INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.815

(02/2000)

SERIES Q: SWITCHING AND SIGNALLING

Specifications of Signalling System No. 7 – Q3 interface

Specification of a security module for whole message protection

ITU-T Recommendation Q.815

(Formerly 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
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
BROADBAND ISDN	Q.2000–Q.2999

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

ITU-T Recommendation Q.815

Specification of a security module for whole message protection

Summary

This ITU-T Recommendation specifies an optional security module to be used with ITU-T Recommendation Q.814, Specification of an Electronic Data Interchange Interactive Agent, that provides security services for whole Protocol Data Units (PDUs). In particular, the security module supports non-repudiation of origin and of receipt, as well as whole message integrity.

Source

ITU-T Recommendation Q.815 was prepared by ITU-T Study Group 4 (1997-2000) and approved under the WTSC Resolution 1 procedure on 4 February 2000.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. 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 turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSC Resolution 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

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. 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, 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

Page

ITU-T Recommendation Q.815

Specification of a security module for whole message protection

1 Scope

The security module provides security services for whole Protocol Data Units (PDUs). In particular, it supports non-repudiation of origin and of receipt, as well as whole message integrity.

In the context of EDI-based TMN transactions, on the sender's side, it accepts as input the output of the EDI translator, performs the requested security transformations, and provides the resulting octet string to the Interactive Agent (IA).

On the receiving side it receives from the IA an octet string that it interprets as security module PDU. It then proceeds to verify the validity of the underlying message. In the case of integrity protection, if the message is valid, then it passes that message (without the message integrity code) to the EDI translator.

The security module keeps a log of all the messages it receives from the IA. It provides those messages, along with an indication, for each message, representing the result of the verification procedure. This log is available to the local EDI user.

The specifics of the log, as well as the interface to the log are a local matter outside the scope of this ITU-T Recommendation. The behaviour of the security module when verification fails is also a local matter outside the scope of this ITU-T Recommendation.

Figure 1 below, duplicated from Figure 2/Q.814, is used herein as a reference.

