifies the abbreviated name, target, name/description of each outgoing event. The targets are:
   - STASE-ROSE user (SR-user);
   - ACSE (ACSE);
   - presentation service-provider (PS-provider).

d) Table 12-4 specifies the predicates.

e) Table 12-5 specifies the SRPM state table using the abbreviations of the above table.

Table 12-1/Q.813 – Incoming event list

Abbreviated name	Source	Name and description
AA-ESTAB	ACSE	Positive A-ASSOCIATE response primitive or positive A-ASSOCIATE confirm primitive
SRreq	SR-user	SR-TRANSFER request primitive
APDUua	SRPM-peer	Unacceptable APDU as user data on a P-DATAind event
P-DATAind	PS-provider	P-DATA indication primitive
AA-REL	ACSE	Positive A-RELEASE response primitive or A-RELEASE confirm primitive
ABORTind	ACSE	A-ABORT indication primitive or A-ABORT-P indication primitive.

Table 12-2/Q.813 – SRPM states

Abbreviated name	Name and description
STA01	Idle; unassociated
STA02	Associated

Table 12-3/Q.813 – Outgoing event list

Abbreviated name	Target	Name and description
SRind	SR-user	SR-TRANSFER indication primitive
P-DATAreq	PS-provider	P-DATA request primitive
ABORTreq	ACSE	A-ABORT request primitive

Table 12-4/Q.813 – Predicates

Code	Name and description
p1	Unacceptable APDU and local policy is to drop APDU
p2	Unacceptable APDU and local policy is to abort association

Table 12-5/Q.813 – SRPM state table

Incoming events	STA01	STA02
AA-ESTAB	STA02	
SRreq		P-DATAreq STA02
P-DATAind		SRind STA02
APDUua		p1: STA02 p2: ABORTReq STA01
AA-REL		STA01
ABORTind		STA01

12.1 Conventions

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

In the state table, a blank cell represents a combination of an incoming event and a state that is not defined for the SRPM.

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

A mandatory action list contains:

a) optionally one or more outgoing events; and

b) a resultant state.

A conditional action list contains:

a) a predicate expression comprising predicates and Boolean operators (Ø represents the Boolean NOT); and

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

12.2 Actions to be taken by SRPM

The SRPM state table defines the actions to be taken by the SRPM in terms of an optional outgoing event and the resultant state of the application association.

12.2.1 Invalid intersections

A blank cell indicates an invalid intersection of an incoming event and state. If such an intersection occurs, one of the following actions is taken:

a) if the incoming event comes from the SR-user, any action taken by the SRPM is a local matter;

b) if the incoming event is related to a received APDU, PS-provider, ACSE, the SRPM issues an ABORTReq to ACSE.

12.2.2 Valid intersections

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

a) if the cell contains a mandatory action list, the SRPM takes the action specified;

b) if a cell contains one or more conditional actions lists, for each predicate expression that is true, the SRPM takes the action specified. If none of the predicate expressions are true, the SRPM takes one of the actions specified in 12.2.1.

13 Remote-Operations-Protocol-Machine state tables

This clause is an extension of Annex A "ROPM state tables" in Recommendation X.229. This clause provides the remote-operations-protocol-machine transfer-part (ROPM-TR) state table, if STASE-ROSE is included in the application context and RTSE is not included.

This clause imports the definitions, conventions and states defined in Recommendation X.229. (Refer to Recommendation X.229 for a description of the information). This clause contains the following tables:

a) Table 13-1 specifies the incoming events received from the STASE-ROSE service-provider (SR-provider) by ROSE in addition to those specified in A.1/X.229;

b) Table 13-2 specifies the outgoing events in addition to those specified in A.4 /X.229;

c) Table 13-3 specifies the ROPM-TR state table, if STASE-ROSE is included and RTSE in not included in the application context.

Table 13-1/Q.813 – Incoming event list

Abbreviated name	Source	Name and description
SR-TransInd	SR-provider	SR-TRANSFER indication primitive

Table 13-2/Q.813 – Outgoing event list

Abbreviated name	Target	Name and description
SR-TransReq	SR-provider	SR-TRANSFER request primitive

Table 13-3/Q.813 – ROPM-TR state table for transfer by STASE-ROSE

Incoming Events	STA100	STA200
AA-ESTAB	STA200	
TRANSreq		SR-TransReq STA200
SR-TransInd		TRANSind STA200
AA-REL		STA100
AA-ABreq		ABORTreq STA100
ABORTind		AA-Abind STA100

ANNEX A

Secure CMISE

This annex describes the usage of STASE-ROSE for the implementation of Secure Telecommunications Network Management applications. Figure 2 shows the model for an application context involving ACSE, CMISE, ROSE and STASE-ROSE. This annex defines the application context, association establishment rules and conformance for Secure CMISE.

A.1 Application context

The application context provided here is taken from clause 9.

The application context name where application entity is composed of SMASE, CMISE, ROSE, STASE and ACSE shall have the following object identifier value assignment:

{itu-t recommendation q8xx(8xx) stase(1) stase-application-context (2) secureTMNContext(1)}

and the following object-descriptor value "Secure TMN Interactive Application Context".

A.2 Association establishment rules

Recommendation X.710 defines the parameters for association establishment for CMISE. In addition this Recommendation requires that the association parameters defined in 7.4.2 be exchanged if negotiation of security parameters is desired at association setup.