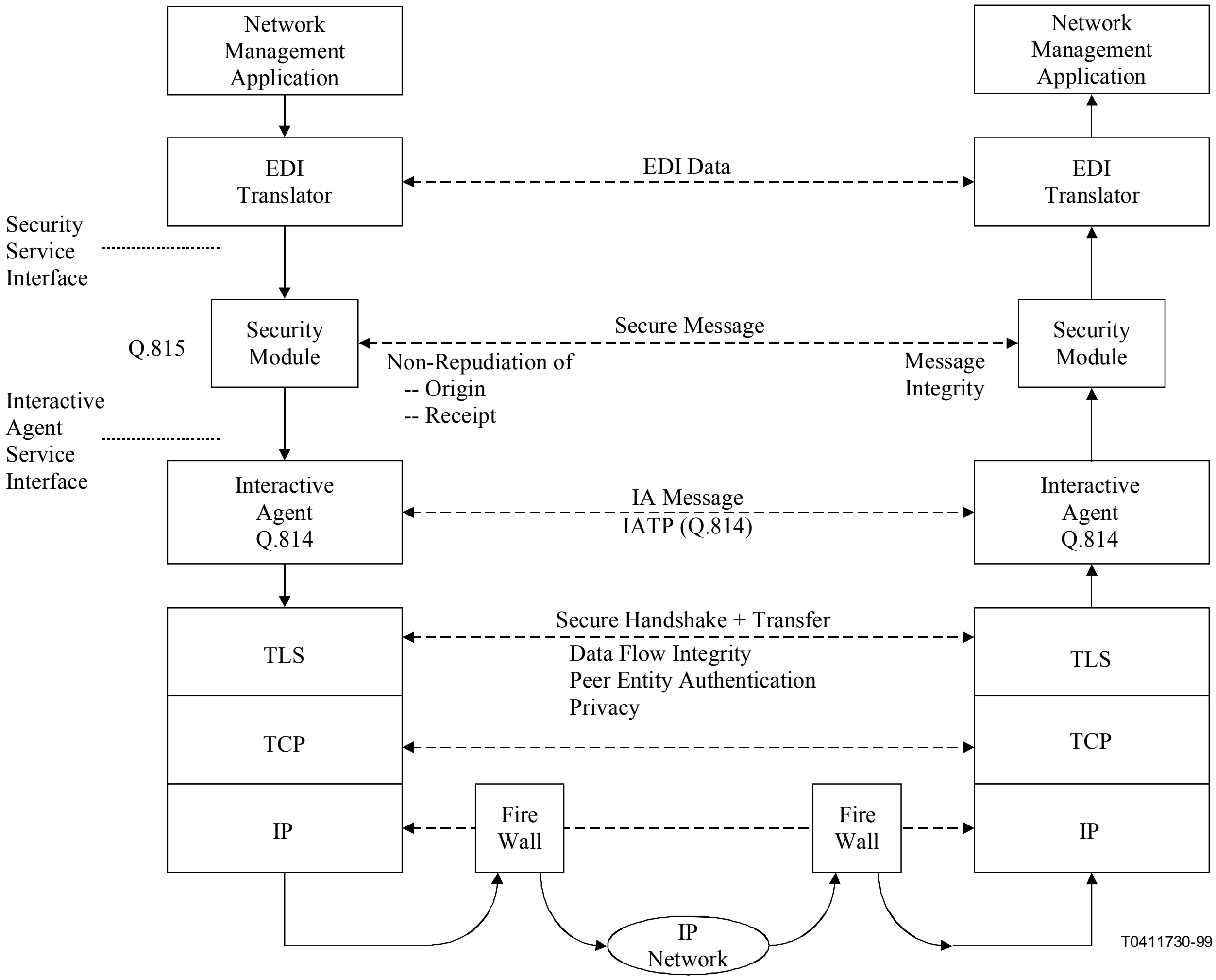

Figure 1/Q.815 – Message Flow Relationship With Message Security Services

2 References

2.1 Normative 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.

- ITU-T Recommendation Q.812 (1997)/Amd.3 (2000), *Upper layer protocol profiles for the Q.3 and X interfaces – Amendment 3: Protocol profile for electronic communications interactive agent.*
- ITU-T Recommendation Q.814 (2000), *Specification of an electronic data interchange interactive agent.*
- ITU-T Recommendation X.509 (1997) | ISO/IEC 9495-8:1998, *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): Parameterization 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).*

ISO – International Organization for Standardization:

- ISO 3166:(All parts), *Codes for the representation of names of countries and their subdivisions.*

2.2 Informative references

- *Directory Implementors Guide (Version 11) (1998).*

3 Definitions

This ITU-T Recommendation defines the following terms:

3.1 application protocol data unit: A packet of data exchanged between two application programs across a network. This is the highest level view of communication in the OSI seven layer model and a single packet exchanged at this level may actually be transmitted as several packets at a lower layer as well as having extra information (headers) added for routing, etc.

3.2 distinguished encoding rules: A restricted form of Basic Encoding Rules defined in (ITU-T X.690) to eliminate the options in BER.

3.3 electronic data interchange: The exchange of documents in standardized electronic form, between organizations, in an automated manner, directly from a computer application in one organization to an application in another.

3.4 electronic data interchange for administration, commerce and transport: The syntax is an ISO standard (ISO 9735), and was adopted by the United Nations (UN) as the basis for the international development of business messages for EDI (UN/EDIFACT). EDIFACT grew out of a desire to bring together previous standards and ASC X12.

3.5 interactive agent: The Interactive Agent (IA) supports the exchange of Electronic Data Interchange (ANSI ASC X12 EDI or EDIFACT) transactions between peer entities within the telecommunications industry.

3.6 RSA: RSA is a public-key cryptosystem developed by Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman in 1977 in an effort to help ensure Internet security. A cryptosystem is simply an algorithm that can convert input data into something unrecognizable (encryption), and convert the unrecognizable data back to its original form (decryption). RSA encryption techniques are described in ITU-T Recommendation X.509.

3.7 secure hash algorithm, revision 1: A 160-bit hash function, mandated by the National Institute for Standards Technology (NIST), with security mechanisms similar to MD5. SHA-1 is defined by the United States Government in FIPS 180-1. It is a mechanism to reduce a lengthy text message to a short digest of 160 bits that is both one-way (i.e. non-reversible) and not susceptible to collisions from multiple different texts. Because SHA generates a 160-bit hash (message digest) it is much safer from brute-force cryptographic attacks than MD5.

Digests are best thought of as the digital fingerprint of a message. It is a relatively fast, low-overhead and secure algorithm. SHA-1 can be used to support integrity protection (by itself), or for non-repudiation (in conjunction with public key encryption). The SHA-1 Message Digest Algorithm is described in Appendix II.

4 Abbreviations

This ITU-T Recommendation uses the following abbreviations:

APDU	Application Protocol Data Unit
ASC	Accredited Standards Committee
DER	Distinguished Encoding Rules (of ASN.1)
EDI	Electronic Data Interchange
EDIFACT	Electronic Data Interchange for Administration, Commerce and Transport
IA	Interactive Agent
IP	Internet Protocol
PDU	Protocol Data Unit
RSA	Rivest, Shamir, Aldeman
SHA-1	Secure Hash Algorithm, Revision 1
SM	Security Module
SR	STASE ROSE
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TMN	Telecommunications Management Network

5 Conventions

The following conventions are used: References to clauses, subclauses, annexes and appendices refer to those items within this ITU-T Recommendation unless another specification is explicitly listed.

6 Q.815 details

6.1 Security Module Message Types

Q.815 Security Module specifies the following guidelines:

Message Integrity and/or Non-Repudiation services may be provided by the Security Module (SM). If the SM provides Message Integrity, it uses SHA-1 to produce the Message Digest (MD). If the SM provides Non-Repudiation of Origin, it uses SHA-1 to produce an MD then uses the RSA digital signature mechanism. Non-Repudiation of Receipt is provided by a mechanism described in this ITU-T Recommendation. A fourth set of security enhancements is available through the use of STASE-ROSE (see ITU-T Recommendation Q.813).

6.1.1 Message Integrity Service

The direct user supplies the SM with an EDIFACT or ASC X12 EDI message. The SM will compute a message digest utilizing the EDIFACT or ASC X12 EDI data as input to the digest algorithm. The

SM DER encodes a Hashed Message in accordance with 7.1. The IA utilizes the DER encoded octet string as the contents of an Enhanced Message, as defined in ITU-T Recommendation Q.814.

6.1.2 Non-Repudiation of Origin Service

The direct user supplies the SM with an EDIFACT or ASC X12 EDI message. The SM will compute a message digest utilizing the EDIFACT or ASC X12 EDI data as input to the digest algorithm. The SM encrypts a DER encoding of the message digest according to the digital signature algorithm utilizing the private key of the sender. The SM DER encodes a Signed Message in accordance with 7.2. The IA utilizes the DER encoded octet string as the contents of an Enhanced Message, as defined in ITU-T Recommendation Q.814.

6.1.3 Non-Repudiation of Receipt Service

The direct user supplies the SM with a unique message identifier and the date and time that the corresponding message was received. Refer to 7.3.2 for information regarding the unique identifier and date/time. Optionally the direct user may be, through bilateral agreement, required to supply a message digest or a digital signature as a component of the receipt. The digest or signature must correspond to the uniquely identified message described above. The SM DER encodes a IA Received Message in accordance with 7.3. The IA utilizes the DER encoded octet string as the contents of an Enhanced Message, as defined in ITU-T Recommendation Q.814.

6.2 General Characteristics

Message Digests and Digital Signatures are computed on the clear (unencrypted) text of the EDIFACT/ASC X12 EDI/General String messages.

Digital Certificates shall be compatible with ITU-T Recommendation X.509 version 3.