As specified in clause 8, STASE-ROSE provides ACSE with any proposed values for (some of) the encryption parameters. The mechanism by which STASE-ROSE informs ACSE is an implementation matter and is not addressed by this Recommendation. This information will be carried in the ACSE user information field using the EncryptionParametersSelection defined in clause 5. During the same phase the CMISE may also provide ACSE with information relevant to the peer CMISE. All such information is carried in the ACSE user information field. The ACSE user information field consists of a SEQUENCE OF EXTERNAL. This Recommendation specifies that information supplied by STASE-ROSE, if any, shall be in the first EXTERNAL, followed by the EXTERNAL for CMISE if any, followed by the EXTERNAL for SMASE if any.

A.3 Conformance

A conforming Secure CMISE system shall comply with the following requirements.

A.3.1 Static requirements

a) The system shall comply with all requirements defined in 8.1/X.711.

b) The system shall support Distinguished Encoding Rules defined in Recommendation X.690.

c) The system shall support the STASE-ROSE protocol defined in clause 10.

A.3.2 Dynamic requirements

a) The system shall comply with all requirements defined in 8.2/X.711.

b) The system shall support the STASE-ROSE procedures defined in clause 7 and 9.5.

ANNEX B

ASN.1 Syntax defined in this Recommendation

This annex gathers the various ASN.1 syntax definitions provided in this Recommendation.

B.1 Abstract syntax for public key authenticator

The following module for public key authentication, to be carried in the Authentication-value field of the authentication FU of ACSE when peer entity authentication with public key encryption is required.

```
STASE-ROSE-Authentication-value {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0)
abstractSyntax(1) stase-authentication-value(0) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS everything

IMPORTS

SenderId, ReceiverId, Signature, SignatureCertificate
FROM Secure-Remote-Operations-APDUs {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0) stase-
data(2)};
```

```
Authentication-value ::= CHOICE {
                                explicit            [0] ExplicitAuthenticator,
                                gssAuthenticator    [1] GssAuthenticator
    -- to be used only if the two communicating entities use GSS-API.
            }

ExplicitAuthenticator ::= SEQUENCE {
                            senderId        [0] SenderId,
                            receiverId      [1] ReceiverId,
                            time            [3] GeneralizedTime,
                            encryptedSymmetricKey [4] INTEGER    OPTIONAL,
    -- a symmetric encryption key encrypted with the receiver's public key
                            signature       [5] Signature,
    -- the sender's signature of the preceding fields encoded as ASCII characters
                            certificate     [6] SignatureCertificate        OPTIONAL
    -- the sender's public key certificate for the key used for the signature
    }

GssAuthenticator ::= SEQUENCE {
                    gssMechanism            [0] OBJECT IDENTIFIER   OPTIONAL,
                    gssInitialContextToken  [1] OCTET STRING
                    }

END
```

This Recommendation assigns the ASN.1 object identifier value:

{itu-t recommendation q(17) q813(813) stase(1) abstractSyntax(1) stase-authentication-value(0) }

as an abstract syntax name for the set of all presentation data values each of which is a value of ASN.1 type.

STASE-ROSE-Authentication-value.Authentication-value.

The corresponding object descriptor value shall be "STASE-ROSE-Authenticator".

B.2 Abstract syntax for negotiation of security parameters

The following module for negotiation of security parameters, to be used in the UserInformation field of ACSE is registered.

```
STASE-A-ASSOCIATE-Information
{itu-t recommendation q(17) q813(813) stase(1) stase-pci(0) stase-userinfo(1)}

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS everything

IMPORTS

SenderId, ReceiverId, Signature, KeyId , PublicKeyCertificate, EncryptionCertificate, SignatureCertificate,
EncryptedAuthenticatedSymmetricKey
FROM Secure-Remote-Operations-APDUs {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0)
stase-data(2)};

EncryptionParametersSelection ::= SET
        {symmetricKeyIds        [0] SET OF KeyId                    OPTIONAL,
         publicKeyIds           [1] SET OF KeyId                    OPTIONAL,
         sealKeyIds             [2] SET OF KeyId                    OPTIONAL,
         signatureKeyIds        [3] SET OF KeyId                    OPTIONAL,
         passwordIds            [4] SET OF KeyId                    OPTIONAL,
```

```
        initializationVector        [5] OCTET STRING (SIZE(8))              OPTIONAL,
        feedBackBits                [6] INTEGER (1..63)                     OPTIONAL,
  -- for k-bit output feedback mode or k-bit cipher feedback mode of DES
        symmetricAlgorithms         [7] SET OF OBJECT IDENTIFIER            OPTIONAL,
        publicKeyAlgorithms         [8] SET OF OBJECT IDENTIFIER            OPTIONAL,
        signatureAlgorithms         [9] SET OF OBJECT IDENTIFIER            OPTIONAL,
        sealAlgorithms              [10] SET OF OBJECT IDENTIFIER           OPTIONAL,
        hashAlgorithms              [11] SET OF OBJECT IDENTIFIER           OPTIONAL,
        keyDigest                   [12] OCTET STRING (SIZE(8..64))         OPTIONAL,
        -- for verification of public keys
        blockSize                   [13] INTEGER                            OPTIONAL,
        -- for square mod-n hashing
        keySizes                    [14] SET OF INTEGER                     OPTIONAL,
        -- for RSA
        publicKeys                  [15] SET OF SEQUENCE
                                            {modulus        INTEGER,
                                             exponent       INTEGER
                                            }                               OPTIONAL,
        sequenceNumber              [16] INTEGER                            OPTIONAL,
        timeStamp                   [17] GeneralizedTime                    OPTIONAL,
        encryptedKey                [18] OCTET STRING (SIZE(64..128))       OPTIONAL,
        -- symmetric session key, encrypted with Key-Encryption-Key
        encryptedSymmetricKey       [19] INTEGER                            OPTIONAL,
        -- symmetric session key, encrypted with the receiver's public key
        keyEncryptionKey            [20] SEQUENCE (SIZE (1..3)) OF KeyId    OPTIONAL,
        -- one to three symmetric keys used for encrypting a session key
        keyListIds                  [21] SET OF KeyListId                   OPTIONAL,
        -- list of encryption keys that can be used during the association
        encryptionCertificate       [22] SET OF EncryptionCertificate       OPTIONAL,
-- X.509 certificates or certification paths of the sender's public keys used for encryption only
        signatureCertificate        [23] SET OF SignatureCertificate        OPTIONAL,
-- X.509 certificates or certification paths of the sender's public keys used for digital signatures only--
        encryptedAuthenticatedSymmetricKeys [24] SET OF
        EncryptedAuthenticatedSymmetricKey                                  OPTIONAL,
-- symmetric session key, encrypted with the receiver's public key and signed with sender's key--

        macAlgorithms               [25] SET OF OBJECT IDENTIFIER           OPTIONAL,
        publicKeyCertificate        [26] SET OF PublicKeyCertificate        OPTIONAL,
-- X.509 certificates or certification paths of the sender's public keys with no usage restrictions--
        ...
        }
-- EncryptionParametersSelection is optionally used during association setup to negotiate which algorithms and other
-- encryption parameters will be supported during the association. It is not used in STASE-ROSE PDUs.--

KeyListId ::= CHOICE    {identifier     OBJECT IDENTIFIER,
                         name           GraphicString,
                         number         INTEGER
                        }

END
```