7 General Syntax

The general syntax of the Security Module message is as follows:

```
SecureMessage ::=  CHOICE {
hashedMessage      [0] EXPLICIT HashedMessage,
signedMessage      [1] EXPLICIT SignedMessage,
messageReceipt     [2] EXPLICIT IaReceiptMessage,
sr-APDU            [3] EXPLICIT SR-APDU,
…
}
```

NOTE – An ASN.1 Module containing the general syntax and each of the specific syntaxes defined in this clause can be found in Annex A.

7.1 Hashed Messages

Message Integrity services are provided by the *HashedMessage* defined as:

```
HashedMessage :: =             SEQUENCE {
      hashedVersion            Version DEFAULT v1999 ,
      hashAlgorithmIdentifier  AlgorithmIdentifier,
      hashedContent            HashedContent,          -- Data
      messageDigest            OCTET STRING ( SIZE (20) )
  }
HashedContent :: =       CHOICE {
hashedContent1           GeneralString,
hashedContent2           IA5String
}
```

7.1.1 Object Identifiers Referenced by Hashed Messages

In the hashedMessage the value to be used for the hashAlgorithmIdentifier is:

```
hashAlgorithmIdentifier OBJECT IDENTIFIER :: = {
        iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }    -- SHA-1
```

7.1.2 Value Information for Hashed Messages

hashedContent consists of an EDIFACT, ASC X12 EDI or another string message.

messageDigest consists of the SHA-1 digest of the hashedContent.

7.2 Signed Messages

Non-Repudiation of Origin services are provided by the *SignedMessage* defined as:

```
SignedMessage :: = SEQUENCE {
        signedVersion                       Version DEFAULT v1999 ,
        signedDigestAlgorithms              SET OF AlgorithmIdentifier,
        signedContent                       SignedContent,        -- Data
        signerInfos                         SET OF SEQUENCE {
                signerVersion               Version DEFAULT v1999 ,
                issuerAndSerialNumber       SEQUENCE {
                        issuerCountry       SEQUENCE OF SET OF SEQUENCE {
                                country             OBJECT IDENTIFIER,
                                countryValue        PrintableString
                        },
                        issuerOrg           SEQUENCE OF SET OF SEQUENCE {
                                organizationName    OBJECT IDENTIFIER,
                                organizationValue   PrintableString
                        },
                        serialNumber        INTEGER
                },
                signedDigestAlgorithm       AlgorithmIdentifier,
                digestEncryptionAlgorithm   AlgorithmIdentifier,
                encryptedDigest             OCTET STRING
        }
}
SignedContent :: =          CHOICE {
signedContent1              GeneralString,
signedContent2              IA5String
}
```

7.2.1 Object Identifiers Referenced by Signed Messages

In the SignedMessage the value to be used for the signedDigestAlgorithms is:

```
signedDigestAlgorithms OBJECT IDENTIFIER :: = {
   iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }    -- SHA-1
country OBJECT IDENTIFIER :: = { 2 5 4 6 }
organizationName OBJECT IDENTIFIER :: = { 2 5 4 10 }
```

In the SignedMessage the value to be used for the signedDigestAlgorithm is:

```
signedDigestAlgorithm OBJECT IDENTIFIER :: = {
   iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }    -- SHA-1
```

In the SignedMessage the value to be used for the digestEncryptionAlgorithmIdentifier is:

```
digestEncryptionAlgorithmIdentifier OBJECT IDENTIFIER :: = {
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 1 } -- RSA
```

7.2.2 Value Information for Signed Messages

signedContent consists of an EDIFACT, ASC X12 EDI, or another string message.

countryValue is the two character representation of the issuer's country name as found in ISO 3166.

organizationValue is the certificate issuer's organization name.

serialNumber is a unique number assigned to the certificate by the issuer.

encryptedDigest consists of a DER encoding of the SHA-1 digest of the signedContent, RSA encrypted with the sending party's private key.

signedDigestAlgorithms – the set shall be comprised of a single member (AlgorithmIdentifier for the message digest algorithm). The algorithm shall be the same in both signedDigestAlgorithms and signedDigestAlgorithm.

signerInfos – the set shall be comprised of a single member.

The private/public key pair associated with the digital certificate used for authentication is to be used for signing/verifying.

issuerCountry – the set shall be comprised of a single member.

issuerOrg – the set shall be comprised of a single member.

NOTE – The RSA Digital Signature Algorithm converts the INTEGER data type of a digital signature to an OCTET STRING.

7.3 IA Receipt Message

Non-Repudiation of Receipt services are provided by the *IaReceiptMessage* defined as:

```
IaReceiptMessage :: =      SEQUENCE {
        uniqueIdentifier   OCTET STRING, -- A unique identifier within the message
        dateTimeStamp      PrintableString ( SIZE(15) ),
        enhancements       Enhancements OPTIONAL
}
Enhancements :: =  CHOICE {
        withDigest   [0] EXPLICIT WithDigest,
        withDigSig   [1] EXPLICIT WithDigSig
}
WithDigest :: =                        SEQUENCE {
        receiptDigestAlgorithm         OBJECT IDENTIFIER,
        receiptMessageDigest           OCTET STRING
}
WithDigSig :: =                        SEQUENCE {
        receiptSignatureAlgorithm      OBJECT IDENTIFIER,
        receiptDigitalSignature        OCTET STRING
}
```

7.3.1 Object Identifiers Referenced by IA Receipt Messages

In the IaReceiptMessage the value to be used for the receiptDigestAlgorithm is:

```
receiptDigestAlgorithm OBJECT IDENTIFIER :: = {
  iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }  -- SHA-1
```

In the IaReceiptMessage the value to be used for the receiptSignatureAlgorithm is:

```
receiptSignatureAlgorithm OBJECT IDENTIFIER :: = {
  iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1 5 }
```

7.3.2 Value Information for IA Receipt Messages

uniqueIdentifier consists of any field in the data of the message that is unique from one message to the next (e.g. ASC X12 EDIs ISA segment). Peer entities will determine the data field to be utilized for this purpose via a mechanism outside the scope of this ITU-T Recommendation.

dateTimeStamp is formatted as follows: CCYYMMDDhhmmssz

where: CC = century
YY = year
MM = month
DD = day
hh = hour
mm = minute
ss = second
z = alpha time zone indicator.

A blank character for the time zone indicator indicates local observed time. The date and time in the timestamp should be the time that the receiving party received the complete message being receipted. The use of coordinated universal time (Z) in the receipt is recommended to avoid any time zone ambiguities.

receiptMessageDigest is the digest of the message being receipted. If the message was of the SimpleHashed format, this digest is that which was received with the message and was verified by the recipient. If the original message was of any other data type, this digest will need to be calculated by the recipient using the EDIFACT/ASC X12 EDI/General String message data as input to the specified message digest algorithm.

receiptDigitalSignature is computed by formatting a concatenation of the uniqueIdentifier of the EDIFACT/ASC X12 EDI/General String message, the dateTimeStamp of the receipt (15 octets) and the Digital Signature received with the original message. This octet structure is then signed by encrypting it in accordance with the RSA digital signature algorithm using SHA-1 as the digest algorithm and the private key of the receipt generating party. The party receiving the receipt should construct a concatenation of: the uniqueIdentifier (from either the transmitted message or the body of the receipt), the dateTimeStamp of 15 octets (from the body of the receipt) and the encryptedDigest (digital signature) from the original message. This concatenated structure, together with the digital signature of the receipt message itself (receiptDigitalSignature) should be passed to the digital signature verification algorithm. The public key of the receipting party is used to verify the signature. This receipt type requires that the original message being receipted for must have been of the type SignedMesssage.

7.4 Stase-Rose PDU

The syntax for this message type is imported from ITU-T Recommendation Q.813.