This Recommendation assigns the ASN.1 object identifier value:

{itu-t recommendation q(17) q813(813) stase(1) abstractSyntax(1) stase-userinfo(1)}

as an abstract syntax name for the set of all presentation data values each of which is a value of ASN.1 type.

STASE-A-ASSOCIATE-Information.EncryptionParametersSelection

The corresponding object descriptor value shall be "STASE-ROSE-User-Information".

B.3 Abstract syntax definition of APDUs

In addition to the ASN.1 Types defined in Recommendation X.229, the following types are defined for STASE-ROSE.

```
Secure-Remote-Operations-APDUs {itu-t recommendation q(17) q813(813) stase(1) stase-pci(0) stase-data(2)}

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS everything

IMPORTS

ROSEapdus
FROM Remote-Operations-ADPUs {joint-iso-ccitt remote-operations(4) apdus(1)}

AE-title
FROM ACSE-1 {joint-iso-ccitt association-control (2) abstract-syntax(1) apdus(0) version(1)}

DistinguishedName
FROM InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework (1)}

-- the referenced module and corresponding syntax are found in Annex D/X.711 – 1998.

Certificate, CertificationPath
FROM AuthenticationFramework {joint-iso-ccitt ds(5) modules(1) authenticationFramework(7)};

SR-APDU ::= CHOICE
        { clear                     [0] ROSEapdus,
          simpleConfidential        [1] OCTET STRING,
          confidential              [2] Enciphered ,
          simplePublicEnciphered    [3] SimplePublicEnciphered,
          publicEnciphered          [4] PublicEnciphered ,
          hashed                    [5] HashedROSEpdu ,
          sealed                    [6] SealedROSEpdu ,
          signed                    [7] SignedROSEpdu ,
          confidentialSigned        [8] ConfidentialSigned,
          confidentialMAC           [9] ConfidentialMAC,
          confidentialSealed              [10] ConfidentialSealed,
          gssToken                  [11] GssToken,
          ...
         }
Enciphered ::= SEQUENCE
                    {encrypted              OCTET STRING,
                     encryptionParameters   EncryptionParameters OPTIONAL
                    }

-- encrypted represents the DER encoded and encrypted ROSE PDU.
-- encryptionParameters represents the parameters used for encryption.

SimplePublicEnciphered ::= CHOICE
                                { integers    SEQUENCE OF INTEGER,
                                  string      OCTET STRING
                                }

-- SimplePublicEnciphered represents the DER encoded and public key encrypted ROSE PDU.
-- A large PDU may be broken into smaller blocks, each of which may be encrypted
-- as an INTEGER. The size of such blocks depends on the public key encryption algorithm
```

```
-- used and on the size of the public key; specification of such block sizes is outside the
-- scope of this Recommendation.
-- In some cases the result of public key encryption may be represented as an OCTET STRING.

PublicEnciphered ::= SEQUENCE
                        {publicEncrypted   SimplePublicEnciphered,
                         encryptionParameters     EncryptionParameters OPTIONAL
                        }

-- publicEncrypted represents the DER encoded and public key encrypted ROSE PDU.
-- encryptionParameters represents the parameters used for encryption.

Hash ::= SEQUENCE{
                        hashValue               OCTET STRING (SIZE(8..64)),
                        encryptionParameters    EncryptionParameters OPTIONAL
                       }

-- hashValue represents the message digest resulting from hashing the DER encoded
-- ROSE PDU.
-- encryptionParameters represents the parameters used for the hashing algorithm.

HashedROSEpdu ::= SEQUENCE
                {data       OCTET STRING,
                 hash       CHOICE { hash           Hash,
                                     simpleHash     OCTET STRING (SIZE (8..64))
                                    }
                }

-- data represents the DER encoded ROSE PDU.
-- hash represents the hash value either as a simple OCTET STRING or the Hash
-- structure defined above.

Seal ::= SEQUENCE
                {sealValue              OCTET STRING (SIZE(8..64)),
                 encryptionParameters   EncryptionParameters OPTIONAL
                }

-- sealValue represents the seal value for the DER encoded ROSE PDU.
-- encryptionParameters represents the parameters used by the seal generation algorithm.

SealedROSEpdu ::= SEQUENCE
                {data               OCTET STRING,
                 seal               CHOICE    {seal           Seal,
                                               simpleSeal     OCTET STRING (SIZE(8..128))
                                                }
                }

-- data represents the DER encoded ROSE PDU.
-- seal represents the seal value either as a simple OCTET STRING or the Seal structure
-- defined above.

Signature ::= SEQUENCE
                        {signatureValue         SEQUENCE (SIZE(1..4)) OF INTEGER,
                         encryptionParameters   EncryptionParameters OPTIONAL
                        }

-- signatureValue represents the signature for the DER encoded ROSE PDU.
-- encryptionParameters represents the parameters for the signature algorithm.
```

```
SignedROSEpdu ::= SEQUENCE
        {data           OCTET STRING,
         signature      CHOICE {signature               [1] Signature,
                                simpleSignature         [2] SEQUENCE (SIZE(1..4)) OF INTEGER
                                }
        }

-- data contains the DER encoding of the ROSE PDU.
-- signature represents the signature of the DER encoded ROSE PDU, either as a simple
-- INTEGER or the Signature structure defined above.

ConfidentialSigned ::= SEQUENCE
                { encrypted  OCTET STRING,
                  signature  CHOICE {signature          [1] Signature,
                                     simpleSignature    [2] SEQUENCE (SIZE(1..4)) OF INTEGER
                                     }
        }

-- encrypted represents the encryption of the DER encoded ROSE PDU.