ANNEX A

ASN.1 Production Module

```
SecurityModule {itu-t(0) recommendation(0) q(17) q815(815) sm(0) messages(0)}
DEFINITIONS IMPLICIT TAGS :: = BEGIN

-- EXPORTS everything

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

```
-- Useful Types

AlgorithmIdentifier :: = SEQUENCE {
                algorithm OBJECT IDENTIFIER,
                parameters NULL
}

Version :: = INTEGER
v1999 Version :: = 0


-- General Syntax

SecureMessage :: = CHOICE {
hashedMessage       [0] EXPLICIT HashedMessage,
signedMessage       [1] EXPLICIT SignedMessage,
messageReceipt      [2] EXPLICIT IaReceiptMessage,
sr-APDU             [3] EXPLICIT SR-APDU,
…
}


-- Hashed Message Syntax

HashedMessage :: =                          SEQUENCE {
        hashedVersion               Version DEFAULT v1999 ,
        hashAlgorithmIdentifier     AlgorithmIdentifier,
        hashedContent               HashedContent,              -- Data
        messageDigest               OCTET STRING ( SIZE (20) )
        }
HashedContent :: =          CHOICE {
hashedContent1              GeneralString,
hashedContent2              IA5String
}


-- Signed Message Syntax

SignedMessage :: =                          SEQUENCE {
        signedVersion                       Version DEFAULT v1999 ,
        signedDigestAlgorithms              SET OF AlgorithmIdentifier,
        signedContent                       SignedContent,              -- Data
        signerInfos                         SET OF SEQUENCE {
                signerVersion               Version DEFAULT v1999 ,
                issuerAndSerialNumber       SEQUENCE {
                        issuerCountry       SEQUENCE OF SET OF SEQUENCE {
                         country            OBJECT IDENTIFIER,
                 countryValue               PrintableString
                },
                issuerOrg                   SEQUENCE OF SET OF SEQUENCE {
                 organizationName           OBJECT IDENTIFIER,
                 organizationValue          PrintableString
                },
                serialNumber                INTEGER
        },
        signedDigestAlgorithm               AlgorithmIdentifier,
        digestEncryptionAlgorithm           AlgorithmIdentifier,
        encryptedDigest                     OCTET STRING
  }
}
```

```
SignedContent :: =        CHOICE {
signedContent1            GeneralString,
signedContent2            IA5String
}

-- Receipt Message Syntax

IaReceiptMessage :: =     SEQUENCE {
        uniqueIdentifier  OCTET STRING, -- A unique identifier within the message
        dateTimeStamp     PrintableString ( SIZE(15) ),
        enhancements      Enhancements OPTIONAL
}

Enhancements :: =         CHOICE {
  withDigest              [0] EXPLICIT WithDigest,
  withDigSig              [1] EXPLICIT WithDigSig
}

WithDigest :: =                   SEQUENCE {
  receiptDigestAlgorithm          OBJECT IDENTIFIER,
  receiptMessageDigest            OCTET STRING
}

WithDigSig :: =                           SEQUENCE {
  receiptSignatureAlgorithm       OBJECT IDENTIFIER,
  receiptDigitalSignature         OCTET STRING
}

END
```

APPENDIX I

Non-normative references

- ISO 9735:1988, *Electronic data interchange for administration, commerce and transport (EDIFACT) – Application level syntax rules.*
- ANSI ASC X12: American National Standards Institute (ANSI) Accredited Standards Committee X12. The Committee was chartered by ANSI in 1979 to develop uniform standards for the electronic interchange of business documents.

APPENDIX II

The SHA-1 Message Digest Algorithm

II.1 Introduction

For a message of length < 2^{64} bits, the SHA-1 produces a 160-bit condensed representation of the message called a message digest. The message digest is used during generation of a signature for the message. The SHA-1 is also used to compute a message digest for the received version of the message during the process of verifying the signature. Any change to the message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

The SHA-1 is designed to have the following properties: it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.

II.2 Bit strings and integers

The following terminology related to bit strings and integers will be used:

a) A hex digit is an element of the set {0, 1, ..., 9, A, ..., F}. A hex digit is the representation of a 4-bit string.

Examples: 7 = 0111, A = 1010

b) A word equals a 32-bit string which may be represented as a sequence of 8 hex digits. To convert a word to 8 hex digits each 4-bit string is converted to its hex equivalent as described in a) above.

Example: 1010 0001 0000 0011 1111 1110 0010 0011 = A103FE23

c) An integer between 0 and $2^{32} - 1$ inclusive may be represented as a word. The least significant four bits of the integer are represented by the right-most hex digit of the word representation.

Example: the integer 291 = $2^8+2^5+2^1+2^0$ = 256+32+2+1 is represented by the hex word, 00000123

If z is an integer, $0 \leq z < 2^{64}$, then $z = 2^{32}x + y$ where $0 \leq x < 2^{32}$ and $0 \leq y < 2^{32}$. Since x and y can be represented as words X and Y, respectively, z can be represented as the pair of words (X,Y).

d) block = 512-bit string. A block (e.g. B) may be represented as a sequence of 16 words.

II.3 Operations on words

The following logical operators will be applied to words:

a) Bitwise logical word operations:

X ^ Y = bitwise logical "and" of X and Y.

X $\vee$ Y = bitwise logical "inclusive-or" of X and Y.

X XOR Y = bitwise logical "exclusive-or" of X and Y.

~ X = bitwise logical "complement" of X.

Example:

```
          01101100101110011101001001111011
XOR       01100101110000010110100110110111
          ---------------------------------------------
     =    00001001011110001011101111001100
```

b) The operation X + Y is defined as follows: words X and Y represent integers x and y, where $0 \leq x < 2^{32}$ and $0 \leq y < 2^{32}$. For positive integers n and m, let n mod m be the remainder upon dividing n by m. Compute $z = (x + y) \bmod 2^{32}$.

Then $0 \leq z < 2^{32}$. Convert z to a word, Z, and define Z = X + Y.

c) The circular left shift operation $S^n(X)$, where X is a word and n is an integer with $0 \leq n^{32}$, is defined by $S^n(X) = (X << n)$ OR $(X >> 32–n)$.

In the above, X << n is obtained as follows: discard the left-most n bits of X and then pad the result with n zeroes on the right (the result will still be 32 bits). X >> n is obtained by discarding the right-most n bits of X and then padding the result with n zeroes on the left. Thus $S^n(X)$ is equivalent to a circular shift of X by n positions to the left.

II.4 Message padding

The SHA-1 is used to compute a message digest for a message or data file that is provided as input. The message or data file should be considered to be a bit string. The length of the message is the number of bits in the message (the empty message has length 0). If the number of bits in a message is a multiple of 8, for compactness we can represent the message in hex. The purpose of message padding is to make the total length of a padded message a multiple of 512. The SHA-1 sequentially processes blocks of 512 bits when computing the message digest. The following specifies how this padding shall be performed. As a summary, a "1" followed by m "0"s followed by a 64-bit integer are appended to the end of the message to produce a padded message of length 512 * n. The 64-bit integer is l, the length of the original message. The padded message is then processed by the SHA-1 as n 512-bit blocks.

Suppose a message has length $l < 2^{64}$. Before it is input to the SHA-1, the message is padded on the right as follows:

a) "1" is appended. Example: if the original message is "01010000", this is padded to "010100001".

b) "0"s are appended. The number of "0"s will depend on the original length of the message. The last 64 bits of the last 512-bit block are reserved for the length l of the original message.

Example: Suppose the original message is the bit string:

01100001 01100010 01100011 01100100 01100101

After step a) this gives:

01100001 01100010 01100011 01100100 01100101 1

Since l = 40, the number of bits in the above is 41 and 407 "0"s are appended, making the total now 448. This gives (in hex):

61626364 65800000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000

c) Obtain the 2-word representation of l, the number of bits in the original message. If $l < 2^{32}$ then the first word is all zeroes. Append these two words to the padded message.

Example: Suppose the original message is as in b). Then l = 40 (note that l is computed before any padding). The two-word representation of 40 is hex 00000000 00000028. Hence the final padded message is hex:

61626364 65800000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000028

The padded message will contain 16 * n words for some n > 0. The padded message is regarded as a sequence of n blocks M_1 , M_2, ..., M_n, where each M_i contains 16 words and M_1 contains the first characters (or bits) of the message.

II.5 Functions used