-- signature represents the signature of the DER encoded ROSE PDU in either a simple form
-- or as Signature type defined above.

ConfidentialMAC ::= SEQUENCE
        { encrypted   OCTET STRING,
          mac         CHOICE {mac           [1] Hash,
                              simpleMAC     [2] OCTET STRING (SIZE (8..64))
                              }
        }

-- encrypted represents the encryption of the DER encoded ROSE PDU.
-- mac represents the MAC of the DER encoded ROSE PDU in either a simple form
-- or as Hash type defined above.

ConfidentialSealed ::= SEQUENCE
        { encrypted    OCTET STRING,
          seal         CHOICE {sealed        [1] Seal,
                               simpleSealed  [2] OCTET STRING (SIZE (8..64))
                               }
        }

-- encrypted represents the encryption of the DER encoded ROSE PDU.
-- seal represents the seal of the DER encoded ROSE PDU in either a simple form
-- or as Seal type defined above.

EncryptionParameters ::= SET
        {symmetricKeyId         [0] KeyId                       OPTIONAL,
         publicKeyId            [1] KeyId                       OPTIONAL,
         sealKeyId              [2] KeyId                       OPTIONAL,
         signatureKeyId         [3] KeyId                       OPTIONAL,
         passwordId             [4] KeyId                       OPTIONAL,
         initializationVector   [5] OCTET STRING (SIZE(8))      OPTIONAL,
         feedBackBits           [6] INTEGER (1..63)             OPTIONAL,
 -- for k-bit output feedback mode or k-bit cipher feedback mode of DES
         symmetricAlgorithm     [7] OBJECT IDENTIFIER           OPTIONAL,
         publicKeyAlgorithm     [8] OBJECT IDENTIFIER           OPTIONAL,
         signatureAlgorithm     [9] OBJECT IDENTIFIER           OPTIONAL,
         sealAlgorithm          [10] OBJECT IDENTIFIER          OPTIONAL,
         hashAlgorithm          [11] OBJECT IDENTIFIER          OPTIONAL,
         keyDigest              [12] OCTET STRING (SIZE(8..64)) OPTIONAL,
-- for verification of public keys
         blockSize              [13] INTEGER                    OPTIONAL,
```

```
-- for square mod-n hashing
        keySize                 [14] INTEGER                            OPTIONAL,
-- for RSA
        publicKey               [15] SEQUENCE
                                        {modulus    INTEGER,
                                         exponent   INTEGER
                    }                                                   OPTIONAL,
        sequenceNumber          [16] INTEGER                            OPTIONAL,
        timeStamp               [17] GeneralizedTime                    OPTIONAL,
        encryptedKey            [18] OCTET STRING (SIZE(64..128))       OPTIONAL,
-- symmetric session key, encrypted with Key-Encryption-Key
        encryptedSymmetricKey   [19] INTEGER                            OPTIONAL,
-- symmetric session key, encrypted with the receiver's public key
        keyEncryptionKey        [20] SEQUENCE (SIZE (1..3)) OF KeyId
                                                                        OPTIONAL,
-- one to three symmetric keys used for encrypting a session key
        publicKeyCertificate    [21]    PublicKeyCertificate            OPTIONAL,
-- X.509 certificate or certification path of the sender's public key with no usage restrictions
        encryptionCertificate   [22] EncryptionCertificate      OPTIONAL,
-- X.509 certificate or certification path of the sender's public key used for encryption only
        signatureCertificate    [23] SignatureCertificate               OPTIONAL,
-- X.509 certificate or certification path of the sender's public key used for digital signatures only
        encryptedAuthenticatedSymmetricKey
                                [24] EncryptedAuthenticatedSymmetricKey         OPTIONAL,
-- symmetric session key, encrypted with the receiver's public key and signed with sender's key
        macAlgorithm            [25] OBJECT IDENTIFIER                  OPTIONAL,

        ...
        }

-- EncryptionParameters is an extensible type that is used as a catch-all for any
-- parameters that may be used by any of the STs. In most applications only a small
-- number, if any, of the components of EncryptionParameters will be used.

KeyId ::= CHOICE        {
                         name       GraphicString,
                         number     INTEGER
                        }

PublicKeyCertificate ::= CHOICE {certificate            [0] Certificate,
                                 certificationPath      [1] CertificationPath
                                 }

EncryptionCertificate ::= CHOICE {certificate           [0] Certificate,
                                  certificationPath     [1] CertificationPath
                                  }

SignatureCertificate ::= CHOICE {certificate            [0] Certificate,
                                 certificationPath      [1] CertificationPath
                                 }

EncryptedAuthenticatedSymmetricKey ::= SEQUENCE {
                                encryptedSymmetricKey           INTEGER,
            -- symmetric session key, encrypted with the receiver's public key
                                time                    GeneralizedTime,
                                sender                  SenderId,
                                receiver                ReceiverId,
                                signature               Signature
-- the signature is computed over ASCII representation of the preceding four fields with the sender's private key.
                                }
```

```
SenderId ::= CHOICE {
                        identifier      [1] DistinguishedName,
                        name            [2] GraphicString,
                        application     [3] AE-title                    }
ReceiverId ::= SenderId

GssToken::=CHOICE {
        micToken        [1]     MicToken ,
        wrapToken       [2]     OCTET STRING
        }

MicToken::=SEQUENCE {
                rosePDU     [1]     OCTET STRING ,
                token       [2]     OCTET STRING
        }

END
```

B.4 Abstract syntax object identifier

This Recommendation assigns the following object identifier:

{itu-t recommendation q(17) q813(813) stase(1) abstractSyntax(1) stase-data(2)}

as an abstract syntax name for the set of presentation data values each of which is a value of the ASN.1 type.

Secure-Remote-Operations-APDUs.SR-APDU

where the ROSE PDUs argument components are filled by the user of ROSE.

The corresponding Object descriptor value shall be "STASE-ROSE-Data".

B.5 Application contexts names

The application context name where application entity is composed of SMASE, CMISE, ROSE, STASE-ROSE and ACSE shall have the following object identifier value assignment:

{itu-t recommendation q(17) q813(813) stase(1) stase-application-context (2) secureTMNContext(0)}

and the following object-descriptor value "Secure-TMN-Interactive-Application-Context".

The application context name where application entity is composed of X.500, ROSE, STASE-ROSE and ACSE shall have the following object identifier value assignment:

{itu-t recommendation q(17) q813(813) stase(1) stase-application-context (2) secureDirectoryContext(1)}

and the following object-descriptor value "Secure–Directory-Application-Context".

APPENDIX I

Monotonically increasing time for security

This Recommendation specifies the use of monotonically increasing time for some security purposes. This appendix describes a possible construction of such a time parameter. It is provided for illustrative purposes only. Other approaches may be possible.

A real system clock may suffer various degradations:

- its rate may fluctuate, causing it to be ahead or behind the true time;
- it may stop ticking for some time;
- it may lose the current time, in which case it defaults to some fixed time well in the past, this loss of current time may or may not be accompanied by stoppage.

This appendix describes the construction of a clock, to be used by security mechanisms, that provides uninterrupted service even if any of the mishaps described above occur to the real system clock.

In this appendix we distinguish among four types of time:

1) GMT is the "astronomically correct" time.
2) System Clock (SC) is time shown by the system clock.
3) Virtual Time (VT) is the only time used by security mechanisms (and possibly by other system components).
4) External Time (ET) is the time that appears in an incoming PDU.

VT is read every time an outgoing PDU containing a timestamp is generated, and every time an incoming PDU containing a timestamp is received (as well as for other purposes outside the scope of this Recommendation). Every time the VT is read it is first updated, then the updated value is provided in response to the read request. The updated value is also stored in non-volatile memory. The procedure for updating the VT is defined by following pseudo-code:

if:

$$VT < SC$$

then:

$$VT = SC$$

else:

$$VT = VT + 1 \text{ tick}$$

where 1 tick is the smallest amount by which the (virtual) clock can be incremented; it should be small enough that the maximum possible rate of (virtual) ticking should be higher than the peak rate at which the VT is read. Typically 1 tick may be 10 ms.

If the SC stops, or is reset to its default value, VT continues to "tick" at a "virtual" rate, corresponding to the frequency of its usage. VT catches up with the real time after the SC is updated to GMT.