A sequence of logical functions f_0, f_1, ..., f_{79} is used in the SHA-1. Each f_t, $0 \leq t \leq 79$, operates on three 32-bit words B, C, D and produces a 32-bit word as output. f_t(B,C,D) is defined as follows: for words B, C, D:

f_t(B,C,D) = (B AND C) OR ((NOT B) AND D) ($0 \leq t \leq 19$)

f_t(B,C,D) = B XOR C XOR D ($20 \leq t \leq 39$)

f_t(B,C,D) = (B AND C) OR (B AND D) OR (C AND D) ($40 \leq t \leq 59$)

f_t(B,C,D) = B XOR C XOR D ($60 \leq t \leq 79$)

II.6 Constants used

A sequence of constant words K(0), K(1), ..., K(79) is used in the SHA-1. In hex these are given by:

K = 5A827999 ($0 \leq t \leq 19$)

K_t = 6ED9EBA1 ($20 \leq t \leq 39$)

K_t = 8F1BBCDC ($40 \leq t \leq 59$)

K_t = CA62C1D6 ($60 \leq t \leq 79$)

II.7 Computing the message digest

The message digest is computed using the final padded message. The computation uses two buffers, each consisting of five 32-bit words, and a sequence of eighty 32-bit words. The words of the first 5-word buffer are labelled A, B, C, D, E. The words of the second 5-word buffer are labelled H_0, H_1, H_2, H_3, H_4. The words of the 80-word sequence are labelled W_0, W_1, ..., W_{79}. A single word buffer TEMP is also employed.

To generate the message digest, the 16-word blocks M_1, M_2, ..., M_n defined in clause 4 are processed in order. The processing of each M_i involves 80 steps.

Before processing any blocks, the $\{H_i\}$ are initialized as follows: in hex:

H_0 = 67452301

H_1 = EFCDAB89

H_2 = 98BADCFE

H_3 = 10325476

H_4 = C3D2E1F0

Now M_1, M_2, ..., M_n are processed. To process M_i, we proceed as follows:

a) Divide M_i into 16 words W_0, W_1, ..., W_{15}, where W_0 is the left-most word.

b) For t = 16 to 79 let $W_t = S^1(W_{t-3}$ XOR W_{t-8} XOR W_{t-14} XOR $W_{t-16})$.

c) Let A = H_0, B = H_1, C = H_2, D = H_3, E = H_4.

d) For t = 0 to 79 do:

TEMP = $S^5(A) + f_t(B,C,D) + E + W_t + K_t$;

E = D; D = C; C = S^{30}(B); B = A; A = TEMP;

e) Let $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$.

After processing M_n, the message digest is the 160-bit string represented by the 5 words:

H_0 H_1 H_2 H_3 H_4.

II.8 Alternate method of computation

The above assumes that the sequence W_0, ..., W_{79} is implemented as an array of eighty 32-bit words. This is efficient from the standpoint of minimization of execution time, since the addresses of W_{t-3}, ..., W_{t-16} in step b) are easily computed. If space is at a premium, an alternative is to regard

{ Wt } as a circular queue, which may be implemented using an array of sixteen 32-bit words

W[0], ..., W[15]. In this case, in hex let MASK = 0000000F.

Then processing of M_i is as follows:

a) Divide M_i into 16 words W[0], ..., W[15], where W[0] is the left-most word.

b) Let $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$.

c)
```
For t = 0 to 79 do
s = t ^ MASK;
if (t ≥ 16) W[s] =$S^1$(W[(s + 13) ^ MASK] XOR W[(s + 8) AND MASK] XOR
W[(s + 2) ^MASK] XOR W[s]);
TEMP = $S^5$(A) +$f_t$(B,C,D) +E + W[s] +$K_t$;
E = D; D = C; C = $S^{30}$(B); B = A; A = TEMP;
```

d) Let $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$.

II.9 Comparison of methods

The methods of II.7 and II.8 yield the same message digest. Although using the method of II.8 saves sixty-four 32-bit words of storage, it is likely to lengthen execution time due to the increased complexity of the address computations for the { W[t] } in step c). Other computation methods which give identical results may be implemented in conformance with the standard.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of 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 and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Geneva, 2001