In order to allow for substantial System Clock (SC) drift (e.g. 30 minutes) between consecutive resets of the SC to GMT, systems may accept PDUs with ET that differs by up to twice the allowed drift (e.g. 1 hour) from the VT. The exact value of this permissibility parameter can be adjusted to match the characteristics of the clocks of the communicating systems. Of course, with drifting clocks delay detection will be reduced to a broader granularity.

In order to accommodate catastrophic SC failures (extended stoppage and/or loss of current time) a second, more generous (e.g. 4 hours) permissibility parameter may be introduced. If a PDU arrives with ET value between the two permissibility parameters it would still be accepted, but the event might be logged in a security audit trail. If the incoming PDU has an ET which is outside the limit allowed by second permissibility parameter, then a security alert might be issued in addition to logging the event in a security audit trail; the decision on whether to continue, release or abort the association in this case is a matter of local security policy.

All the security audit trail logs can be done with VT. This assures that the relative order of events is strictly maintained.

Every time the SC is reset to GMT, the event may be logged in the security audit trail. The log may contain the values of the SC and VT before and right after the update. This information may be useful to correlate VT and GMT for a security audit trail analysis.

APPENDIX II

Negotiation of security algorithms example

Figure II.1 illustrates a possible scenario for the negotiation of security algorithms[5].

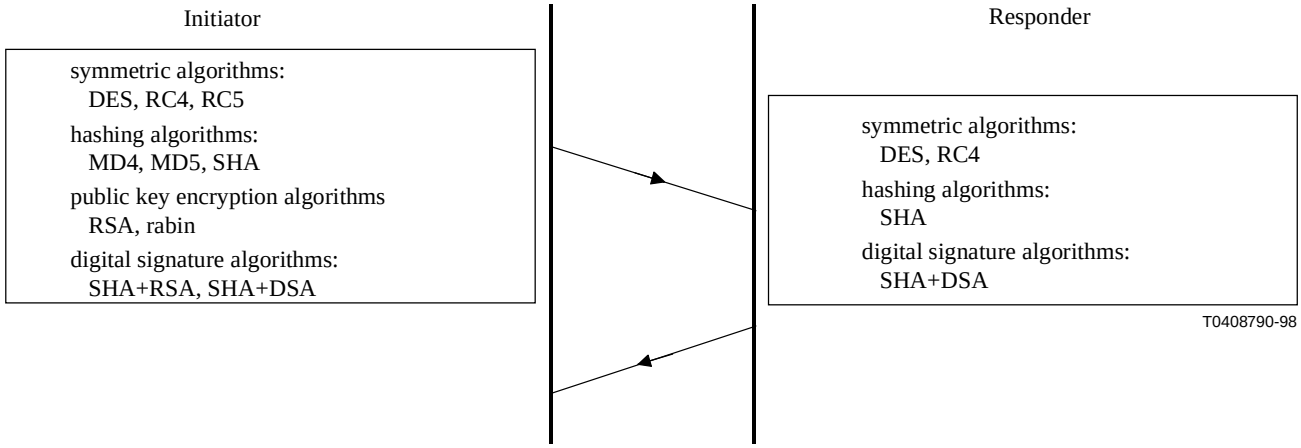

Figure II.1/Q.813 – Negotiation of security algorithms

[5] RABIN (M. O.), Digital Signatures and Public Key Functions as Intractable as Factorization, *MIT Laboratory for Computer Science*, Technical Report, MIT/LCS/TR-212, January 1979.

RIVEST (R. L.), The MD4 Message Digest Algorithm RFC 1320, April 1992.

RIVEST (R. L.), The RC4 Encryption Algorithm, *RSA Data Security Inc*., March 1993.

RIVEST (R. L.), The RC5 Encryption Algorithm, *Dr. Dobb's Journal*, Version. 20, No. 1, pp. 146-148 January 1995.

At the end of the exchange illustrated in Figure II.1, the initiator may decide that the sets of algorithms that the responder is ready to support for the proposed association are unacceptable. In that case the initiator will reject the association. If the initiator is willing to accept the sets of algorithms proposed by the responder, then the association will proceed with the following defaults:

- symmetric algorithm: DES, since it is the default specified in this Recommendation and it is among the negotiated options for symmetric algorithms.
- hashing algorithm: SHA since it is the only mutually acceptable hashing algorithm.
- public key encryption algorithm: No public key encryption algorithm can be used during this association since none has been agreed upon.
- digital signature algorithm: SHA+DSA since it is the only mutually acceptable digital signature algorithm.
- digital seal algorithm: MD5+DES since this is the default specified in this Recommendation and the negotiation did not involve digital seal algorithms.

APPENDIX III

GSS-API use with STASE-ROSE

GSS-API is a high-level API for integration of communication security services. The latest version of the API (GSS-API version 2) is documented in RFC 2078.

The use of GSS-API can give several benefits for OSI-stack vendors that wants to implement STASE-ROSE. First of all, as GSS-API is a high-level API, it gives a very simple way of integrating security services for application implementers. Secondly, the use of GSS-API for STASE-ROSE can ensure that security algorithms or even entire security mechanisms can be changed without having to modify STASE-ROSE.

This appendix describes how STASE-ROSE can realize its cryptographic functionality (security transformations upon ROSE PDUs.) through the use of GSS-API (*Generic Security Services API*). This appendix describes the use of GSS-API during the different phases of communication.

III.1 Association Establishment phase

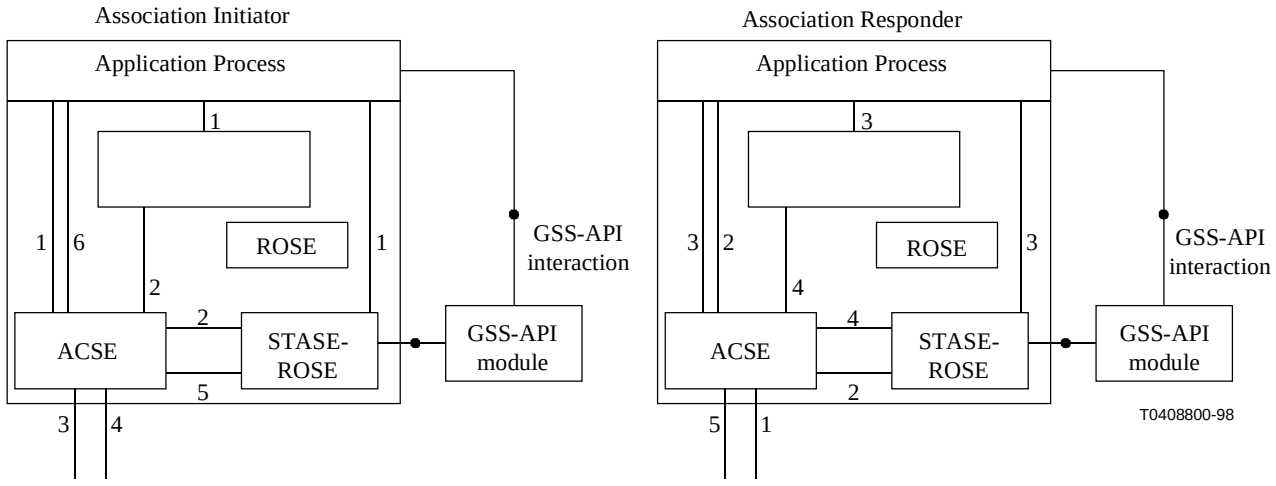

Figure III.1/Q.813 – Use of GSS-API with STASE-ROSE at association establishment time

Figure III.1 is based on Figure 4. It indicates a possible way of using GSS-API during the association establishment phase. It shows that both the application process and STASE-ROSE will need to have access to the same supporting GSS-API cryptographic module. The application process will use the GSS-API module for authentication purposes during association establishment, while STASE-ROSE will use the GSS-API module during the data transfer phase. A description of the interaction between the different components in Figure III.1 at the initiator side (left) and the responder side (right) is provided below.

The following describes the interaction on the association initiator:

a) The application performs GSS_aquire_cred() to acquire its credentials from the GSS-API module.

b) The application calls GSS_init_sec_context() to initiate a security context with a specified association responder. The application must decide on a set of security parameters for the association (e.g. whether to apply unilateral or mutual authentication, whether message sequence protection and replay protection is required). The GSS-API module will return an initial_context token to the application.

c) The application will initiate an ACSE A-ASSOCIATE request to the association responder, providing ACSE with the initial_context token to be carried in the Authentication-value field. The token structure syntax proposed in 2.1 should in this case be supported by ACSE. At the same time, the application may provide to STASE-ROSE information relevant to data protection during the data transfer phase (as a minimum the GSS-API context_handle parameter should be provided as a security context reference). This step is analogous to step 1 as described in 8.1.1.

d) The remaining steps are similar to step 2-6 of 8.1.1.

The following describes the interaction on the association responder:

a) Steps 1 and 2 are identical to the two first steps in 8.1.2.

b) When the application receives an A-ASSOCIATE indication, the application will call GSS_aquire_cred() to acquire it's credentials from the GSS-API module (if the application has not already acquired its credentials at this point in time). The application will thereafter call GSS_accept_sec_context() with the token received from the initiator (within the ACSE Authentication-value field) as one of the input parameters. The GSS-API module will authenticate the initiator by verifying that the token is valid. If mutual authentication is requested by the initiator, the application will receive from the GSS-API module a second token that must be transmitted back to the initiator.

Steps 3, 4 and 5 will be performed as described in 8.1.2. As part of the A-ASSOCIATE response provided in step 3, the application will provide the second token (in the case of mutual authentication) to be carried in the Authentication-value field. Again the token structure syntax proposed in 2.1 should be supported by the ACSE A-ASSOCIATE response Authentication-value parameter.

Security context negotiation

As part of the initial-context token exchange at association establishment time, the initiator and target GSS-API modules will negotiate (transparent to STASE-ROSE) a common valid set of integrity and confidentiality algorithms for the established security association. By default, the negotiated valid set of algorithms will always be the largest common set of algorithms that is supported by both parties. This algorithm negotiation is done automatically by the communicating GSS-API modules without being controlled by the GSS-API user (application). This level of negotiation is sufficient for interoperability reasons, however it does not give the communicating applications any flexibility to, for example, restrict the number of valid algorithms further.

To allow for a more flexible security policy, an option would therefore be to add a second negotiation mechanism at the STASE-ROSE level (external to the GSS-API modules), using the negotiation parameters defined in 5.3. In particular, the following negotiation parameters would be relevant:

– symmetricAlgorithm;
– publicKeyAlgorithm;
– signatureAlgorithms;
– sealAlgorithms;
– hashAlgorithms.

This negotiation facility would mean that two STASE-ROSE entities can agree upon the use of a set of algorithms that is a subset of the set of algorithms that is being negotiated by the GSS-API modules. A prerequisite in this case is that the STASE-ROSE entities know what algorithms their local GSS-API modules support. The principles for negotiation are the same as for STASE-ROSE in general.

III.2 Data transfer phase

Figure III.2 is based on Figure 7. It indicates the use of GSS-API during the data transfer phase. The interactions between the different components at the initiator side (left) and the responder side (right) will be the same as described in 8.4.1 and 8.4.2.

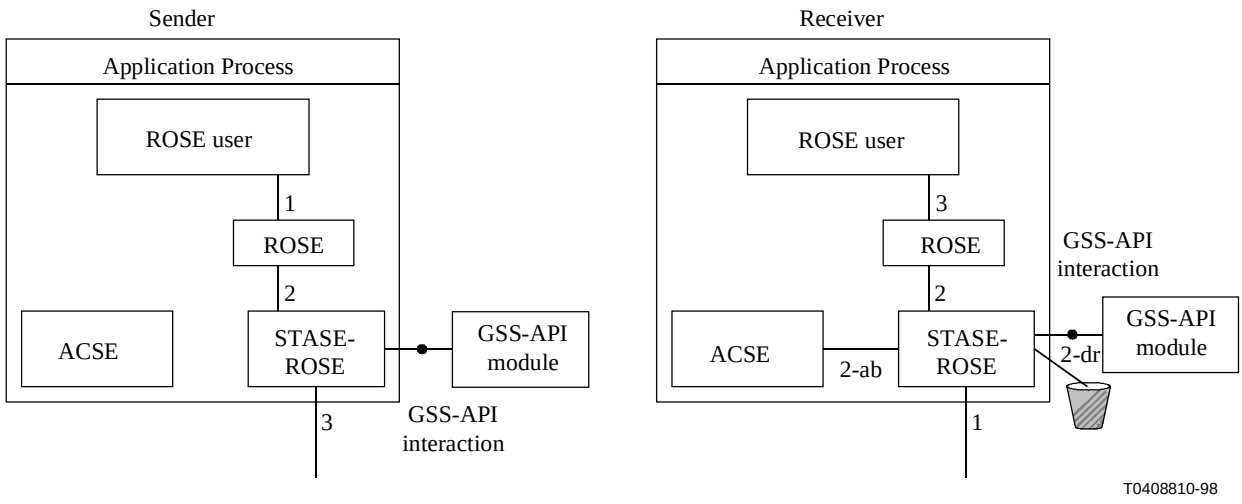

Figure III.2/Q.813 – Use of GSS-API with STASE-ROSE during data transfer

During this phase, each STASE-ROSE ASE will interface with its local GSS-API module by using the GSS-API primitives GSS_wrap() and GSS_get_mic() for providing security transformations. At this point in time, each application must have provided to their local STASE-ROSE ASE the GSS-API context handle which is needed as an input parameter to all GSS_wrap()/GSS_get_mic() function calls. In addition, the communicating GSS-API modules and the STASE-ROSE ASEs (if additional context negotiation was used during the association establishment phase – refer to 3.1.1) have already negotiated what are the set of alternative algorithms for the association and thereby the possibilities for Quality of Protection (QoP).

For each message that is sent, ROSE will request a certain level of protection from STASE-ROSE. It is not discussed here how ROSE decides what level of protection is needed (it should preferably get this information from the local application).

At a minimum, ROSE must inform STASE-ROSE of what type of encryption to use, if any. STASE-ROSE receive this information through the SR-TRANSFER parameter "Encryption-Type". These requests must be mapped onto a gss_wrap()/gss_get_mic() function call. Each time some kind of confidentiality protection (with or without integrity protection) is requested, STASE-ROSE must use of the gss_wrap() function. Each time integrity or non-repudiation protection is requested without any confidentiality protection, STASE-ROSE must use the gss_get_mic() function.

However, it is not sufficient for the STASE-ROSE using entity to know whether to use gss_wrap() or gss_get_mic(). The initiating STASE-ROSE entity must also know what quality of protection level (QoP-level) to ask for when calling these functions (qop_req input parameter of gss_wrap() and gss_get_mic()). STASE-ROSE can in principle decide about the Quality of Protection Level in two ways:

1) STASE-ROSE is informed about the required quality of protection by ROSE through the SR-TRANSFER parameter "Encryption-Parameters" (refer to 7.4.4 in STASE-ROSE specification).

2) If the SR-TRANSFER "Encryption-Parameters" parameter is not used, the default level of protection for "Encryption-Type" indicated by SR-TRANSFER will be used. In this case, it is assumed that the STASE-ROSE entities already have agreed upon a default level of protection at context establishment time.

Whatever solution used, it is important to ensure that STASE-ROSE selects a QoP level that stays within the limits of protection that was negotiated at association establishment time. The implementer of STASE-ROSE must therefore decide what should happen if for example the SR-TRANSFER primitive from ROSE requests a higher Quality of Protection level than the established context is able/allowed to provide.

After a gss_wrap() token or a gss_get_mic() token has been created, it will be inserted into the STASE-ROSE protocol using the gssToken protocol field defined in 2.3. The receiving entity will verify the tokens through calls to gss_unwrap() or gss_verify_mic().

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	**Switching and signalling**
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Programming languages